Powell Rangers: Smart Mini Thermostat

Jimmy Howerton, Keerthi Radhakrishnan, Kevin Dela Merced, Nathan Olszewski

December 17, 2018

Capstone Design ECE 4440 / ECE4991

Signatures

Statement of work:

Keerthi Radhakrishnan:

At the beginning, I helped lay out some of the groundwork for the design of our project by putting out ideas about the fundamental design. For instance, I proposed the idea of switching the line with some form of triac / relay / MOSFET which we then actually pursued. I helped pick some of the parts such as the ZXMS6004DG MOSFET which we used to source the needed current for fast and accurate line switching at 3.3 volts. I also helped compute some of the component values for circuits like the MOC3043 optoisolator's LED side by doing some of the thinking about the electrical characteristics of the internal BJT of the device.

Afterwards, I worked on implementation of fixed point 32 bit arithmetic, fixed point digital filtering, fixed point PID control, SPI temperature sensor interfacing code, and PWM code. I made the design choice of switching integer cycles of the line with the 'sub timer' method that divided 1 second up into several smaller intervals of 8 cycles for the hardware PWM on the CC3220. I also designed the 32 bit fixed point arithmetic system for our project and used it to design and test the digital filters and PID code quite early on in the semester. From there, I did heavy investigation on TI RTOS and figured out how to implement our program architecture with interrupts and threading, which resulted in huge performance gains that allowed us to tack on lots of additional tasks without any noticeable performance loss.

From there, I found and implemented an interrupt-driven binary encoded rotary encoder state machine that we then used in our final design with great efficiency and quite literally spent 13 hours working on the SPI temperature sensor with Jimmy which we fully set up using hardware SPI and a manual chip select. I then integrated the SPI temperature readings directly into my fixed point code and made the rotary encoder state machine directly control the set point to the PID controller. I then pieced together the main loop of the program. This all resulted in the final product's reliable and accurate operation as shown in the demo.

Jimmy Howerton:

Being the one who came up with the idea for the project, I spent a lot of time figuring out how the project should work and what type of technologies we could use to reach those goals. This meant figuring out how triacs and the many forms of optical isolators actually worked and then choosing the specific parts that would allow them to work together to meet our goal. My team definitely helped choose parts but I would argue that I was the one who did the bulk of the decision making in this area.

After choosing parts, it was necessary to create the boards on which these parts would go. My teammates helped significantly on the production of the Low Power Board (LPB) that would slot onto the CC3220SF Launchpad, however, I did the complete ultiboard designs for the High Power Board (HPB) including figuring out how to route the high current and high voltage traces that we would need to reach our goal.

Having the boards assembled with working parts, it was necessary to implement code to make all of this work with our MCU. As such, I developed and debugged the code to interface with the SPI temperature sensor with Keerthi and the I2C LCD screen. Furthermore, I setup and debugged the code for the rotary encoder that was developed by Keerthi. Finally, I performed all soldering for the temperature sensor, screen, and rotary encoder and their subcomponents.

Nathan Olszewski:

I focused my contributions on circuit and enclosure design as my software skills are considerably weaker than my CpE major teammates. I refined many component selections including fusing, power conversion, and power connection. I found a suitable 120VAC to 3.3VDC power converter for our CC3220 microcontroller. I assisted in soldering components onto our circuit boards as well as assisted Kevin in verifying PCB footprint dimensions.

Due to the usage of 120VAC, we were required to meet safety standards defined by the National Electric Code which calls for an enclosure for our project. I found a suitable class of NEMA rated plastic enclosures to fit our size needs. I had the idea of a clear enclosure to showcase our PCB designs and more importantly display our screen within the box greatly simplifying our overall design and decreasing production time. I made all of the necessary modifications to the enclosure to accommodate our design and mounted both boards securely into the enclosure to finalize our product into a single package.

Kevin Dela Merced:

My responsibility as one of the Electrical Engineers of the group is to design our project's circuitry. We initially planned to have all our components on one PCB, but we were advised to make two separate boards for the high power and the lower power systems. My responsibility was mainly to design the low power board which included the LCD Screen, rotary encoder, small active and passive components, and ports for the cable connection between the low power board and the high power board and the temperature sensor. My job included having to look at multiple datasheets in order to understand how each component's footprint looks like and how they work. I also had to be meticulous with making sure that the low power board fits on top of the CC3220 Launchpad.

After completing the boards, I helped test a few components on both the high power board and the lower power board. For the lower power board, I tested the rotary encoder, and for the high power board, I tested the zero crossing system.

While Keerthi and Jimmy were working on the code, I helped Nathan modify our enclosure. My main responsibility in this stage was to just make plexiglass platforms so that the high power board and the lower board would be able to fit in the enclosure vertically.

Table of Contents

Table of Figures

(This should list the page of each figure used in your document, including the full caption.) Word has tools to help you do this very easily)

Abstract

While HVAC thermostats are commonplace, fewer temperature-regulation devices exist for the hobbyist in application areas such as reptile enclosures, aquariums, freezers, home brewing, etc. Our project presents an opportunity for individuals to reduce power consumption and most importantly, provide an easy way to monitor and control a localized temperature area. While devices like these exist, we planned to connect this device to a network to allow the user to regulate temperature wirelessly via a smartphone.

Background

We chose this project from the assumption that there is a more efficient way to regulate temperature in an enclosed space. Our teammate with a pet reptile recognized this problem and sought to design an automated controller for the animal's enclosure. Properly caring for reptiles can be difficult since many times they require certain parts of their cage to be heated to specific temperature ranges, and it is difficult to tell if a certain heat lamp will be able to produce the right amount of heat. We believe that automation of this task will make the users experience move enjoyable of whatever he or she needs to control the temperature of. Examples of this are noted in the Abstract.

It turns out similar devices exist for the specific purposes that we wanted to use them for, however, none of them provide a means to control the device from a phone and none of them provide a timing mechanism. For example, the iPower digital heat thermostat controller is a device that plugs into the wall, has a connected thermometer, and has a port for a heating device to plug into it like our device[1]. However, it uses seven-segment displays and physical buttons to show and control the heat and there is no way to have it run for certain periods of time. The Inkbird ITC-308 [2] digital temperature thermostat accomplishes the same goal with a higher max output of 1100w and with 2 input plugs, but is largely the same device, still void of any network connection or any way to control when it is running.

To differentiate our project from other previous products, we intended to implement wireless capabilities to our project which would allow the user to monitor and modify the temperature anywhere and even receive alerts if the temperature goes out of a certain range. Furthermore, the user can set certain time periods for the heating to run which is particularly useful for simulating a realistic environment for a reptile. A possible extension is that multiple mini-thermostats could interface with one another, allowing for more complex heating scenarios in larger cages with specific heat gradients. While other smart products essentially functions the same as our project with its programmability and wireless capabilities, our project will be using PWM to vary the intensity of our heat bulb.

We have decided that we want to use one of the TI CC SimpleLink devices that are similar to those used in the Embedded Systems class because they are the only boards that TI provides that offer Wi-Fi integration to a single chip solution. This makes setting up a standalone PCB much easier when we reach that point in the project. We have chosen the TI CC 3220SF Launchpad as the basis of our design since it comes with a number of powerful features that will significantly reduce development time but ensure an effective solution.

Since we will need to add ports to the device, we plan on using our skills from FUN 1, 2, and 3 to design a PCB header board for the CC that will contain power and temperature outlets, a switched mode power supply, an LCD display for manual control along with a potentiometer,

and a triac with an opto-isolator to allow the CC to control the incoming 120V AC from the wall. We will use another opto-isolator as a zero-crossing circuit to keep our PWM in phase with the wall, allowing us to do forward phase-dimming. For designing the software application that will be used to control the device, we plan on using our knowledge of Java from CS 2110 and our general understanding of Data Structures and Algorithms from CS 2150 and CS 4102 to build an app in Android Developer or in TI's provided SDK. For specific networking knowledge, we plan on using information learned from our CS 4457 Networks class that several of us are currently enrolled in.

Constraints

Design Constraints

One of the first constraints that we encountered while designing our PCBs was the trace width of the copper traces that would carry 120 VAC. In order to safely pass 120 VAC through, we had to increase the trace width which also meant that the size of our high power board would need to increase. While working with 120 VAC, we also had to consider the separations between the traces and pins that are connected to the 120VAC traces. To avoid potential problems with electrical arches, we increased the separation between the traces. We also distorted the pins of some components like the Triac so that they could fit in the widely separated traces. Furthermore, using 120VAC has a high possibility of producing a dangerous amount of heat. In order to alleviate this problem, we placed a heat sink on the Triac. Finally, we also had to get a NEMA rated enclosure to fit national standards since we are working with 120 VAC. While looking for enclosures, we had to rethink how our user interface would be placed since we need the LCD screen to be seen and the rotary encoder to be interactable. We fixed this problem by getting a clear enclosure which allowed easy visual access to the LCD screen and elevating the whole low power board which had the rotary encoder and drilling a round hole through the enclosure roof so that the rotary encoder could be interacted with through the top.

While working with the software, we initially assumed that the CC3220 would have a floating point unit considering the sophistication of the platform (cryptography, file systems, Internet streaming, etc). Keerthi discovered that the ARM Cortex M4 on board was not the variant with the built-in floating point unit. So either we had to use floating point and potentially take a massive performance hit through software emulation, or design and fine tune a fixed point system to achieve this level of precision manually. We chose fixed point because of all of the other things running on the machine at the same time.

Since our project utilizes zero-crossed 120VAC at 60 Hz to power our lamp, our PWM was constrained to having a maximum frequency of 60 Hz. This caused the PWM to be slow and be considerably noticeable when we tested our project with a light bulb. This problem could have been fixed with a frequency multiplier or a rectifier however, we did not have time to consider testing and implementing those fixes.

Economic and Cost Constraints

As one of the goals of our project is to save energy and therefore money by being as efficient as possible with power consumption, we also considered getting components with the most reasonable prices. This included our MCU where the myRIO MCU though it probably

would have been a good choice in our project since it can connect to Wifi, it was an extremely expensive equipment that no regular lizard owner would probably invest in. Instead we aimed for an MCU that was much cheaper and found the CC3220SF which had similar functions. The other components like the LCD Screen, voltage regulator, rotary encoder, etc. were not that expensive which alleviated our economic constraints.

In order to be appealing to general lizard pet owners, we had set our total cost to somewhere around $200. Though the price may seem high, this number does not consider automation and bulk purchases.

External Standards

Our project incorporates connection to the 120 VAC at 60 Hz line, so it is naturally a very dangerous product. To combat the danger factor of our design, we incorporated the NEMA and NEC standards to ensure safe operation of the device: we picked up a NEMA 4 rated enclosure for the header boards, and we picked up NEC rated sockets / plug outlets for the high power board. [44] [45]

Our controller was designed to be placed outside of the reptile enclosure, so not much water protection was needed there since the odds of a sudden jet of water / reptiles urinating on the box are quite low. So the biggest danger, hands down, was the threat of touching line voltage-carrying components such as the triac.The triac has pins that can be touched with fingers, so we needed to protect from shocks due to fingers touching the line. However, we needed to fit our 6 pin connector into the box, which was ~16 mm wide. This mandated an IP20 rating at the very least, where 2 specified protection against objects larger than 12.5 mm but smaller than 50 mm. [46] This led to us drilling a small hole of about 15 mm for the cable. This was on the low power side, so it was not as critical as the high power socket side. On the socket side, we aimed for a very tight fit with the sockets to prevent objects much smaller than fingers from falling through. We would estimate that the gap is around 1 mm around the edge of the side female socket and the top female socket. This resulted in the low power side being about IP20 resistance and the high power side being about IP30 - IP40 resistance. This collectively means that our design choices downgraded our NEMA 4 box to a NEMA 1 box. So in terms of safety factor, our design will prevent sudden electrocution from normal operation of the device, but that is about it.

We used I2C protocol for our MCU by sending a slave address to the slaves then sending an alternating commands and databytes. [47] We also used SPI protocol which required us to drive the chip select to low, send commands, and drive it back high. [48]

Tools Employed

In the beginning of our project, we spent an immense amount of time browsing Digikey and forming documents holding our large lists of possible items. These documents would eventually be condensed into spreadsheets that Chris used to order our parts. As such, I don't think it would be fair to say that our project could have existed without the knowledge-sharing capacity of tools such as google docs and google sheets.[3]

Having our parts was very important, but mapping them out in multisim[4] let us get a better grasp on how they related to one another. Our prior experience with multisim was enough to let us use simple components to represent our more specific components, however, we soon learned that this would not do. We had to learn how to create components in multisim using the

component wizard and how to connect these components to ultiboard footprints. Furthermore, we learned how to make the pins in multisim correlate to the proper symbology by copying the visuals from similar parts and modifying them to fit our new ones in the symbol editor.

However, to effectively use the part we needed to create footprints which we learned to do in ultiboard.[5] This meant looking at data sheets and converting their dimensions into ultiboard and adding metal contacts and holes of the right dimensions in the right locations. We messed this up a bit in the beginning, but by our third board we had all of the parts fitting properly. We also learned that changing net names in multisim will propagate into ultiboard, making it easier to keep track of which net does what, which nets should be widened to allow for higher current, and which nets need to be spaced from one another to avoid arcing.

When testing our project, we had several components that required several logical inputs and outputs. This meant learning how to better use the virtual bench to deal with logic on the logic analyzer.[6] We saw how to map the pins to certain names but also how to have the Virtual Bench automatically parse the data read from an I2C or SPI connection which helped us debug our data transfers.

Beyond the physical components of our board, we used Code Composer Studio extensively for this project.[7] While we had experience with this tool from Intro to Embedded Systems, the CC3220SF Launchpad is on an entirely different level of complexity than the MSP430 that we used in that class. This meant that we had to figure out how the CC handled setting pin configurations and had to use plenty of the TI examples to figure out how things were supposed to be laid out. The CC also required the installation of the CC3220 SDK which provides Code Composer the ability to compile code for it and we needed the TI Pinmux tool to generate the CC3220SF_LAUNCHXL.c and .h files that pertained to our specific pin usages.[8] These files were then consequently modified several times as our pin usages changed. Once we had our code fully functional, we wanted to flash it to the CC which is much more complicated than on the MSP which flashes by default. As a result, we had to learn how the TI Uniflash tool worked and use that to make our code boot from power up.[9] This functionality was still a bit flawed by the time we finished our project, with the code booting properly about 50% of the time and the other 50% having the LCD not display anything.

Ethical, Social, and Economic Concerns

Environmental Impact

Heat lamps are inherently not very environmentally friendly due to enormous power figures. We believe that our project does help cut down on the power figure problem massively because simply leaving the lamp on may result in overheating of the area. Essentially all of the heat generated past the set point in the static heating case is wasted. With our design, the power transmission is gated to easily less than 50% as soon as the temperature gets near the set point. This cuts down on the surplus heat generation and thus surplus power consumption. Furthermore, our zero crossing switching design specifically targets the issue of inrush currents that consume large amounts of power. And considering all of the components in the entire arrangement, they do not really contribute much power usage to the whole system since we

specifically aimed for the lowest possible current that will trigger / satisfy all devices of a kind. So really our power savings are not diminished in the slightest by our board designs either.

Sustainability

Sustainability was the name of the game with our design in the first place. We were specifically targeting the extension of heat bulb lifespan. Light bulb lifespan ties into material reuse which goes a long way in reducing the amount of light bulbs thrown away. This therefore would mean that our design helps improve the sustainability of incandescent light bulbs for pet heating. The idea of minimizing on-time of a heating or cooling source that is being controlled can be related to any of these types of devices to maximize their lifespan and prevent power waste.

Health and Safety

Our design is a relatively contained system. It does not send out harmful radiation or give off large amounts of heat. It also is not very easy to simply reach inside and get electrocuted since we took great care in picking a tough and safe enclosure with carefully drilled cutouts. At present however, the light given off by our design flickers quite strongly when moderate duty cycles are applied to the lamp. This could be risky for epilepsy patients or potentially reptiles if they are prone to photosensitive seizures. [43] However, further iterations of the design could easily fix the stuttering issue.

Manufacturability

Our design would have to modified significantly to become manufacturable and competitive to similar devices on the market. The flickering of a light source when moderate duty cycles are applied must be fixed but could possibly be ignored in applications not involving light. Ideally, we would eliminate the LCD interface entirely and write an android and/or iOS app to control our device remotely. This would be accomplished using WiFi provisioning in order to eliminate any user interface or controls on our device. A custom enclosure must be designed as well to decrease manufacturing time and cost. A redesign of out 120VAC board using integrated circuits would greatly decrease size and cost.

Ethical Issues

The purpose of this project is originally to alleviate the responsibilities of having to manually maintain heat and light for ectothermic pets. However, this could result in completely detaching any commitment and relations that owners would have with their pets effectively contradicting the purpose of keeping pets for companionship.

At our project's current state, the PWM flashes could potentially cause health risks to its owner and the pet as we are constrained to 60 Hz, our PWM is akin to a seizure-inducing strobe light effect.

Intellectual Property Issues

While we designed our product to fit into a niche that would make it something new, it turns out there are some overarching patents out there that we would violate by selling our product.

Our idea was to add wifi connectivity to our temperature controller to differentiate it, however, patents already exist for controlling switching of a power outlet via WiFi. In their patent labelled 'Method and device for WiFi controlled electric switch', Adir Tubi and Eyal Nuriel outline their independent claim on a wireless outlet controller comprising of an MCU using WiFi to control a relay that sends current from a mains connection to a socket adapted for use with standard electric plugs.[10] Furthermore, their patent covers time-based activation and deactivation of the outlet and dynamic updates to the software. This means that the main functionality of our product would violate their patent and the future addition of scheduled heating and OTA updates would also violate the patent.

Similarly, the patent 'Temperature Control Device', as described by author Ming-Chien Chang, covers a temperature controller utilizing an MCU and taking readings from multiple temperature sensors to control heating of an enclosure to achieve a thermostatic effect.[11] While this patent focuses more on having the MCU inside of the enclosure that is being heated, the dependent claim of the patent seems to be general enough to cover our device as well. The patent also specifically states that there is a temperature sensor on both the inside and outside of the enclosure and that a cooling device is provided so, by our omission of those components, we might be able to skirt around this patent through specificity.

Finally, the patent 'Control system for thermoelectric devices' by Anantha Krishnan Rama Rao covers control systems for thermoelectric devices using a temperature sensor connected to a microcontroller where the microcontroller drives the thermoelectric device via PWM.[12] This means that our method of switching the lamp via PWM using the SPI temperature sensor input is patented. Furthermore, via the later part of the independent claim stating 'increase the second duty cycle by the dynamic percentage when the sensed temperature exceeds the reference temperature', our algorithm for keeping the temperature stable is patented.

Detailed Technical Description of Project

What it is

Our project consists of a CC3220SF LaunchXL (CC from here on) microcontroller controlling on/off switching of a heat lamp via PWM to control the temperature detected around an SPI temperature sensor. Our original goal was to allow the setting of this functionality via WiFi but we were unable to reach this goal, instead opting for physical setting via a rotary encoder and LCD screen.

Our custom PCBs consist of one header board that we are calling the Low Power Board (LPB) directly connected to the CC via two 10 x 2 GPIO connectors to receive or deliver signals to any of the available GPIO pins on the device and one 3 x 1 GPIO connector to provide VDD and VSS to the device at 3.3VDC. The LPB provides a UI by way of a rotary encoder placed in the center of the board and an LCD screen placed in the top middle of the board. The LPB also

hosts the MOSFET that is used to power componentry on the High Power Board, described in the next paragraph.

The LPB is connected to a second custom board that we are calling the High Power Board (HPB) via a 6 inch long, 4-wire GPIO cable. The HPB hosts the input and output for 120VAC including a fuse and the mechanisms for switching this connection. This includes a triac-driver optocoupler and a triac. The board also hosts the voltage regulator that we use to power the CC, taking 120VAC and converting it to 3.3VDC. Finally, this board hosts our zero crossing componentry, including a phototransistor optocoupler which triggers interrupts on the CC to signify zero crossings of the AC line.

The LPB is also connected to a third custom board that we are calling the Temperature Board (TB) via a 1 meter long, 6-wire GPIO cable. The TB consists of our temperature sensor and a bypass capacitor to facilitate the proper functioning of this sensor.

Components Used

CC3220SF Launchpad [13]
ADT7310TRZ Temperature Sensor [35]
NHD-C0220BIZ LCD Display [14]
2x 10x2 Header Connector [15]
3 Position Header Connector [20]
Bourns PEC12R Rotary Encoder [16]
ZXMS600 MOSFET [17]
4 Conductor Multi-Conductor Cable [37]
Littelfuse Fuse [26]
MOC3043M Optoisolator Triac [32]
T3035H Triac [30]
VCE03 AC/DC Converter 3.3V [34]
FOD814A Optoisolator [24]
6 Conductor Multi-Conductor Cable [36]

How it Works and Design Decisions

Our device takes in readings from an SPI temperature sensor via a 6-pin cable, converts the temperature data to fixed point, uses this data as an input to a PID algorithm which outputs a number of half-cycles out of 8 to turn a heat lamp on, this becomes the duty cycle for the PWM. This number is out of 8 because the PWM for our device is at a frequency of 15Hz and we must use complete half cycles in order to not switch at high voltage and introduce high inrush currents into our circuit. The temperature data is also sent to the LCD display via I2C to update the ‘current temperature’ reading. The user is able to modify the set-point temperature via a rotary encoder that interfaces with the board through two gpio pins. Turning the rotary encoder clockwise increases the set temperature and turning it counter-clockwise reduces the set temperature.

We will now elaborate on the SPI temperature sensor. The ADT7310 (ADT) sensor provides accurate temperature readings within 0.5C when operating within -40C to 105C. We chose this temperature sensor because we believed that SPI was an easier protocol to debug than

something like I2C, however, due to the grace with which the virtual bench deals with logic analysis, both protocols were similarly difficult in our experience.

Our temperature sensor utilizes 8 pins: SCLK, DOUT, DIN, CS, VDD, GND, CT, INT. We utilize all of these pins except for CT and INT in our design. These two extraneous pins are used for sending hardware interrupts in the case that the temperature goes above or below a temperature threshold that is manually programmed into the board. We planned on using a similar functionality, however, believed that it would be simpler to implement this in software than to complicate our design further with two more traces and cables. Ultimately, we never reached the point of even implementing this in software.

This temperature sensor has 3 modes of operation: one-shot mode, 1 sps mode, and continuous mode. We decided to go with the one-shot mode of operation even though it requires waiting 240ms between clock readings because, in our case, temperature changes so slowly that faster readings were unnecessary. On top of that, this mode provides reduced power consumption over both of the other modes.

To implement this type of transaction via hardware SPI, we had to learn a bit about how hardware SPI works on the CC versus how the ADT expects it to work. The ADT expects to receive a single command byte, in our case 010100000 (explanation in appendix), the chip select to stay low for 240ms, then to receive two high bytes. The CC will send the first and second sets of bytes, however, by default it disables the chip select before waiting. This meant that we had to change the chip select pin from a hardware-controlled SPI pin to a software controlled GPIO pin (p50). It turns out that we had also connected MOSI to DOUT on the ADT, meaning we had to switch the wires for MOSI and MISO on the ADT end of the cable.

Upon making all of these hardware and software revisions, we were able to receive data back from the ADT, however, several layers of computation would then take place in order to get a usable value from this data. As per our configuration of the sensor, we received 2 bytes of fixed point temperature data, where the most significant bit was the sign bit, the next 8 bits were the ‘whole’ bits, and the next 7 bits were the decimal bits. These data bytes were then concatenated to create the fixed point temperature data input to a digital filter the PID controller made use of.

We made use of an exponential moving average filter with an alpha of 0.375. The classic EMA filter is a recursive digital filter with an output of the form y[n] = a x[n] + (1 - a) y[n - 1] where x[n] is the input / new sample and y[n], y[n - 1] are the current and old values of the average respectively. It functions as a low pass filter with a transfer function of H(z) = a / (1 - (1 - a) / z). So picking the alpha and the sampling frequency is important for figuring out the degree of smoothing / tracking desired. For this, the -3 dB frequency is Fs / 2pi arccos[1 - (a a) / [2(1 - a)]]. The frequency response of this type of digital filter is approximately linear only in the ‘early’ parts of the frequency response; as it gets closer to the aliasing frequency Fs / 2, the filter response warps and overshoots the expected value. As alpha decreases, the cutoff frequency and the amount of attenuation for a given frequency f both change, resulting in much stronger attenuation. At the same time, it also results in a larger phase shift so a careful tradeoff between delay and smoothing must be made. The appendix contains the derivation of the -3 dB frequency.

We chose an alpha of 0.375 in order to smooth out noise in the temperature values. We were sampling at 4 Hz, so the aliasing frequency occurred at 2 Hz. We found that the air heated up very quickly with even a 50 W bulb, so we did not want to set our cutoff frequency very low and respond slower to temperature changes, but also not so high that all noise is passed.

Admittedly, this part of picking filter behavior was not very precise at all, so we chose an alpha of 0.375 because it was 1) a sum of powers of two that could be exactly represented by our fixed point arithmetic, 2) resulted in a cutoff frequency of ~0.2 Hz and an attenuation of about ~9.5 dB at the aliasing frequency. So noise in the system would be lightly filtered to at minimum ⅓ of its original value; admittedly we could have probably dropped the coefficient to something like 0.125 to strongly attenuate almost everything. However, the denominator of the EMA transfer function is 1 - (1 - a) cos(wT) - j (1-a) sin(wT) which would result in a phase shift of arctan[(a - 1) sin(wT) / [1 - (1 - a) cos(wT)]] which increases in intensity the lower the alpha gets. The phase shift goes negative which delays our signals. Since we were not too sure about how much that would matter in practice (temperature grows slowly), we stayed conservative with the alpha. The appendix contains the frequency response analysis.

We chose to implement all of our mathematical calculations with the same fixed point format as the temperature data, so 7 bits of decimal precision (~3 digits of decimal precision). Considering that our PID duty cycle was extremely imprecise with only 9 possible values, the precision behind our calculations up until the calculation of the actual result was never going to be the bottleneck. The inherent imprecision of the output was the chief bottleneck in precision. So to compute the fixed point multiplication, we had both operands in the same format. Both numbers in a fixed point multiplication can be represented as some X = Su and Y = Sv, where u and v are the 'true' values and S is the fixed point scale factor, a power of 2 that is determined by the number of bits in the decimal place. For instance, a number like ⅝ could be represented as 5 by scaling it up by the fixed point precision of 8. In our case, the fixed point scale factor was 128. Going back to the multiplication, the product XY is S^2 uv, which is equal to the fixed point representation of the product uv, scaled up an extra time. Thus we have to divide by the scale factor back out. So using this , we computed both a x[n] and (1 - a) y[n - 1] by keeping track of a's value in fixed point as well as the old value of the average. Essentially we computed Sa, Sx[n], and calculated S^2 ax[n] / S for instance. We then summed the two products P1 = ax[n] and P2 = (1 - a) y[n - 1] which does not require any sort of scaling up or down because both operands are in the form X = Su and Y = Sv, so X + Y = S(u + v). At the end of all of this calculation, we had an updated estimate of the temperature in fixed point mode.

This fixed point temperature average was then fed directly into the PID control algorithms. The classic PID control equation in Laplace terms is Y(s) = PE(s) + IE(s) / s + sDE(s) where E(s) = SP(s) - X(s). X(s) is some input in the control system, SP(s) is the set point for the system, and Y(s) is some output for the control system, and the constants P, I, and D are weights for the proportional , integral, and derivative error terms. PID essentially generates correcting outputs based on the difference between the current state of the system and a set point, and these weights determine the magnitude of the correction through a combination of the 'error now', 'error so far' and 'error trend' to minimize the error over time. In this case, our input is the temperature reading and our output is the duty cycle of the PWM timer, which must be clamped to an integer between 0 and 8.

The control algorithm was implemented with a class and all math was done in fixed point. At the most basic level, we had to keep track of the PID coefficients in fixed point, as well as the integral term. We also had to perform all of the multiplications and additions in fixed point with the same method above. So we took the input temperature and a fixed point set point temperature, computed E = SP - I where SP and I are the set point and input, respectively as a fixed point error term. We then computed I0 + E where I0 is the previous value of the error.

Lastly we compute (E - E0) / dt where E0 is the previous value of the error. We knew that the controller sample rate would be 4 Hz since the SPI sensor could only output data that quickly, so we hard coded our dt (time step) to be 250 ms in fixed point notation. We could then compute (E - E0) / dt. So now we had the error term, the new integral of the error, and the derivative of the error. Since all of these were done in fixed point, we could then multiply these results by the fixed point PID coefficients. We were then able to compute the final PID output.

A couple of adjustments were made to the PID algorithm to eliminate some issues such as 'derivative kick', where a sudden change in the set point can cause massive spikes in the output of the controller, and 'reset windup', where the output value shoots past the maximum / minimum value of the system and has to spend time 'catching up' since it overcorrected past the allowable output range. This results in lags in the response time of the controller. To solve the 'derivative kick' problem, we keep track of the current state of the output of the controller and take the derivative of that instead. Since dE/dt = dSP/dt - dI/dt where SP, I are the set point and input, if the set point changes drastically, the error (and thus output) will follow. We can filter this out by instead ignoring these transients in the set point and assuming that the set point is constant. In that case, dE/dt = -dI/dt so we replace the derivative term sDE(s) with -sDX(s), which will prevent extreme overcorrection. To solve the 'reset windup' problem, we have to clamp the possible outputs for each individual term in the controller and the final output to the realistic minimum / maximum value for the output in the system. In our case that meant clamping the output duty cycle to [0, 8]. Since the integral term would offer the largest potential for overcorrection, we clamped that to the same values as well. This would cause the controller's state to not change past the limits of the output, solving instability issues in the PID controller. We could then move on to switching.

To perform the switching of the mains, we utilize several layers of componentry. First we utilize a pin on the CC, specifically p64 that performs hardware PWM. The low and high data from this pin is then translated to the gate of a MOSFET, more specifically, the ZXMS6004DG (ZXMS). We utilize a MOSFET in this scenario for assured functionality because our CC GPIO pins can only provide 6mA of current and our triac driver requires 5mA to switch. The ZXMS gate requires effectively 0mA to switch the source to drain. We connect the source of the ZXMS to the GND of our voltage regulator. The drain of the ZXMS is then connected to pin02, the cathode of our triac driver, the MOC3043 (MOC).

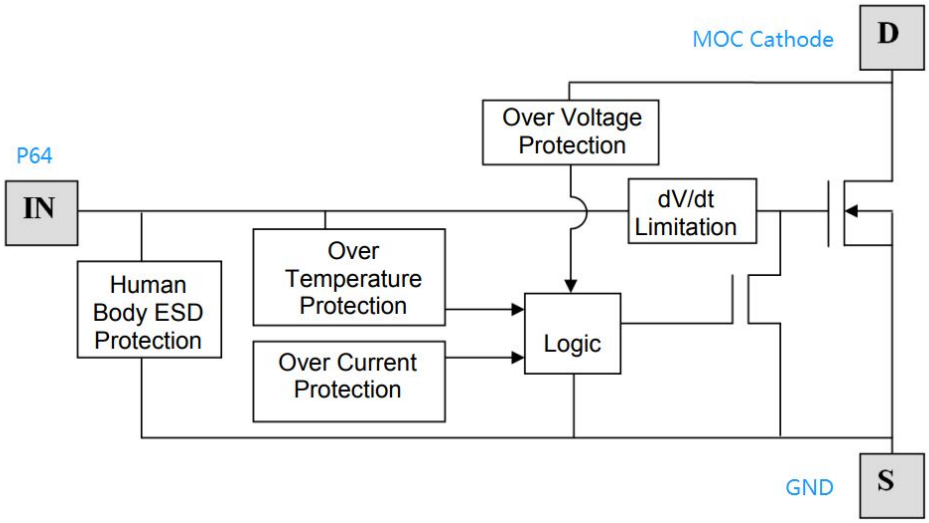


Figure 1: ZXMS MOSFET Functional Block Diagram

A 360 ohm resistor is put in series with pin01, the anode of the triac driver, which is then connected to the VDD of our voltage regulator which outputs a voltage of 3.3VDC and a maximum current of 910mA, 450mA of which could be taken by the MCU and 30mA of which could be taken to power the LCD backlight. We decided that we wanted to use 6mA of current to control the MOC, so we were able to use the Q-point provided via the MOC's datasheet to determine our resistor value. We saw that 1.15 volts across the diode would give us the desired current, meaning we could simply subtract the voltages and use Ohm's Law, 3.3-1.15 = 2.15, 2.15 = 0.06*R, R = 360 ohms.

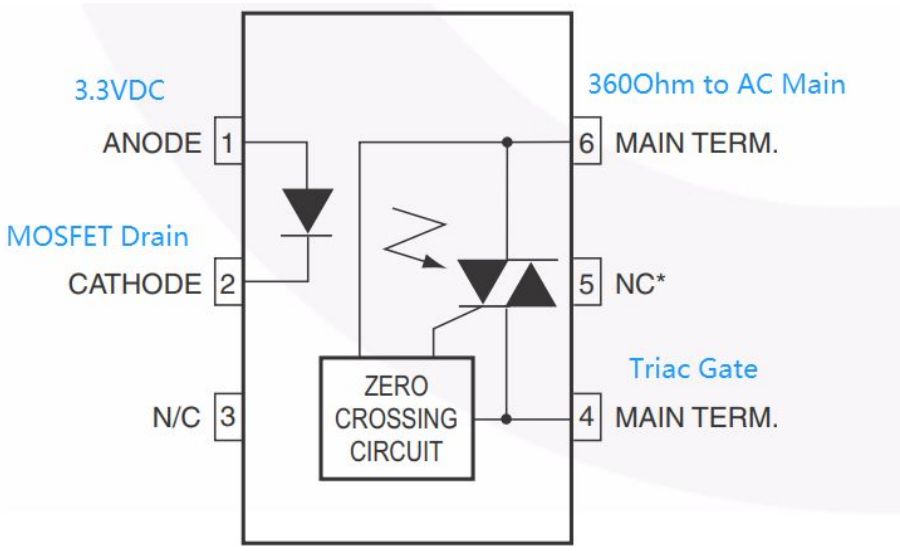


Figure 2: MOC3031 Schematic

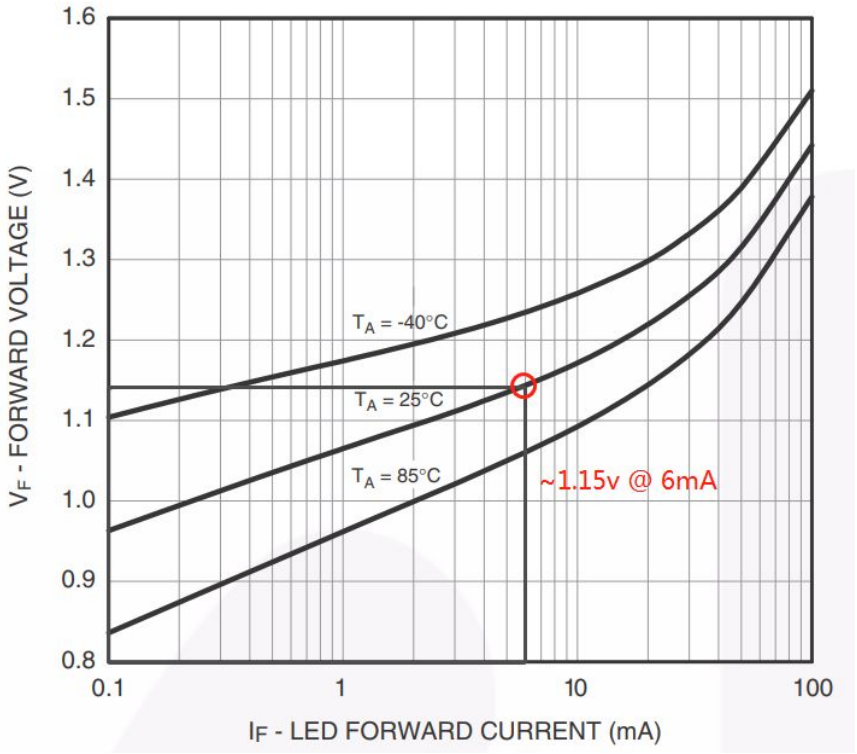


Figure 3: LED Forward Voltage vs. Forward Current

The MOC is then used to control the gate of our triac, the T3035H. This component requires 35mA of current at its gate to allow current to pass from pin A2 to pin A1, it also automatically turns off at zero crossings. We source the current here from the wall itself, connecting 120VAC to the pin04 main terminal of our triac driver through a 360 ohm resistor. We choose 360 ohms here because our MOC can handle a maximum of 1A of current and we needed 35mA so 120/360 ~ 333mA which will undoubtedly turn on the triac and not damage the MOC. Furthermore, considering this current is used to enable the triac, it will only pass through for the amount of time necessary to turn the triac on, all current from that source then travels through the triac and, from there, to the load. We additionally use a 330 ohm resistor from MOC pin04 to the triac A1 to drop excess current to the load upon the switching off of the triac.

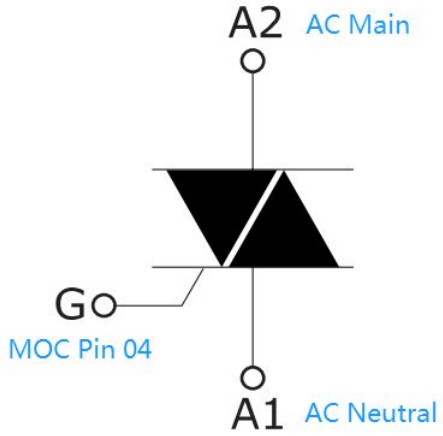


Figure 4: Triac Schematic and Symbol

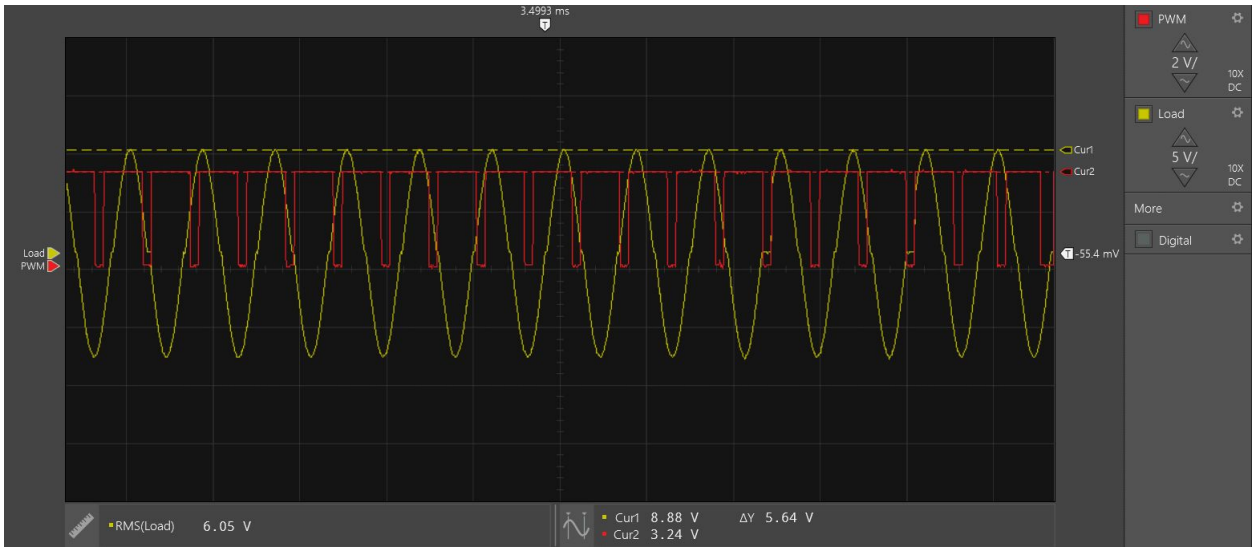


Figure 5: No Zero Crossing Monitoring

To limit the switching on to only happen at zero crossings, we implemented a phototransistor optocoupler, the FOD814 (FOD) to monitor said zero crossings. Pin03 of the FOD is connected to GND while pin 04 is connected to p15 on the launchpad. Pin 01 is connected to the AC Main while pin 02 is connected to the AC Neutral. We chose a bi-directional phototransistor optocoupler as we did not want the high voltages introduced on the AC line to reverse bias our optocoupler and cause it to behave abnormally. As a result, our design relies on the connection from pin03 to pin04 being made at all times except for when there is no voltage across pins 01 and 02. This means we get a peak whenever a zero-crossing occurs. We chose to add a pullup resistor to P15 as the internal pullup resistor of the CC was not very strong so our voltage peaks were not consistently high enough to trigger the interrupt that would sync our zero crossings with those of the wall. We eventually chose to use Timer0 on the CC to call a callback function every second that enables this zero-crossing interrupt, which

aligns the PWM and then disables itself. This way, the resetting of the PWM timer does not happen an unnecessary 60 times per second, but instead only once per second.

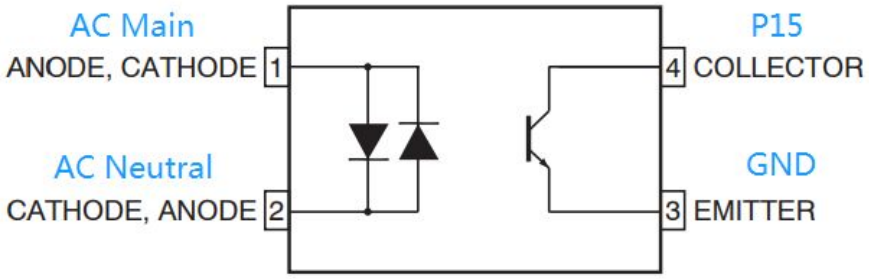


Figure 6: FOD 814 Phototransistor Optocoupler Schematic

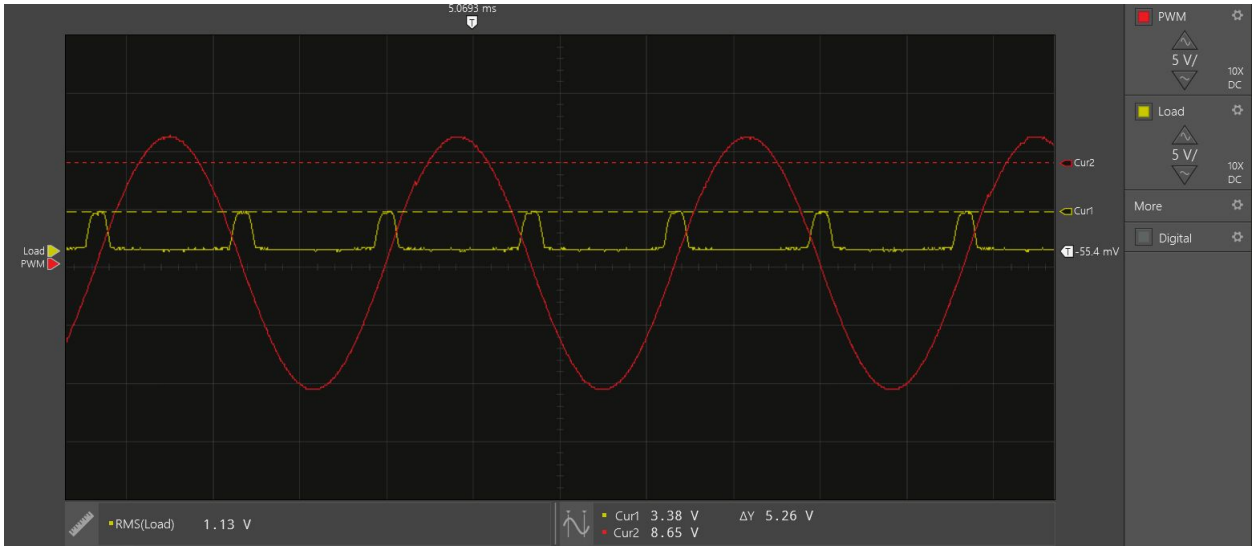


Figure 7: Zero Crossing

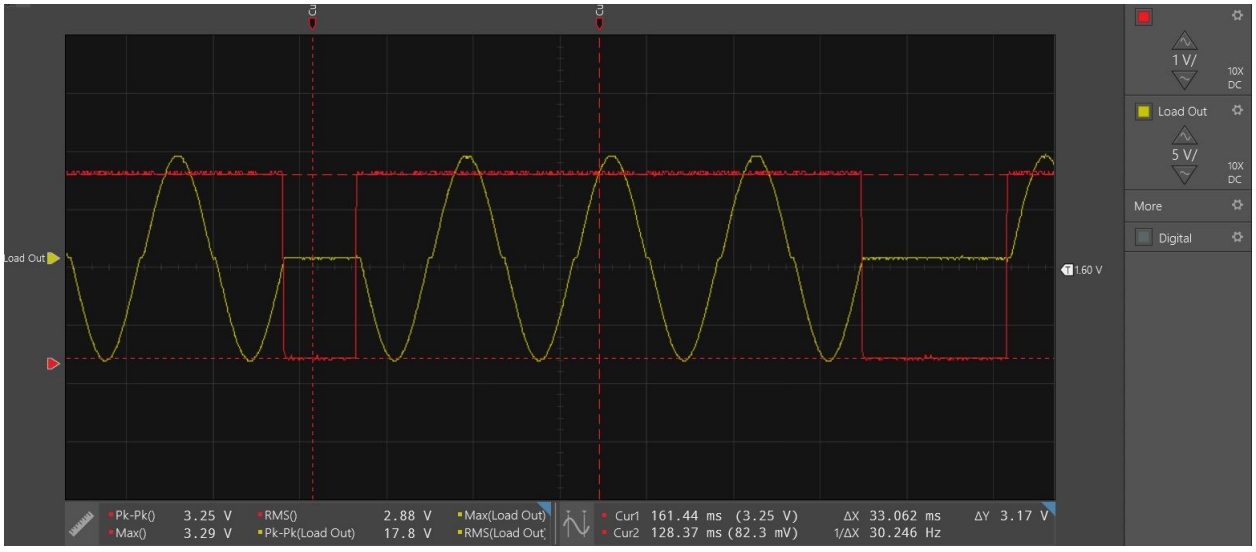


Figure 8: Monitoring with Zero Crossing

Having covered the high power functionality of our board, we move onto the user interface elements. We first set up our rotary encoder to be able to interface with the setpoint that for the temperature as we saw this as of higher importance than visualizing the setpoint. To do this, we used a Bourns PEC12R rotary encoder as they had a simple 5-pin layout with a good datasheet and they had built in pushbuttons. We ordered two of them so that we could see if one felt better than the other or in case one broke, but it turned out that we ended up using the one with a longer shaft as it made it easier to use through our plastic enclosure.

The rotary encoder that we chose was fairly standard, the only difference between this one and the one that we set up in our intro to embedded class was that we implemented hardware debouncing on the terminals to increase the software efficiency. This was done with two simple RC lowpass filters using a 10kOhm resistor and a 0.01uF capacitor. This gives us a breakpoint of around 1.6kHz as per the RC cut-off frequency calculation: fc = 1/(2pi*RC) = 1/(2pi(10000*0.01*10^-6)) = 1591.549Hz which was what the data sheet suggested and worked for us. We are running the rotary encoder at a different voltage than is expected, 3.3v instead of 5v. As a result, our interrupt trigger voltage is going to be lower than for a 5v supply, as the CC minimum high-level input voltage is 0.65*VDD; our bounces need to be lower than 2.145v instead of lower than 3.25v. However, since the bounce amplitude is directly dependent on the VDD, both of these voltages correspond to a decibel value of about -3.75dB. Looking back at the data sheet, the contact bounce is at least 500Hz so the 1600Hz breakpoint is not way off, however, we do not reach the -3.75dB value at 500Hz, we actually reach it closer to 2kHz. Using 40kOhm resistors instead would put us right at our goal value even at the lowest frequency bounces, this gives us a breakpoint of about 400Hz: fc = 1/(2pi*RC) = 1/(2pi(40000*0.01*10^-6)) = 397.887Hz. However, this ignores the possibility of overdamping the rotary encoder and causing issues recording all of the ticks. This should not be an issue with our rotary encoder as the maximum rated RPM is 100, meaning our maximum rotations per second is 1.6666. Considering we have a 24-phase encoder with 4 ticks per phase, this means the maximum number of ticks per second is 1.6666*24*4 = 160 which is only attenuated about 1.5dB with our 40kOhm design.

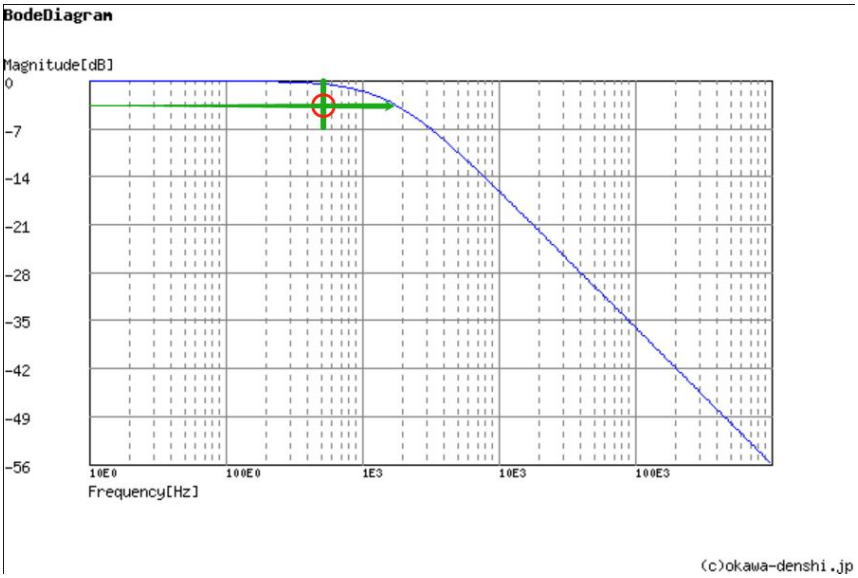


Figure 9: Frequency Response of the Rotary Encoder Debouncer Using 10k Ω

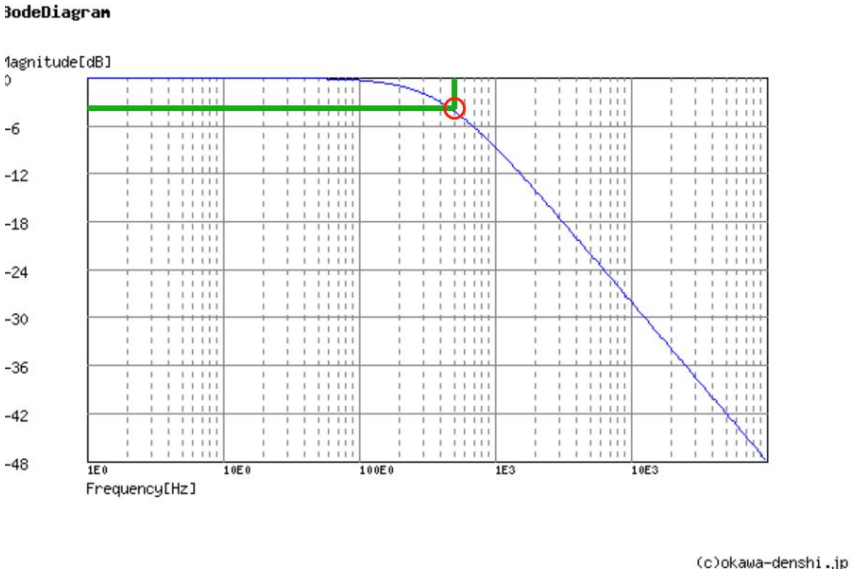


Figure 10: Frequency Response of the Rotary Encoder Debouncer Using 40k Ω

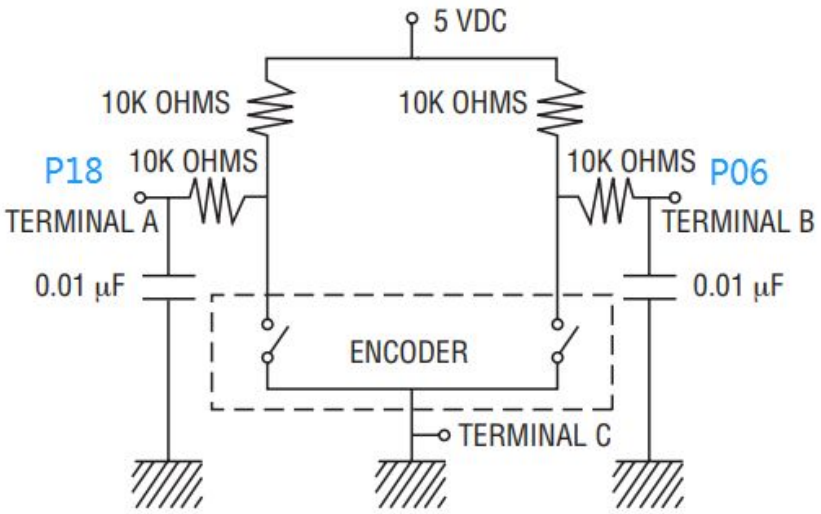


Figure 11: Rotary Encoder Schematic

The software implemented for our rotary encoder was rather simple, using binary to keep track of the old state and new state and then using this value to index into a lookup table of possible results. The result, being 0, 1, or -1 would then be added to a count variable that would increment the set temperature upon reaching 4 or decrement the set temperature upon reaching -4 and set itself back to 0 in either case.

The rotary encoder software was essentially a state machine. Our encoder is capable of 4 state transitions per tick, so there are 16 possible ways the state of the encoder could go. So we implemented a Mealy state machine to take care of all possible transitions.

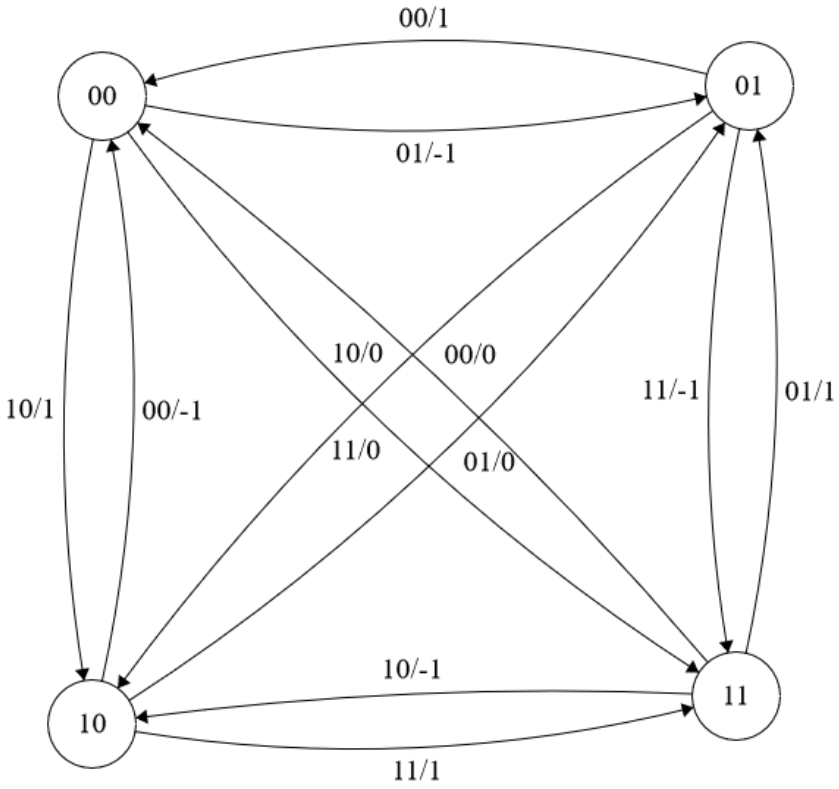


Figure 12: Rotary Encoder State Machine

The states are 2 bit binary numbers representing the current value of the encoder switches A and B as a binary number AB. The transitions are set up in standard Mealy format where the input is the new value of the state AB, and the output is what that transition is interpreted to be as an offset to some global counter we store. For instance, a transition from 00 to 01 would imply that the encoder B switch activated first. Going off the circuit diagram, we can see that this is a clockwise motion. So we increment the counter. Going the other way implies the opposite. We can see that completing 4 transitions in a certain direction would exhaust all possible states a tick in the direction would take, and would result in the counter either taking on a -4 or a +4. At this point, we have also come full circle so we must reset the counter to zero. All transitions with an output of 0 are considered to be bad transitions because more than one switch changed at a time, which cannot happen.

This basic idea led to an extremely efficient interrupt-driven state machine that encoded all of these transitions in binary. We chose to use a 4 bit binary number that we interpreted as a size 2 queue, where the most significant 2 bits are the old binary state of the encoder, and the least significant 2 bits are the current binary state of the encoder. Essentially what we did is assign an interrupt that both encoder switches would share. Upon trapping into the interrupt, we read the pin values at that instant in time. We then enqueue this in the state by left shifting the old state over and masking the least significant 2 bits with this new pin value. This resulting binary number was then matched up to the output function by way of a lookup table with each

index being one possible value of the 4 bit queue. The value stored at an index therefore is the output function value as shown on the state diagram. Using this output, we adjusted our global counter every time the interrupt fired. We finally checked for a complete transition by checking the value of the counter against -4 / +4 and either incremented or decremented the set point. We then had all of the tools we needed to display and adjust data on the LCD screen.

We chose the Newhaven C0220BiZ LCD screen (LCD) because it supported I2C and unsigned char string conversion over a small 8-pin bus, greatly simplifying the work we would need to do to set up our own code for sending the proper data to the proper location on an LCD. Furthermore, the screen offers 40 character spaces, allowing us a relatively large amount of freedom when it came to what data we would display.

For the physical setup, we followed the guidelines provided in the LCD datasheet by connecting a 1uF electrolytic capacitor across pins 7 and 8, a 0.47uF capacitor across pins 6 and 4, a 10kOhm resistor across pins 5 and 2 and a 10kOhm resistor across pins 5 and 3. The hardware setup for this part was made extremely difficult due to the fact that the pin layout on page 3 of the datasheet is completely mirrored on page 4. As a result, all 8 of our traces were incorrect and had to be redone using jumper wires. Furthermore, after testing the LCD with pins 03 and 04 for SCL and SDA, we realized that they did not function properly and had to switch to Using pins 01 and 02. This resulted in even more jumper wires and cut traces as the pull up resistors for P03 and P04 were now supposed to be connected to P01 and P02 and we had originally used P01 for our PWM controlling the MOSFET. This is a lesson in paying very close attention to the data sheet as to not get yourself into a bind where hours of time are wasted just to get back to where something can even be tested properly.

When implementing the I2C functionality for the LCD, we came across a few issues. The main one is described above, as our pins were incorrect, however, we originally also used a ceramic capacitor instead of an electrolytic for the capacitor across pins 07 and 08. This made the logic analyzer display the correct information, but left the screen completely blank.

We used the code examples provided on the LCD datasheet as a basis for implementing our own functions and, in utilizing the hardware I2C functionality of the CC, avoided unnecessary lines of code. The basis of our display system was to first send a series of commands upon booting up that set up the proper registers within the RAM of the device, this was described in the void init_LCD() function provided on the datasheet. Each time we would display to the screen, we would first clear the display using the control byte 0x00 and then the clear command byte 0x01. Next we would use our setCursor function to set the cursor to the 0th location on the 0th line, this meant calling the set DDRAM address command. Finally, we send the control byte and then the RAM write command, followed by a series of ASCII-encoded unsigned chars. The controller for the LCD screen is able to understand this data and moves the cursor along accordingly as the data is received. This function and some of our defines for the memory locations use a library for this specific LCD screen, however, we had to modify all of these functions to work with the CC and specifically with its hardware I2C functionality as the library was designed for an Arduino.

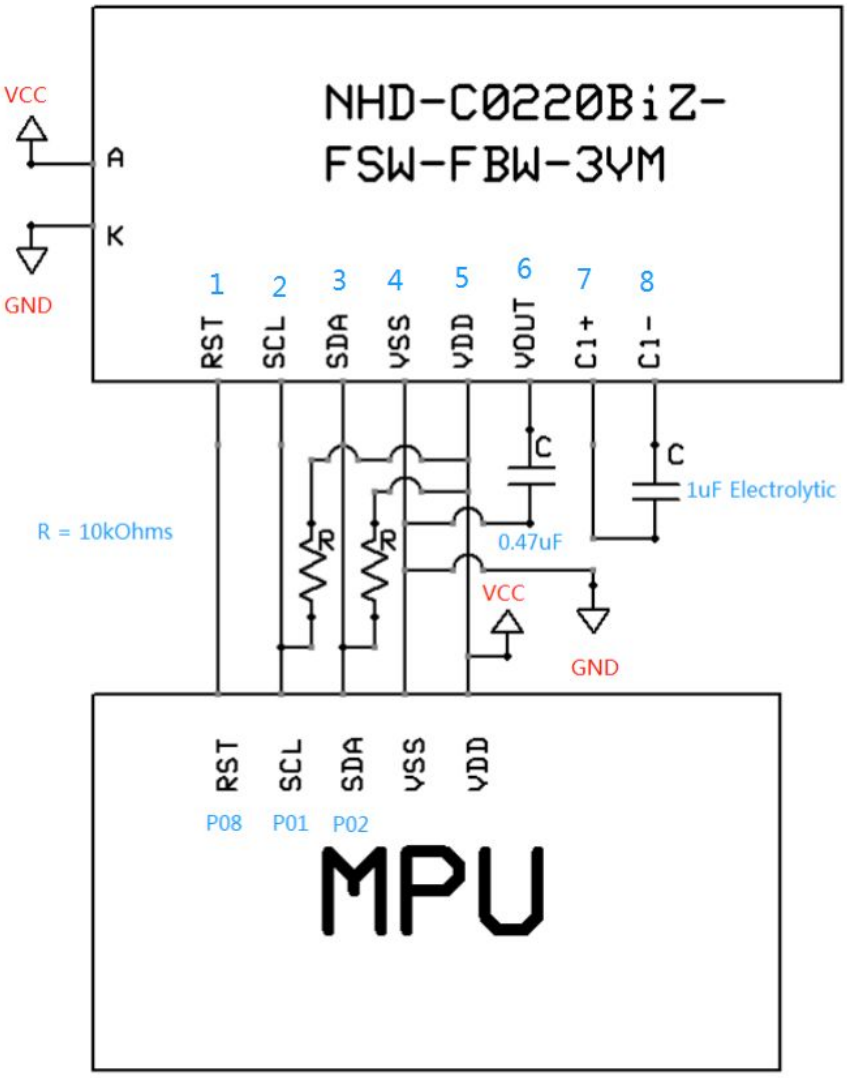


Figure 13: LCD Screen Suggested Implementation

Block Diagrams

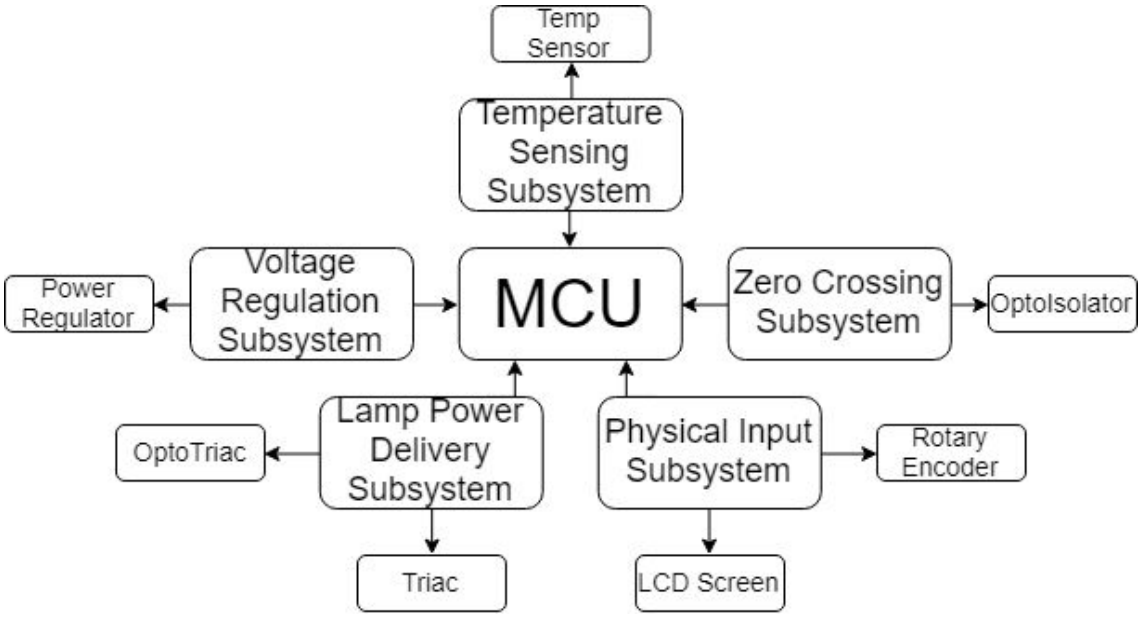


Figure 14: Overall Block Diagram

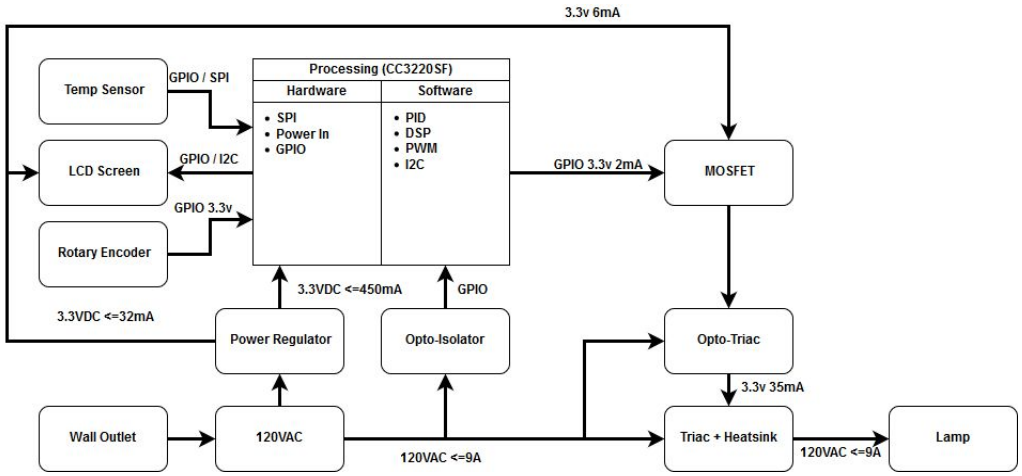


Figure 15: Comprehensive Block Diagram

Schematics

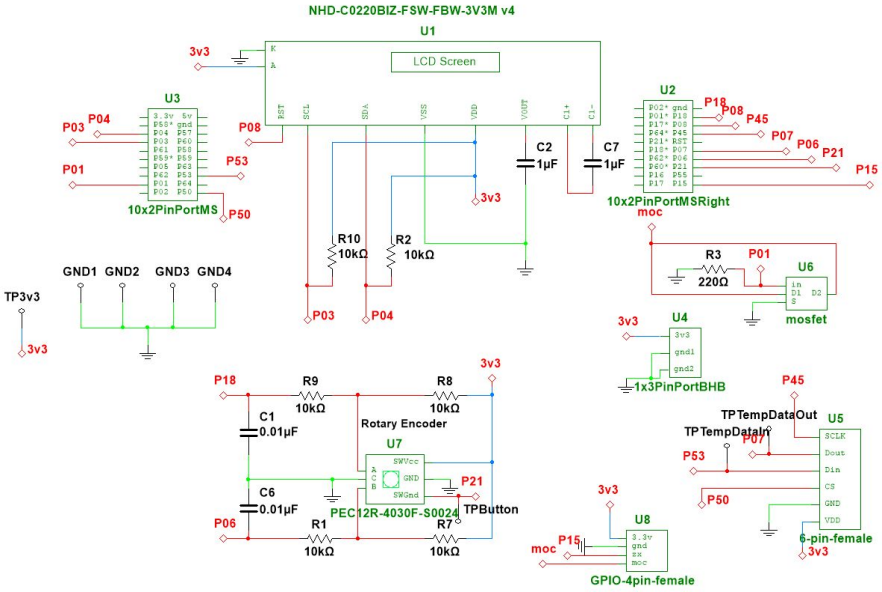


Figure 16: Low Power Board Multisim Schematic

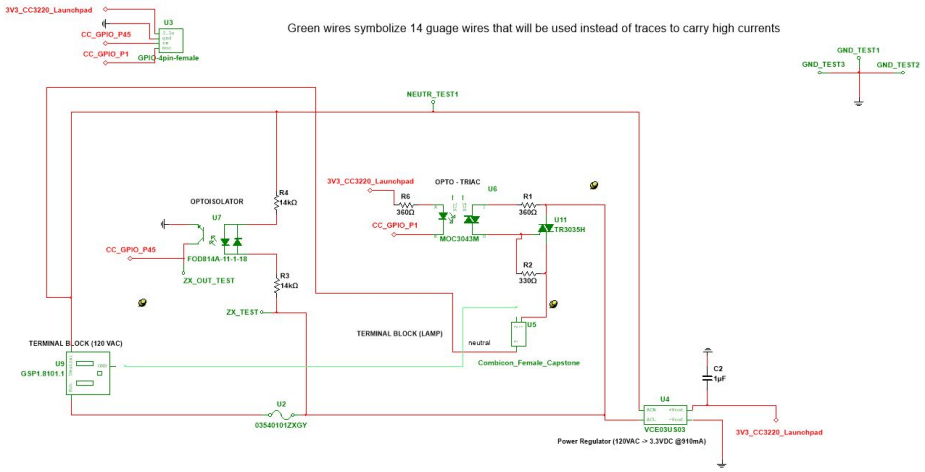


Figure 17: High Power Board Multisim Schematic

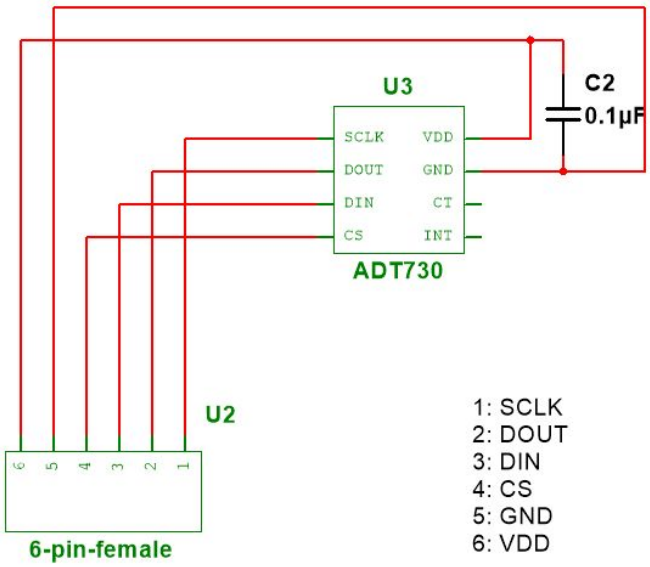


Figure 18: Temperature Sensor Multisim Schematic

Board Layouts

Our understandings of our parts evolved over the course of the projects and with them, our boards.

Lower Power Board

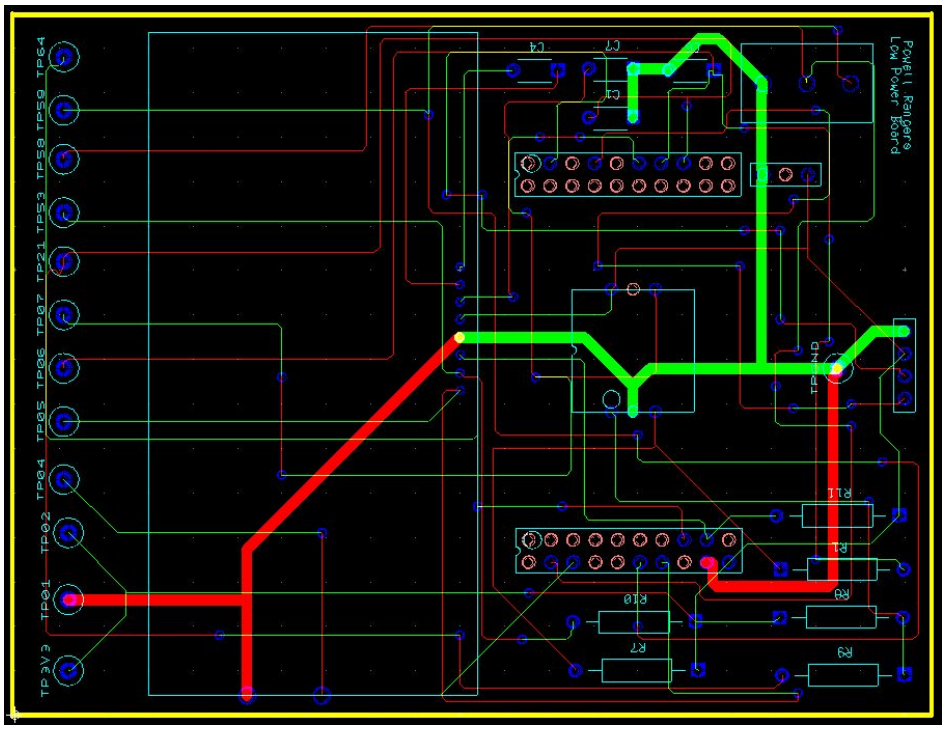


Figure 19: Low Power Board Ultiboard Schematic and Trace Version 1

At the time that we created our first LPB, we did not know what kind of connector we would be using so we assumed that we would just start off with jumper cables. The change here can be seen by the connector on the right changing from a simple opening in the board for a pin socket to a surface-mounted cable dock. At this time, we thought that we would be using a dallas 1-wire sensor, indicated by the 3-pin connector at the top of the board in both versions 1 and 2. The resistor layout and general locations of the rotary encoder and LCD screen stayed the same for the first 2 boards, with the only major difference being that the test pins were integrated into the board instead of being all at the top in the 2nd version. Lastly, our 3x1 connector for the VCC and GND of the Launchpad was shifted over to lineup with the Launchpad itself and the LCD screen footprint was modified to make it fit.

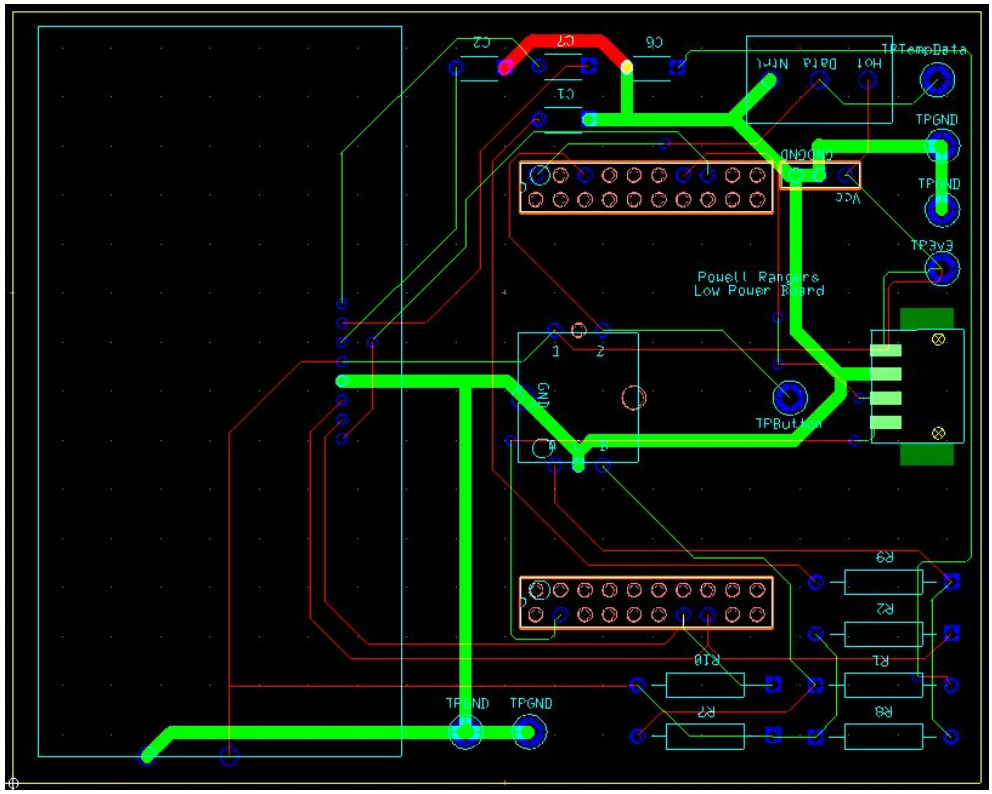


Figure 20: Low Power Board Version 2

The changes between the 2nd and 3rd LPBs are less drastic than between the first and second. The largest visible difference is the addition of a 6-pin surface-mounted cable dock in place of the 3-pin connector at the top right of the board. This was because we switched to an SPI temperature sensor and designed our own board for that in the 3rd iteration.The other final addition to the 3rd LPB was the MOSFET. We decided to put this on the LPB instead of the HPB as there was no ideal place for it and we had already finished the plans for the HPB. The addition was, as mentioned before, to be sure that there was enough current to reliably switch the MOC optocoupler.

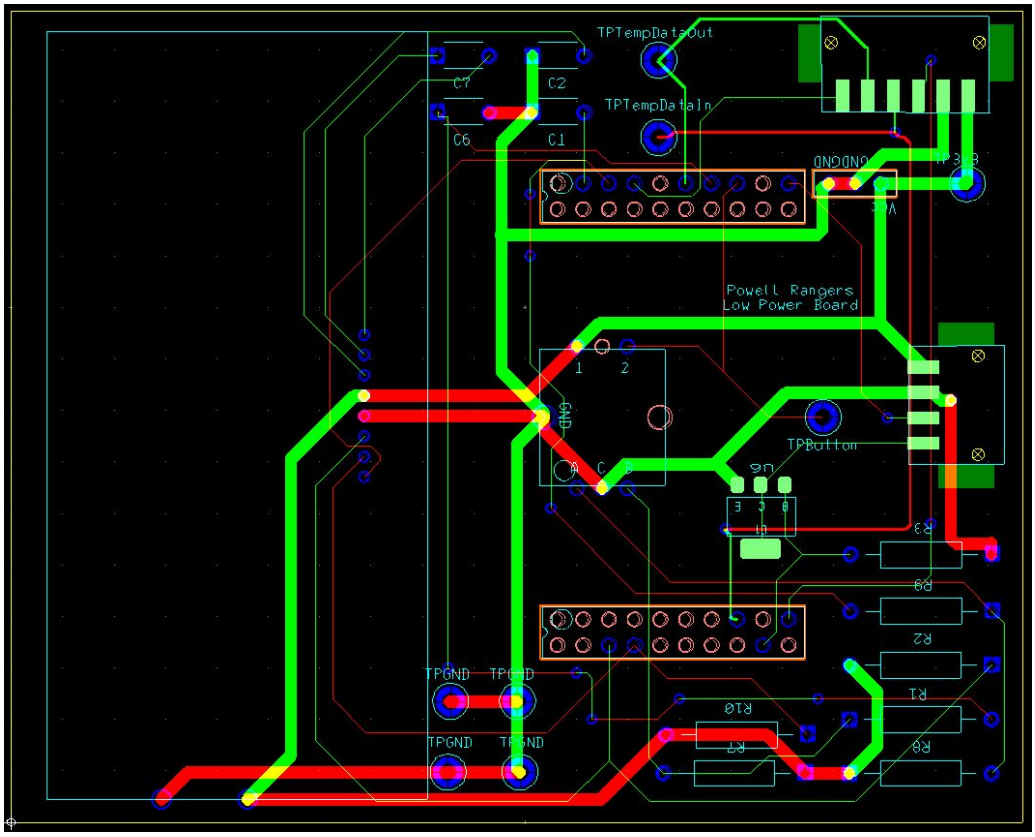


Figure 21: Low Power Board Version 3

High Power Board

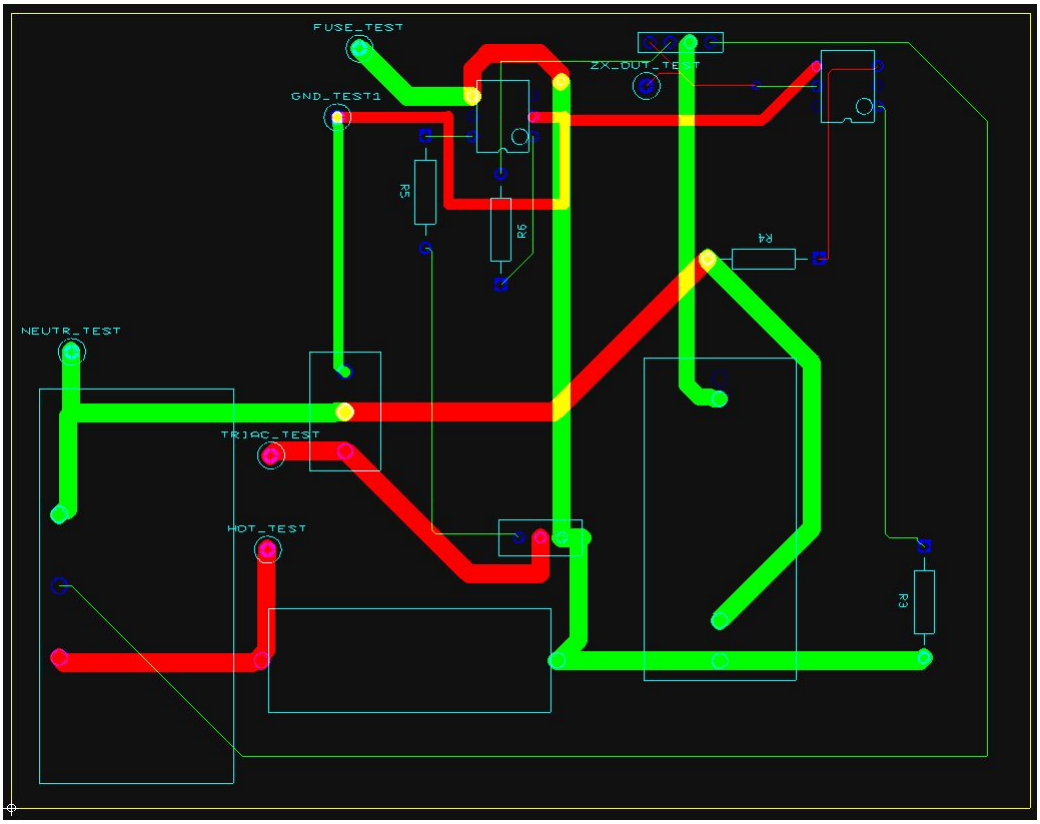


Figure 22: High Power Board Ultiboard Schematic and Trace Version 1

The differences between our first and second HPBs are probably the largest between any of the boards here. We went into the first HPB assuming that we would be able to make traces large enough to handle the current that we wanted to supply, up to 10A. It turns out that handling traces in the default fashion would make them far too close to each other, resulting in arcing between the pins of the triac. As a result, we made our first board with fairly small traces, just big enough to support 1A of current, as a test of our board at low current. Other major changes included not connecting the chassis GND through to the low power board and instead connecting it to the chassis GND of the load, specifically laying out the parts so that there were low and high voltage zones, and using low AWG wires to carry the current instead of traces. Our original idea was that we would use a 3-wire screwable combicon that was soldered into the board to connect to the load was changed to using a combicon that would solder in only for mechanical support and have 3 wire openings on two sides to connect a wire from each of hot, chassis gnd, and neutral to the corresponding wires of the load. This was overly complicated and changed in the 3rd design. Finally, we changed our triac footprint to allow the pins to be further apart, reducing the chance of arcing.

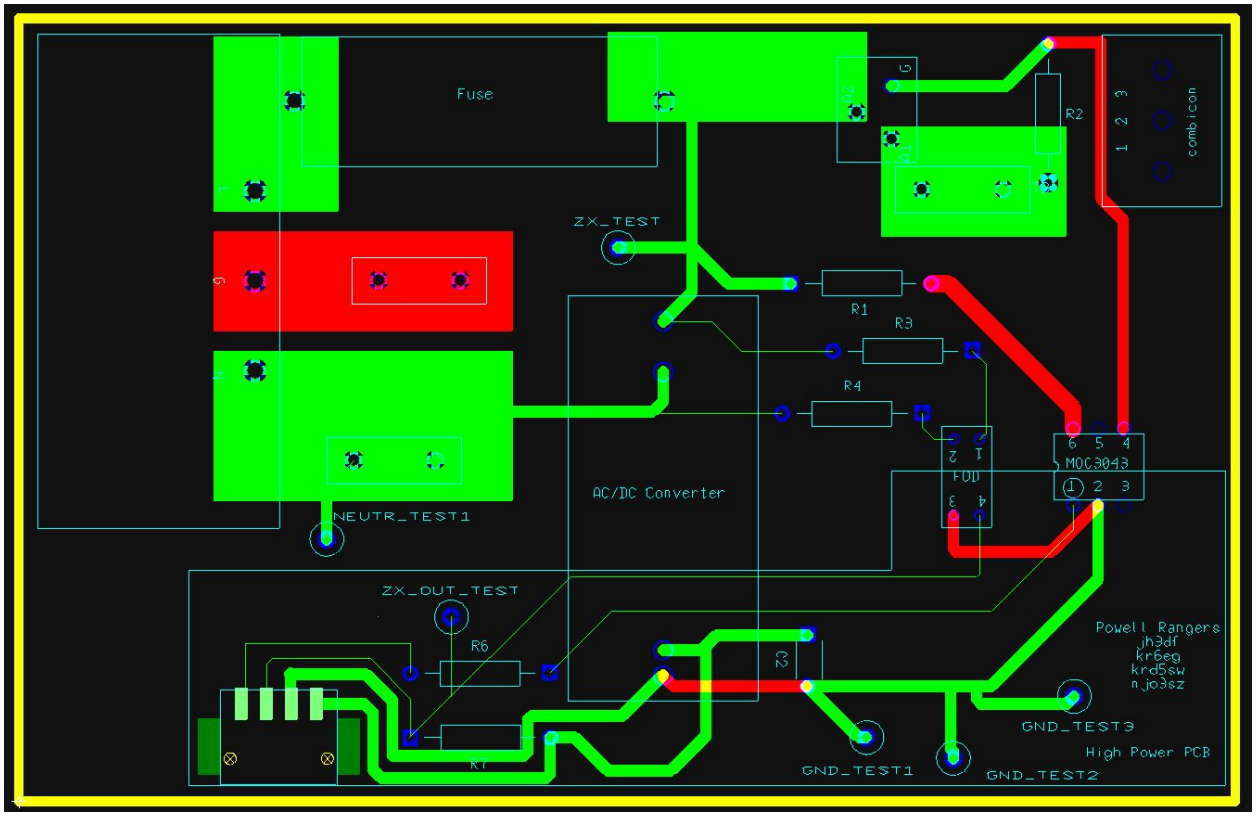


Figure 23: High Power Board Version 2

Foregoing the idea of using wires to connect hot, chassis gnd, and neutral, we ultimately decided that we could do it all just with traces, very large trace polygons. This required us to shift a lot of our parts around to keep valid separation between the traces with 120VAC on them. We also changed our pads to be much thicker in this design and changed our combicon to be a simple through-hole to male pin 3-pin combicon. This way, the wires from the load could be screwed into a female combicon that could be inserted or removed from the board as necessary. We also learned that the chassis gnd could be connected via a spade connector, so we were able to remove the entire trace for the chassis gnd from the board. While we specifically made room for the triac heatsink on the second board, we had a trace immediately below it on the copper top that would be carrying 120VAC, this was very dangerous. In our 3rd revision, we moved this trace to copper bottom to remove this risk. While the 2nd board was a complete overhaul of the 1st, the 3rd board was a drastic simplification of the 2nd. Ultimately, the use of trace polygons and the increased size of the pads along with manual checking of trace widths and separations left us with a much safer and cleaner board.

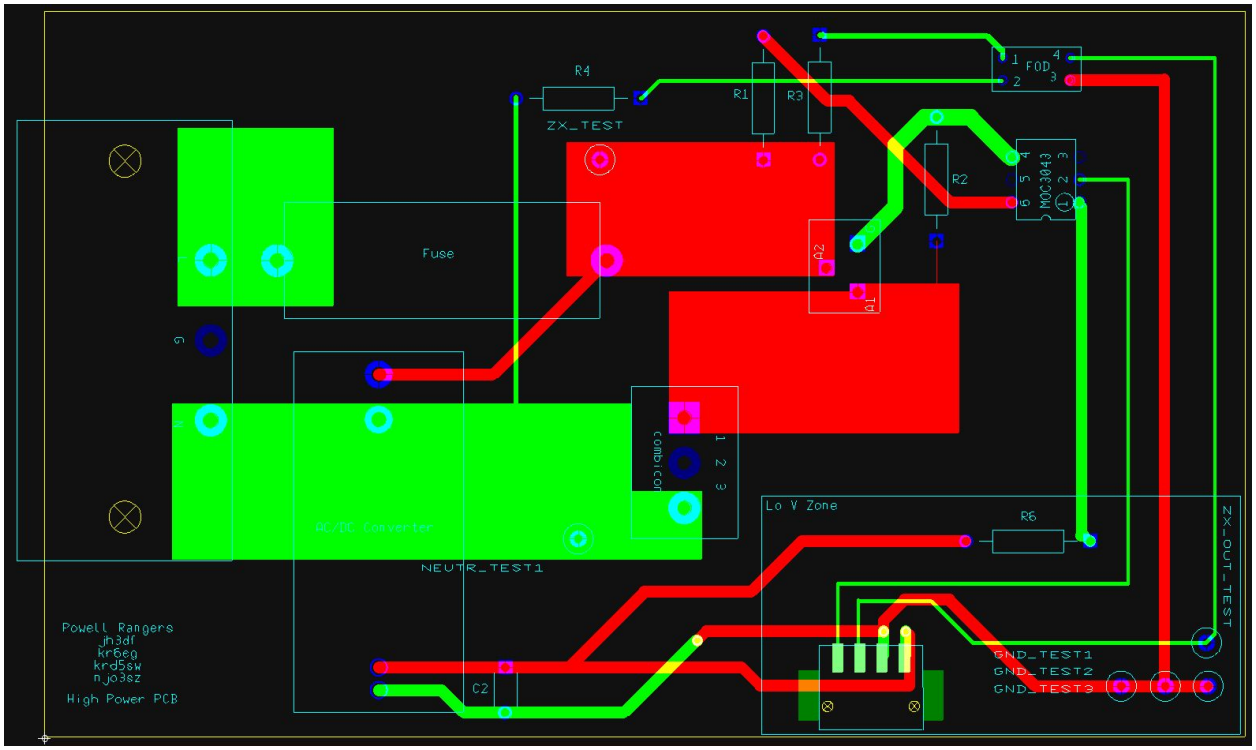


Figure 24: High Power Board Version 3

Temperature Sensor Board

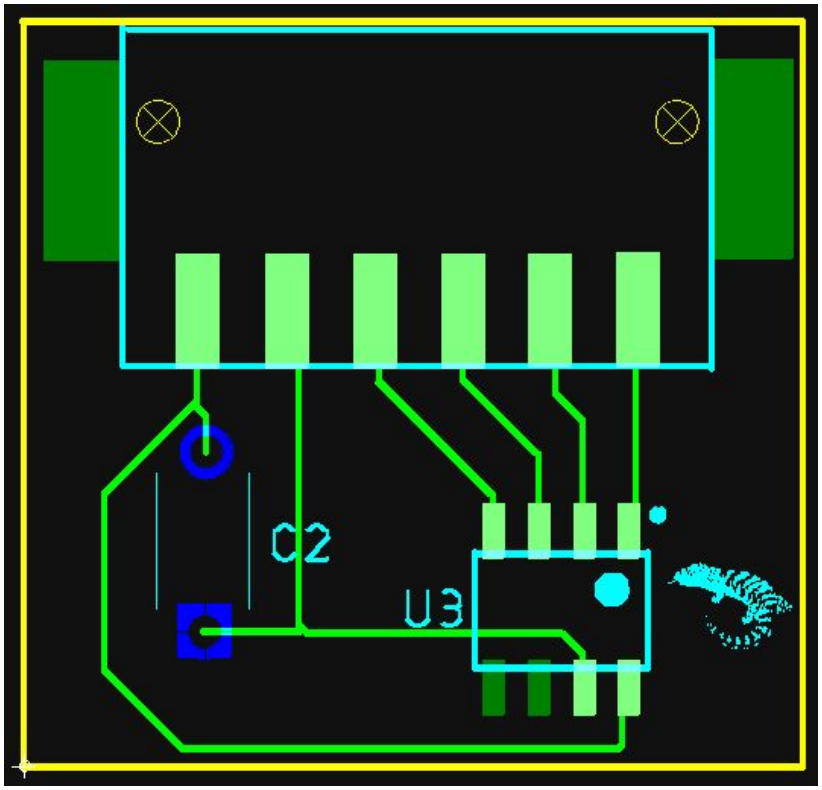


Figure 25: Temperature Sensor Board First and Final Version

We had only a single TB as it was decided on after the 2nd board submission. This board was less difficult to design than it was to physically solder. It turns out that we had moved the traces for the TMP sensor too close to the center of the footprint so, while it was possible to make a connection, the metal pad on the board was immediately below the pin, making it difficult to get the solder to adhere. We were able to add a little embellishment of a skink on this board just for fun, most people would never see it but we would know it was there.

Project Time Line

Proposal Gantt Chart

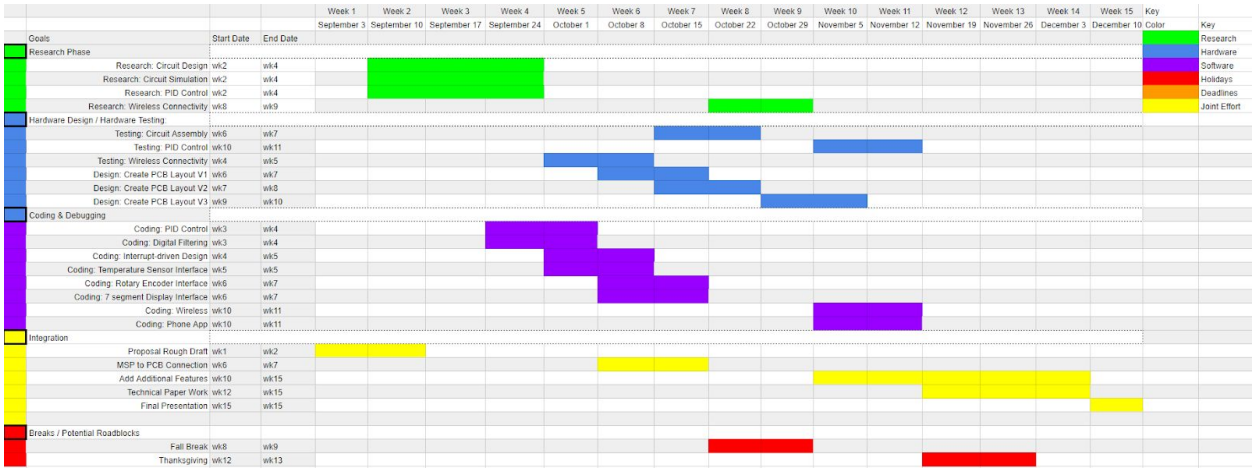


Figure 26: Initial Proposal Gantt Chart

Final Gantt Chart

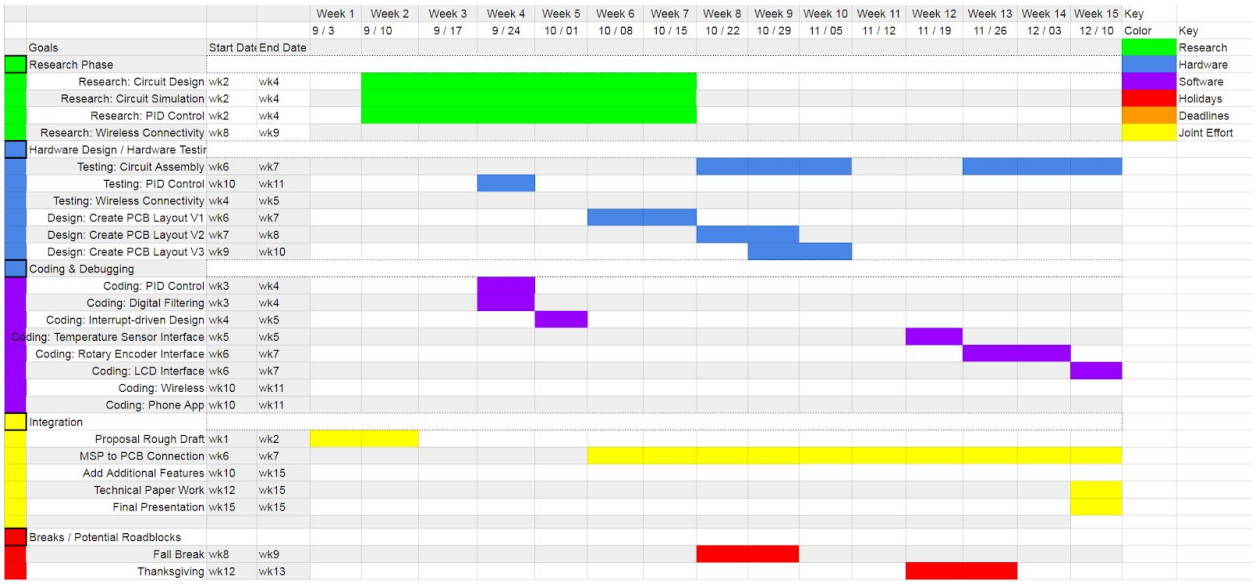


Figure 27: Final Gantt Chart

Our timelines changed dramatically from what we proposed originally because almost all of the software design, save for the PID and fixed point math, was dependent on the hardware setup built around the microcontroller. We needed to know what temperature sensor to interface to, what switching platform we were going to rely on, and a number of other similar dependencies. Another thing is that we spent a lot of time trying to get the boards right; we finally realized the 1000W platform that we presented in the third board submission.

We had the code ready for whatever was not changing in the setup, but that alone was not sufficient for testing the whole project. As a result, just about everything was serialized aside from the hardware design and testing, where we all pitched in to pick out parts, figure out dimensions, and generate layouts. The Android development could have admittedly been started earlier, but neither Jimmy nor Keerthi were very familiar with any of it and spent a lot of time just trying to grasp the concepts. This was their first experience with writing an app of this scale and integrating it with computer networking concepts they had never seen before.

Roles

Keerthi Radhakrishnan:

- Primary: Programming CC and Android Developer
- Secondary: PCB Design/Parts Research

Jimmy Howerton:

- Primary: Programming CC and Android Developer
- Secondary PCB Design/Parts Research

Kevin Dela Merced:

- Primary: PCB Design
- Secondary: Parts Research/Programming CC

Nathan Olszewski:

- Primary: Parts Research & Standards Tracking
- Secondary: PCB Design

Test Plan

<u>Deprecated Test Plan:</u>

1. Make sure the header-board fits on the launchpad unit
    a. Solder on the appropriate pin connectors and slot the board onto the launchpad
    b. Remove the board from the launchpad
2. Test for shorts and connections using a multimeter
    a. This first involves installing all of our test pins
    b. Use the virtual bench to apply 5v to HOT_TEST and check all other test points, all should show 0 voltage.
        i. Repeat this for test points SMPS_OUT_TEST, CC_VDD_TEST, TEMP_PWR_TEST, TEMP_OUT_TEST, GND_TEST1, OPTOTRIAC_CTRL_TEST
    c. Use the virtual bench to apply 5v to FUSE_TEST, make sure all other pins except for ZX_TEST show no voltage and that ZX_TEST shows 5v.
3. Install voltage regulation units
    a. We have chosen a 120VAC - 3.3VDC voltage regulator and it is vital to the MCU performance so we will test it first.
    b. Solder on the Female power socket to the board
    c. Connect the male power cable to the female socket

d. Plug the male power cable into the wall (CAREFUL)
e. Measure the voltage across the HOT_TEST and NEUTR_TEST pins and make sure it is reading 120VAC at 60Hz and that the solder on the board and the path connecting the outlet isn't melting.
    i. Unplug the male outlet wire from the wall
f. Solder in the power regulator
g. Individually test the inrush current limiter and the fuse
    i. Inrush current limiter
        1. Test the resistance of it to see if it is within 11.25 and 18.75 Ohms as per the data sheet's 15Ohms +-25% rating.
        2. Pass the highest amount of current that the virtual bench can supply through it for 5 minutes while continuing to measure the resistance
        3. Take measurements of the resistance once per minute and make sure that it approaches a resistance of 2-5Ohms. The data sheet rates its resistance lower at max current but the virtual bench cannot supply levels that high.
    ii. Fuse
        1. Use a multimeter to measure the resistance across the fuse to make sure it matches the 0.0077Ohm rating.
h. Solder in the inrush current limiter and the fuse
i. Plug the male wall outlet wire into a wall socket
j. Measure the voltage at the output of the power regulator using an oscilloscope to check that it is 5VDC
k. Measure the output voltage ripple to be sure that it is below 300mV that is acceptable by the MCU (it is rated for 120mV).
l. Unplug the male wall outlet wire from the female socket

4. Temperature Sensing Subsystem
    a. Solder on the TMP235 temperature sensor
    b. Use the virtual bench to apply a voltage of 3.3v to TEMP_OUT_TEST
    c. Measure the voltage at TEMP_PWR_TEST while changing the temperature of the sensor using a heating or cooling unit available in the classroom.
        i. This voltage should vary between 0.5V and 1V as the temperature is varied between 0C and 50C.
5. Zero-Crossing Subsystem
    a. Check the accuracy of the 15kOhm resistors using a multimeter to make sure that they are within 5% of the 15kOhms.
    b. Solder on the 15kOhm resistors.

c. Solder on the H11A1 Optoisolator and use the virtual bench to provide 20VAC 60Hz at ZX_TEST and place a 1x oscilloscope on that pin with its view shown on the virtual bench
d. Place 1V at ZX_OUT_TEST and place a 1x oscilloscope on that pin with its view shown on the virtual bench
e. Compare the output of the two oscilloscope readings to be sure that when the supply voltage at ZX_TEST goes below 0, the voltage reading a ZX_OUT_TEST goes to 0, and when the supply comes back above 0, the voltage at ZX_OUT_TEST returns to 1v producing a 1v square wave on the ZX_OUT_TEST oscilloscope.
f. Now remove the oscilloscope and voltage supply from ZX_TEST
g. Plug the male wall connector into the wall outlet
h. Check ZX_OUT_TEST to see if the same square wave is produced as before.
i. Unplug the male wall connector from the wall.

6. Lamp Power Delivery Subsystem
    a. Solder on the MOC3043M Optotriac
    b. Test the 360Ohm resistor to be sure that it is within its 5% rating.
    c. Solder on the 360 Ohm resistor
    d. Solder on the TR3035H triac
    e. Pass 3.3VDC at 6mA through the OPTOTRIAC_CTRL_TEST pin and place 5VAC at the FUSE_TEST pin. Place an oscilloscope at the FUSE_TEST pin.
    f. Check TRIAC_TEST pin using an oscilloscope to be sure that the readings match those of FUSE_TEST.
    g. Now turn the voltage at OPTOTRIAC_CTRL_TEST to 0v and check that the voltage at TRIAC_TEST is now 0v.
    h. If these do match, connect the heatsink to the triac
    i. Remove the 5VAC from the FUSE_TEST and the oscilloscope from FUSE_TEST.
    j. Plug the male power plug into the wall outlet
    k. Check TRIAC_TEST to make sure it is still at 0v
    l. Turn the voltage at OPTOTRIAC_CTRL_TEST back to 3.3VDC at 6ma
    m. Check the voltage at TRIAC_TEST to see it at 120VAC 60Hz.
    n. Unplug the male wall connector from the wall.
    o. Solder on the TERMINAL BLOCK (LAMP) part and connect the wires of the female wall connector to the terminal using screws.
7. Physical Input testing
    a. LCD Screen

i. Verify that the capacitance values for C1,C2 for the LCD screen are within 5% of the 1uF. Be absolutely sure they are within the 0.47uF to 2.2uF range.
ii. Supply the screen with 3.3VDC from the virtual bench and see that it lights up.

b. Rotary Encoder
i. Verify that the resistance values for all four resistors are within 5% of 10kOhms and that the capacitance values for both of the capacitors are within 5% of 0.01uF.
ii. Supply 3.3VDC to the input port and measure the output readings from both pins
1. Turn the encoder and see that the pins change from 0v to 3.3v
iii. Change the readings to come from the switch pins
1. Push the switch in and see that the voltage goes high

8. Software Testing
a. Now that the board has been assembled, connect it to the launchpad and carefully plug the male wall connector into the wall, if the LEDs turn on on the launchpad then the tester can begin software testing.

Final Test Plan:

This test plan dictates how the testing should have gone. For what actually happened, see the Test Results section in the Appendix.

Board Connectivity

LPB Fitting Launchpad:

- Connect the 20-pin and 3-pin connectors to the underside of the LPB and make sure that they fit onto the CC3220SF Launchpad that they will be seated onto.

Test Points:

- Connect all of the test points.

LPB-HPB 4-pin Cable:

a. Decide on a cable routing for the LPB-HPB cable and connect both cable connectors with this orientation
i. Black = MOC control output pin
ii. White = ZX interrupt input pin
iii. Red = GND
iv. Yellow = 3.3VCC

b. Once both cables are connected and the orientation has been verified to be correct for both boards (i.e. Vcc for LPB goes to Vcc for HPB), use jumper cables plugged into the connectors with a virtual bench supply to pass a voltage into each individual wire and check it on the other side.
    i. If the voltage is not detected, remove the cable, twist it into a more cohesive piece and reinsert it. If the cable is determined to be too short, remove all cables on that side, trim them down, strip them to be the same length, then reinsert all of them and redo this test.
c. Plug both ends of the cable into the respective boards, connect the LPB to the launchpad and manually change the GPIO values from high to low on the MOC and ZX pins.
    i. Check if the voltage on the HPB side is changing the same way and check that VCC is at 3.3v and GND is at 0v. If this is not the case, reseat the wires as mentioned in step *bi* and redo the test.

LPB-Temp Sensor 6-pin Cable:

d. Decide on a cable routing for the LPB-Temp sensor cable and connect both cable connectors with this orientation
    i. White = VCC
    ii. Green = GND
    iii. Yellow = Chip Select
    iv. Blue = LPB MISO, TB DOUT
    v. Black = LPB MOSI, TB DIN
    vi. Red = SCLK
e. Once both cables are connected and the orientation has been verified to be correct for both boards (i.e. Vcc for LPB goes to Vcc for HPB), use jumper cables plugged into the connectors with a virtual bench supply to pass a voltage into each individual wire and check it on the other side.
    i. If the voltage is not detected, remove the cable, twist it into a more cohesive piece and reinsert it. If the cable is determined to be too short, remove all cables on that side, trim them down, strip them to be the same length, then reinsert all of them and redo this test.
f. Plug both ends of the cable into the respective boards, connect the LPB to the launchpad and manually change the GPIO values from high to low on pins p07(MOSI), p53(MISO), p05(SCLK), and p50(Chip Select). Check if the voltage on the HPB side is changing the same way and check that VCC is at 3.3v and GND is at 0v. If this is not the case, reseat the wires as mentioned in step *bi* and redo the test.

Sensing Subsystem

SPI Temperature Sensor:

a. Because this is a surface mount part with 8 tiny pins and because it is on a board with only 1 other component and a connector, we chose to solder this part directly onto the PCB for testing.
b. Solder on the bypass capacitor
c. Solder on the cable connector
d. Solder the cable connector onto the LPB
e. Connect the LPB to the Launchpad and plug the cable into the LPB and into the temperature board (TB)
f. Toggle the GPIO pins on the launchpad and check the individual pins of the temperature sensor using the sharp oscilloscope to make sure the voltage is fully propagating.
    i. If this is not the case, use a microscope to check that the solder is making full contact between the board and the pins. When we first did this, our GND pin was disconnected.
g. Now that all pins are connected properly, enable the hardware SPI on the launchpad by setting the dataSize to 1 byte, the bitRate to 100kHz and the pole and phase to 0,0. Connect the logic analyzer to pin p07 for MOSI, pin p53 for MISO, pin p45 for SCLK and pin p50 for chip select. Now send blank packets to make sure the SCLK is ticking
    i. If it is not ticking and the config files do not allow options to make it tick, add a jumper wire from P45 to P05 to use the other SCLK pin and verify that this setup produces an SCLK output.
h. Now send data over the SPI line and analyze the automatic settings that the hardware SPI from the launchpad provides.
    i. If the chip select setting will not work for the temperature sensor, go to the CC3220SF_LAUNCHXL.c file and change the settings in the header to allow for software-defined control of the pins necessary.
    ii. If the temperature sensor needs to have data coming in to send data back, create a DUMMY variable that holds 0xFF and send that during the read period.
i. If the data is being sent but nothing is being returned, check the data sheet and re-verify that the cables are taking the right data to the right place.
    i. Make sure to check that you didn't confuse the MOSI and MISO as they are reversed for the launchpad and the temperature sensor. If these are switched, swap the wires pertaining to MISO and MOSI on one end of the cable.

j. If data is being received but it doesn't make any sense, make sure that you are doing the proper calculations provided on the data sheet, for 16 bit reads this means shifting the first bit received to the left 8 bits and ORing it with the second bit received, then shifting that to the right 7 times.
k. Upon receiving the correct temperature information, reconfigure the calculation to take advantage of a fixed point library, mentioned in the algorithm testing part of the test plan to achieve higher precision.

Power Delivery Subsystem Testing

Stage 1: Breadboard

MOSFET/MOC/Triac:

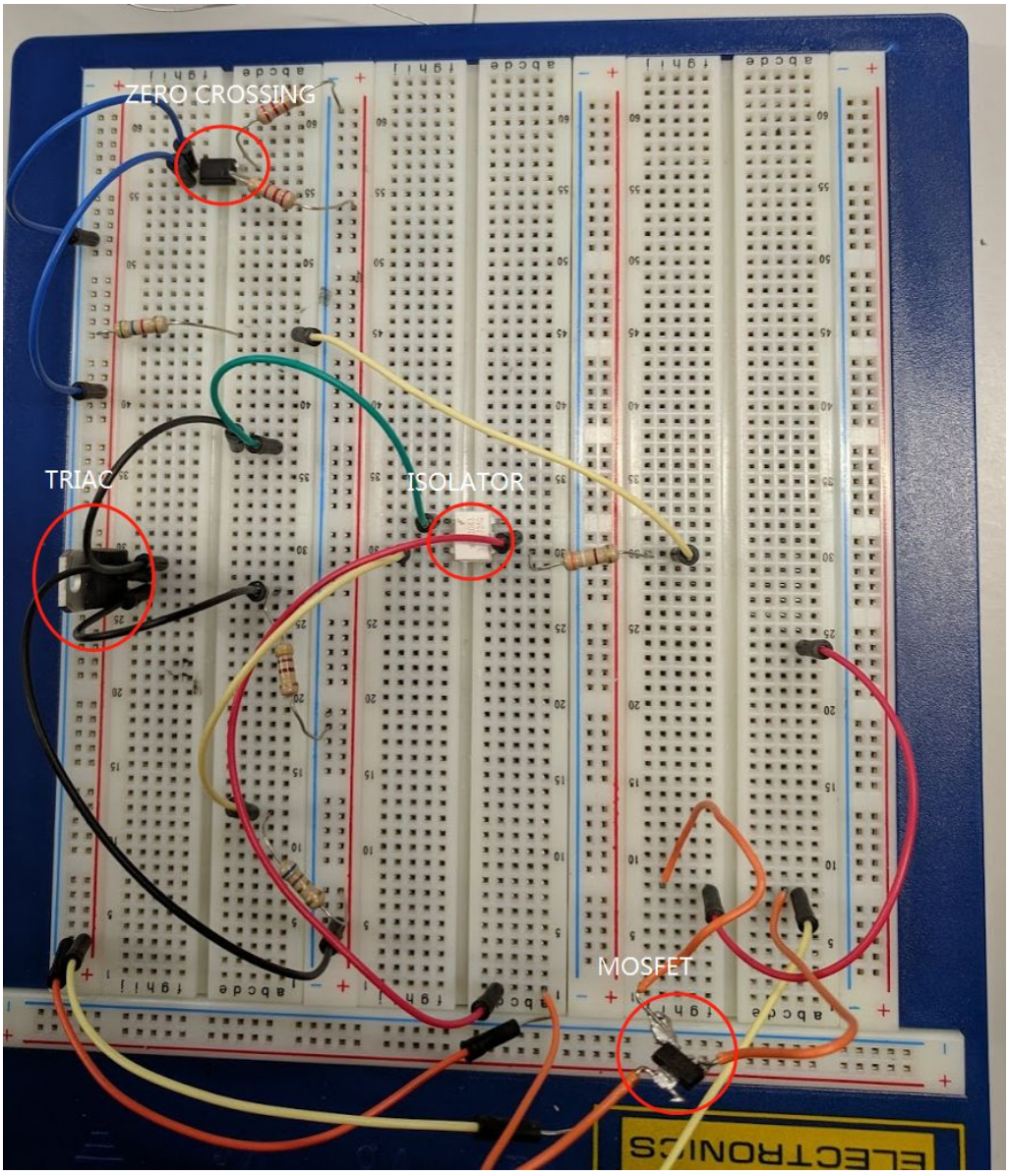
a. We tested these three components together as their functionality is dependent upon one another.
b. For the mosfet, as this is a surface-mount part, we must solder jumper wires to its pins in order to connect it to the breadboard
c. Connect the drain, source, and in pins of the mosfet to three separate breadboard openings
d. Connect the in port to P01 on the launchpad
e. Connect the drain to the cathode (pin02) of the MOC3043
f. Connect the Source to VDD at 3.3v
g. Use software to toggle P01 and measure the voltage at the gate cathode of the MOC
h. Connect the MOC anode (pin 01) through a 50 ohm resistor to VDD. This corresponds to R6 on the HPB.
    i. This resistor should be changed to a 360 ohm resistor when testing high voltage in order to decrease the current through the MOC.
i. Connect one of the main terminals(pin 04, pin 06) of the MOC to the Gate of the Triac
    i. Connect a 330 ohm resistor from the gate to the A1 section of the Triac. This corresponds to R2 on the HPB.
j. Connect the other main terminal(pin 04, pin 06) through a 360 ohm resistor to the A2 terminal of the Triac. This corresponds to R1 on the HPB.
k. Supply an AC voltage of 24VPP to the A2 terminal of the Triac and connect the corresponding ACGND to a different location of the breadboard, this simulates neutral.
l. Connect the A1 terminal of the Triac through a 470 ohm resistor to ACGND. This corresponds to the combicon on the PCB.

    i. The resistance of the lamp will be lower than this but this is purely for testing reasons.

m. Toggle pin 01 on the launchpad and measure the current through the drain of the mosfet to see if the 360 ohm MOC anode resistor is drawing the proper 6mA that it needs.
n. Check the voltage across the A1 and A2 pins of the triac and check the voltage of the GPIO pin and see that the AC voltage is being toggled via the GPIO toggle. The voltage at the triac should be high when the pin is low.
    i. If the voltage is not toggling, make sure that the triac is not setup backwards as terminals A1 and A2 are not reversible.

<u>FOD:</u>

a. Now that P01 is toggling the current through the triac, we connect the FOD to ensure that the switching lines up with zero crossings.
b. Connect a 220 ohm resistor from the 24VAC source to pin 01 of the FOD and connect the ACGND to pin 02 of the FOD. These correspond to resistors R3 and R4 on the HPB.
c. Connect pin 03 of the FOD to the GND of the launchpad and connect pin 04 to P15 (GPIO22) on the launchpad.
d. Set up an interrupt in the software to be called whenever this pin goes high.
e. Run the software and make sure that the interrupt is triggered when the zero crossing is connected and not triggered when the zero crossing is disconnected.
    i. If the interrupt is not being triggered here, include a pullup resistor from the pin to VCC. Using a resistor value like 5.6k ohms seen in the image below should make the peaks of the signal significantly high. There is no designated place for this resistor on the LPB or HPB so it will need to be soldered between two arbitrary locations on the board that connect these voltages. We suggest connecting it between the ZX and VCC pins on the HPB connector.



Stage 2: PCB

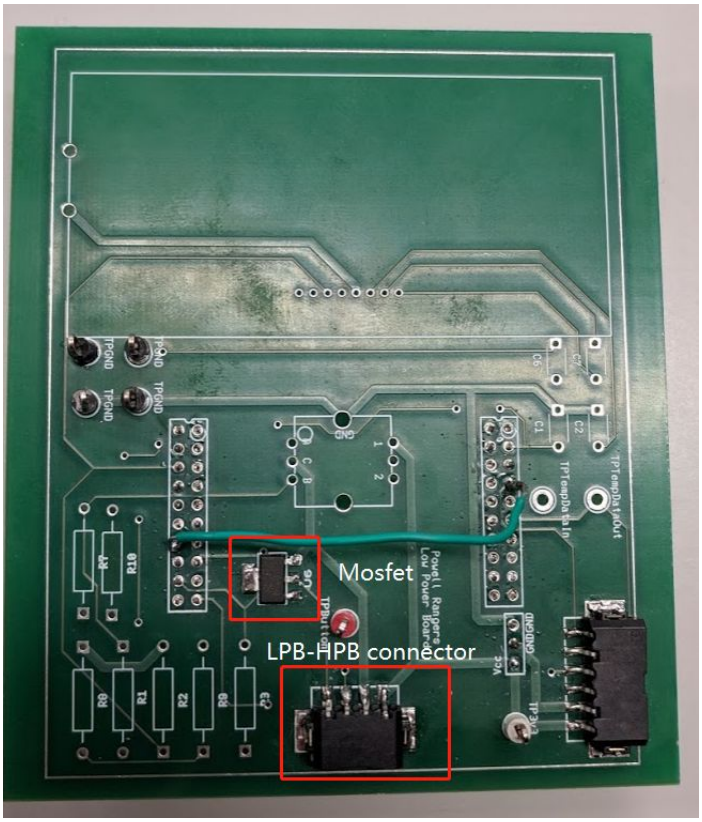
MOSFET/MOC/Triac:

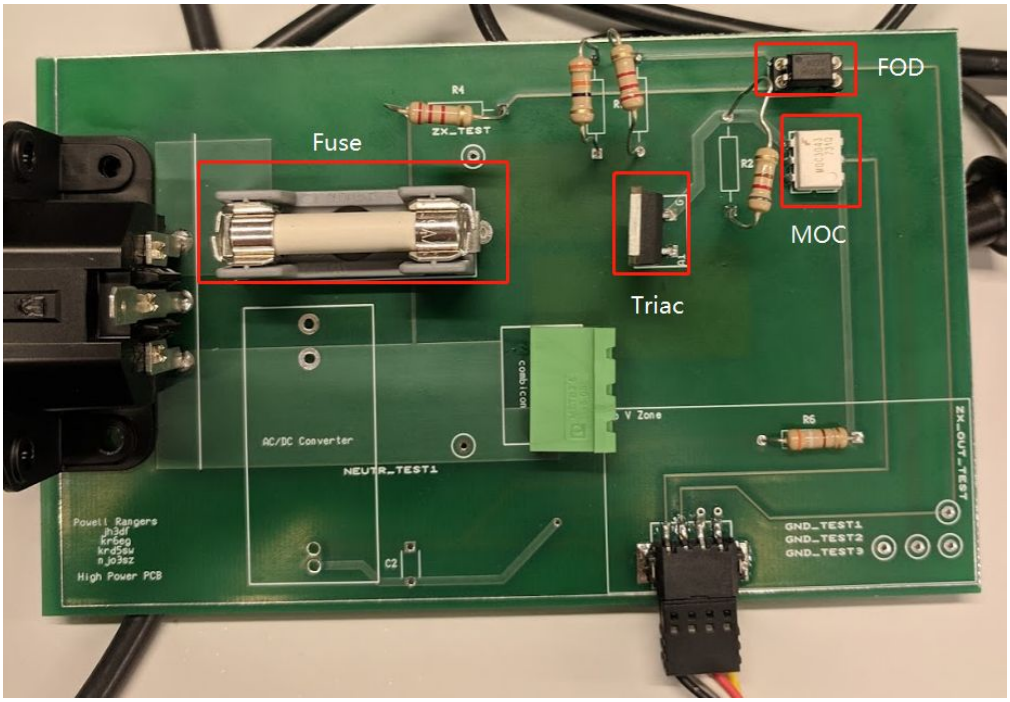
a. Connect the wall outlet connector and the fuse
b. Supply a 24vac voltage across the wall outlet connector and measure the voltage at the fuse to make sure it reaches the other end of the fuse unaffected.
c. Solder on the triac and check the voltage at pin A2 of the triac to be sure that the 24VAC reaches this pin unaffected.
d. Solder the MOSFET onto the LPB
e. Solder the 360 ohm resistor R6 onto the HPB

f. Solder the MOC onto the HPB
g. Now that the low voltage side of the MOC is connected, we can test if the main terminals of the MOC are connected or not by measuring the voltage across them.
    i. The datasheet for the MOC specifies that 3v should be across its terminals, this means we can toggle P01 on the launchpad and measure the voltage across these terminals to see if it goes to 0 when the P01 is high.
h. Solder on R2 using a 330 ohm resistor but don't solder it close to the board as this will be replaced with a higher voltage resistor in the next stage.
i. Measure the voltage across terminals A1 and A2 as P01 is toggled on the launchpad to see if there is an AC of 24VAC when P01 is low and an AC of 0v when P01 is high.
    i. If this is not the case, consider the possibility that the MOC could be broken or soldered in backwards.

<u>FOD:</u>

a. Solder in the chipseat for the FOD
b. Solder in resistors R4 and R3 at values of 220 ohms.
c. Run the software the same way as in part i but assess whether the zero crossing interrupt is being triggered or not.
    i. If not, check that the voltage at the pin 4 of the FOD and see what it looks like over the course of the AC wave. If the peaks are very shallow, the addition of a pull-up resistor could be useful as per step e of the FOD section of stage 1 of this section of the test plan.
d. Now check that the interrupt is actually resetting the PWM of P01 onto the correct part of the wave.
    i. If this is not occurring, perform software debugging to determine what the issue is.
        1. This will require determining if the PWM clock is being reset when the interrupt fires, note that turning the PWM off and then back on will not reset the clock and simply delays the PWM for the amount of time between the off and on.





Stage 3: PCB High Voltage

MOSFET/MOC/Triac:

a. Remove resistors R1 and R2.

b. Solder on high voltage/wattage equivalents in their places using a 360 ohm 0.5w 200v resistor for R1 and a 330 ohm 0.5w 200v resistor for R2.
c. Obtain a long power cable that can be plugged into the wall and into the power port of the HPB.
d. Plug the cable into the HPB and place the board into a safe area, preferably into an enclosure of some kind.
e. Be sure there are at least two people present and that a defibrillator is available in the same building when performing any tests using 120VAC.
f. Plug the board into the wall, wait 10 seconds and then unplug it.
g. Check the board for any visible signs of damage, and, upon finding none, measure the voltage across terminals A1 and A2 as P01 is toggled on the launchpad to see if there is an AC of 120VAC when P01 is low and an AC of 0v when P01 is high. The oscilloscopes for this part must be placed beforehand and not touched while the board is plugged in.
    i. If this is not working, be sure that there is no electrical connection between the pins of the combicon as this could cause a short and blow the fuse.
    ii. Likewise, make sure the oscilloscope probes are far enough apart when measuring this to not cause arcing.

FOD:

a. Remove resistors R3, and R4
b. Solder on 14k 0.5w 200v resistors for R3 and R4
c. Carefully plug the board into the wall with oscilloscopes measuring the voltage across pins 3 and 4 of the FOD.
d. Check that the interrupts are being triggered in the code at the right intervals. From the previous testing section you should know that the interrupts are happening at the right time.

Voltage Regulation

Voltage Regulator:

a. Now that the 120VAC input is safely propagating through the system, we can check that the voltage regulator works as expected.
b. Check the voltage at the ACL pin of the regulator opening on the HPB to make sure it is receiving 120VAC and check the ACN pin to make sure it is receiving 0VAC.
c. The board is currently being powered via the USB cable so check the +Vout pin of the voltage regulator pin on the HPB and make sure it is at 3.3v, check the -Vout pin and make sure it is at 0v.
d. Unplug the board

e. Solder the voltage regulator on, then attach an oscilloscope to measure across +Vout and -Vout
f. Plug the board into the wall
g. Check that the voltage on the oscilloscope is 3.3v.
    i. If this is not the case, check this with the board plugged in or not plugged into the LPB and launchpad.

Interface Testing

Stage 1: Breadboard

Rotary Encoder:

a. Connect the rotary encoder to the breadboard.
b. Measure 4 10kohm resistors and connect them as such:
    i. One across channel A to VDD
    ii. One across channel B to VDD
    iii. One from Channel A to an open area of the breadboard, call this Terminal A
    iv. One from Channel B to a separate open area of the breadboard, call this Terminal B
c. Connect 2 0.01uF capacitor as such:
    i. One from Terminal A to GND
    ii. One from Terminal B to GND
d. Connect channel C to GND
e. Connect Pin 1 to VDD
f. Connect a jumper from P18 to Terminal A and one from P06 to Terminal B.
g. Connect oscilloscopes to Terminal A and Terminal B
h. Turn the rotary encoder and be sure that you see a two square waves that are offset.
    i. If this is not the case, check that the resistors and capacitors are connected properly and that the oscilloscopes are connected in the right place.
i. Set up separate interrupts on the two pins and run the code, make sure that the both interrupts are hit.
j. Now consolidate the interrupts into a single interrupt that does calculations based on the old and new states and have them modify a global integer variable.
k. Turn the rotary encoder one direction and check the global variable, then turn it the other way, being sure that the values increase in one direction and decrease in the other.

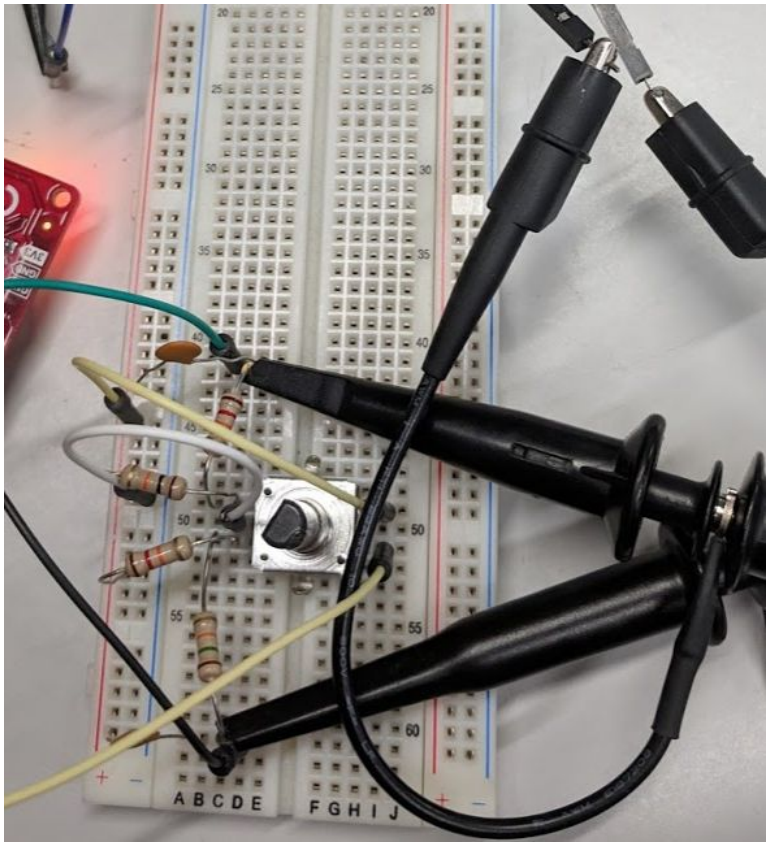


Figure 31: Rotary Encoder Testing

LCD Screen:

a. Solder jumper cables to the pins of the LCD after separating them enough as to not have them touch each other.
b. Plug the jumpers into a breadboard and use more jumper cables to transfer the layout to a more open part of the breadboard so the pins are visible.

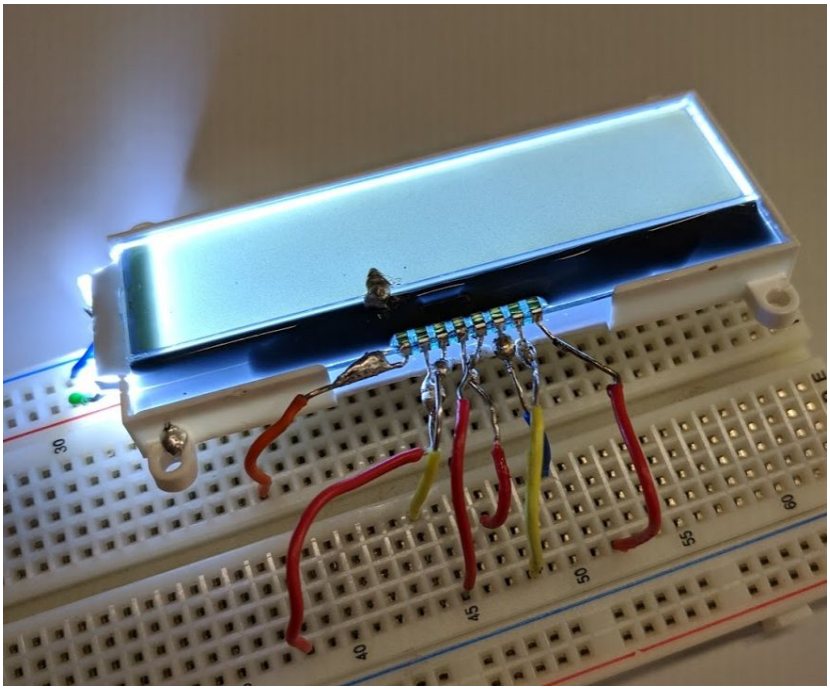


Figure 32: LCD Screen Testing

c. Connect a 1uF electrolytic capacitor across pins 7 and 8.
d. Connect a capacitor in the range 0.47uF - 2.2uF from pins 4 to 6(VOUT to VSS).
e. Measure and record the resistance of two 10k ohm resistors.
    i. Connect one across pins 2 to 5(SCLK to VDD) and one across pins 3 to 5(SDA to VDD).
f. Connect pin 5 (VDD) to the same supply rail that goes to A for the LCD backlight
g. Connect pin 4(VSS) to the same supply GND rail that goes to K for the LCD backlight.
h. Now that the screen is set up, look over the code on the datasheet for a software I2C implementation and view the timing diagram including how an ACK works and what the START and STOP signals should look like.
i. Implement the initialize software I2C function using the hardware I2C syntax.
j. Attach the logic analyzer to pins 03 for SCLK and 04 for SDA and 08 for RST.
    i. Be sure that all pins are high upon connecting the logic analyzer as they should all be high when idling.
    ii. If they are not, alter the pin config in CC3220SF_LAUNCHXL.c to set the output on pin 08 (GPIO17) to default to high.

k. Attempt to send the initialization sequence.
    i. If the address is being sent but nothing else is being sent, make sure that the pins on the data sheet match the pins on the breadboard. Note that the pins on page 3 are reversed in comparison to the pins on page 4.
    ii. If this introduces problems, reroute the wires on the breadboard to match the correct layout.
l. Once the initialization sequence completes, implement simple commands to set the cursor of the screen and to clear the screen in software.
m. Now try sending a simple "Hello World" command preceded by a setCursor(0,0) and a clear() to put it at the top left of the screen.
    i. If this command is not showing up on the screen, make sure that the capacitor values are correct and that the orientation of the electrolytic capacitor is correct.
n. Now that the screen is displaying unsigned char strings, implement the code to allow the screen to represent the temperature data in the main loop of the program.

Stage 2: PCB

Rotary Encoder:

a. Check that the connections from one resistor to another are correct using the virtual bench connection tester.
b. Connect the resistors R1,R7,R8,R9
c. Connect capacitors C1, C6
d. Connect the rotary encoder
e. Slot the LPB onto the launchpad
f. Connect the logic analyzer on pins p18, p06
g. Run the code and see if the logic analyzer is outputting the proper voltages
h. Now see if the global integer is varying in the positive direction with clockwise turns of the rotary encoder and in the negative direction with counterclockwise turns.

Figure 33: LPB Components Soldered on

LCD Screen:

a. Check that the PCB connections are valid before soldering any parts on, seeing that the data sheet pin layouts are reversed from page two to page three.
    i. If this is not the case, cut the traces using an exacto knife and solder on jumper wires to the correct locations.
b. Solder on the LCD screen, resistors R2 and R10 after measuring their values, and capacitors C2 and C7.
c. Any other pin changes that were necessary to be made on the breadboard iteration of testing must be made now, this includes rerouting pull-up resistors if the I2C control pins are changed.
    i. Changing I2C pins 03 and 04 to pins 01 and 02 will require changing the PWM control pin to pin 64 in order to not conflict with the new I2C mapping. This change will have to be reflected in the CC3220_LAUNCHXL.c file.
d. Test the software in the same way as it was done on the breadboard, if this does not work as intended, check the capacitor values and orientation and that the pull up resistors are connected to the proper pins after changes were made.

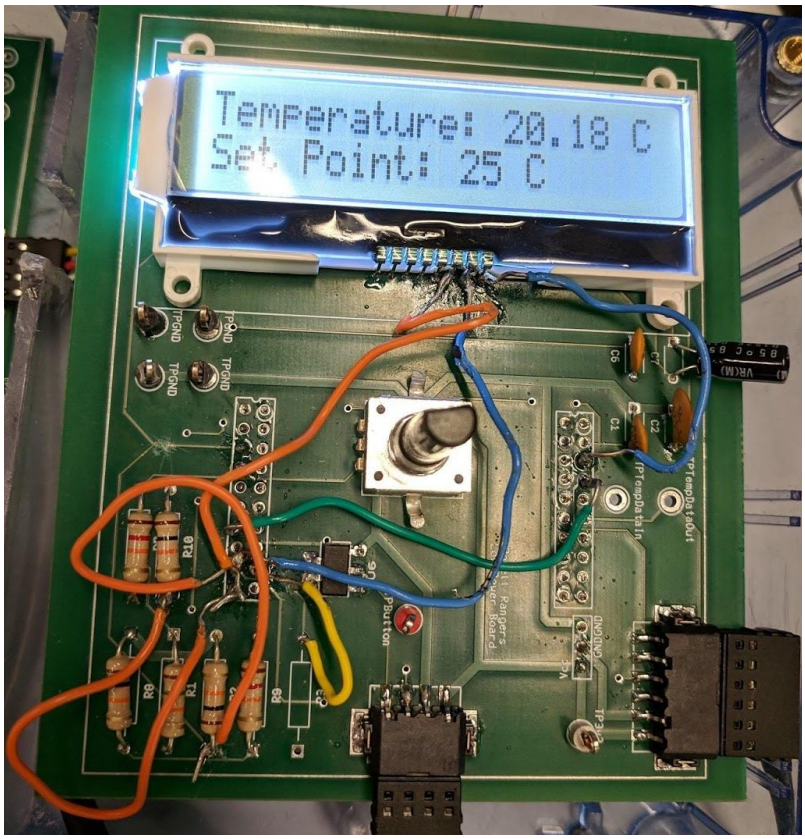


Figure 34: All Components Functioning Properly

Algorithm Testing

Fixed Point:

a. Conversions
    i. Start with a double precision floating point number
    ii. Multiply by the fixed point scale factor of 128 to convert to fixed point
    iii. Divide by the fixed point scale factor of 128 to convert back from fixed point
    iv. Check that the result is approximately equal to the original number, but with a few digits of precision lost and about log_2(7) digits accuracy
b. Arithmetic Operations
    i. Start with 2 double precision floating point numbers
    ii. Perform all possible arithmetic operations on them and keep track of the results
    iii. Convert them to fixed point
    iv. Perform all arithmetic operations on the fixed point version

v. Convert the fixed point results back to floating point and check that the fixed point calculations are approximately the same as the floating point calculations

Digital Filter:

a. Start with some pseudo random input and very noisy input using a random number generator function as an offset to some constant value (think SP = DC + rand() % K where K is the maximum possible offset from the DC)
b. Run the filter algorithm on the input and check that the noise has reduced
c. Ideally to test it properly for the frequency response, the following sequence of steps is needed:
    i. Use the function generator to generate a sinusoidal signal
    ii. Use the sinusoid as an input to the ADC of the microcontroller to extract digital values at some sampling frequency
    iii. Perform the filtering on the digital values
    iv. Reconstruct the digital filter output using a low pass filter that covers the bandwidth of the signal
    v. Measure the gain and plot the whole frequency response, comparing it to the expected analytical response

PID testing:

a. Start with some pseudo random input and very noisy input using a random number generator function as an offset to some constant value (think SP = DC + rand() % K where K - 1 is the maximum possible offset from the DC)
b. Set the input as the set point and keep track of the current output of the PID controller, initially zero. The control has finite response time and will not necessarily track the input with 100% accuracy
c. Compute the difference between the PID output and the set point for the error term and use that in the PID equation.
d. Update the output to the new return value and continue the loop
e. After testing this, switch the input stimulus to the temperature sensor readings and the output variable to the duty cycle of the PWM timer for illumination and the set point to the user input from the rotary encoder
f. Check the duty cycle outputs from the controller by steadily raising the temperature through the set point; return values should steadily decrease as error minimizes and should drop from the maximum to minimum value as the temperature rises past the set point

Integration Testing

Phase 1: Power Delivery to Lamp

a. Once the launchpad can control the 120VAC on the HPB via the LPB:
b. Connect the female power socket via three 14AWG wires to the female combicon
c. Plug this combicon into the male combicon on the HPB
d. Connect a 50w lamp to this female power port
e. Set the software to PWM P01 on the launchpad at 50% duty cycle
f. Load the PID controller code onto the launchpad
g. Connect the HPB to the wall socket
h. See whether the lamp is flickering or not
    i. If the lamp is not flickering, the issue could be related to the zero crossing pin as it is used to begin the PWM timer, make sure that this is not causing issues.
    ii. Another error could be caused if the temperature sensor is controlling the PID at this section as it could be telling the lamp that it needs to heat up or cool down significantly, changing the duty cycle from half to full or zero.

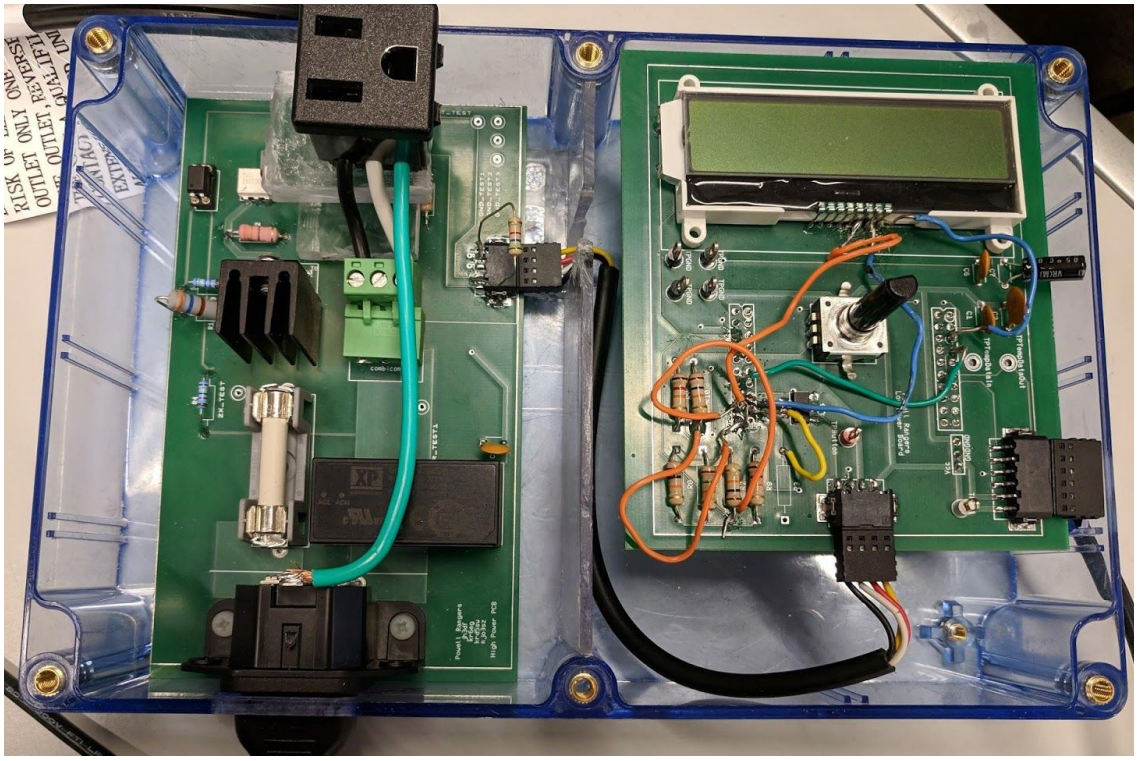
<u>Phase 2: Power Delivery to Lamp Based on Temperature Readings</u>

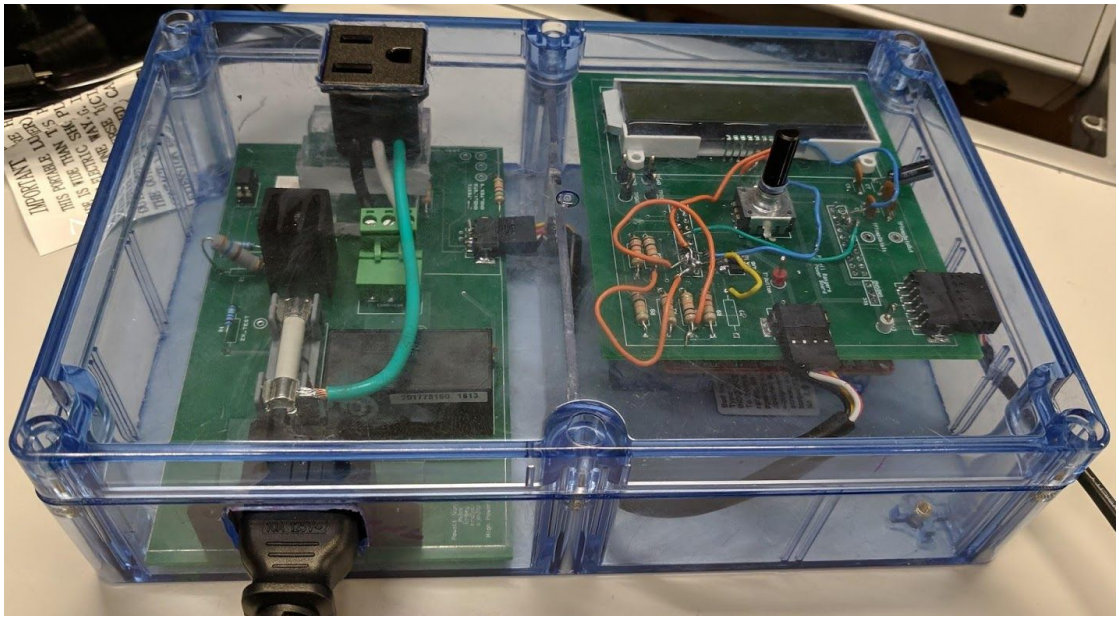
a. Once phase 1 is complete and the temperature sensor has been connected properly.
b. Connect a 50w lamp to the female power port
c. Set the software to PWM P01 on to be controlled via the temperature sensor readings and make the setpoint temperature 40 degrees celsius (assuming the room is 25C)
d. Load the PID controller code onto the launchpad
e. Plug the HPB into the wall and see that the lamp turns on completely to start with.
f. View the temperature readings via print statements and see that they go up over time. As the temperature approaches 40 degrees celsius, see that the lamp begins to flicker and eventually turns off. See that the temperature is within 2C of 40C from this point forward.
    i. If this level of accuracy is not reached, modify the PID parameters to be make the lamp respond more accurately.

<u>Phase 3: Final Integration</u>

a. At this point the rotary encoder and LCD screen should be integrated into the design.
b. Connect a 50w lamp to the female power port
c. Set the software to PWM P01 on to be controlled via the temperature sensor readings and make the setpoint temperature 40 degrees celsius (assuming the room is 25C)
d. Plug the HPB into the wall and see that the lamp turns on completely to start with.

e. Turn the rotary encoder counterclockwise and see that the change in setpoint temperature is reflected on the LCD screen.
f. As the setpoint goes below the current temperature, see that the lamp starts to flicker and then turns off.
g. Now turn the rotary encoder clockwise and see that the setpoint temperature goes higher
h. As the setpoint temperature goes above the current temperature, see that the lamp flickers and then turns back on.
i. If any of these parts fail to work properly, return to the PCB phase of their section of the test plan to resume testing that component.





Major Changes Between Test Plan Revisions:

The differences between our new and old test plan are huge as we were unable to foresee certain necessary changes that would have to be made to make the project work properly. For example, our temperature sensor changed, we added several new wires connecting our boards, we added a new PCB entirely, and we changed our zero crossing opto-isolator. Furthermore, the software was a much more vital part of our design than we originally assumed and, as such, it

played a part in nearly every subsystem testing section instead of being moved to the end as it was in the old test plan.

The order of our testing changed as we thought more about what parts were important to the functionality of our design. For instance, we knew in the new design that we would be able to have a functioning product without a voltage regulator because we could supply voltage from our computers so we instead focused on getting the zero crossing subsystem up and running earlier. This can also be seen in our focus on making the temperature sensor work earlier in the process.

The similarities between our old and new plans lie more in the pivotal testing points for any board, for instance, we check for shorts in both and make sure the header board fits properly on both. However, the newer board required more considerations as, when we got the board, we realized that our test points were too small to fit the test pins that were available. This resulted in more testing being done using the actual pins of devices which is something we would like to mitigate in the future.

Final Results

These were our original goals for the project:

D: Working CC3220SF header-board using manual loading of code to set temperature including working power delivery to the CC and heat lamp and accurate input readings from the thermometer.
C: Project includes an LCD display and allows users to set temperature using a rotary encoder.
B: Project includes internet connectivity allowing data transfer to/from a server.
A: Standalone PCB including all of the previous milestones OR fully functioning Android app working with the headerboard. We will try to make the standalone board but in the case that it was laid out incorrectly and we can't print another, we want to be able to focus our efforts on the Android app instead to get the points.

In total, our current iteration of the project accurately measures the temperature within about 0.008 degrees, updates an LCD screen with the current temperature and set point temperature, is interactive via the rotary encoder (adjusts the set point), and pulse width modulates the AC line at a frequency between 0 and 15 Hz in order to maintain the set point temperature. It also, from our demonstration, was able to capably handle a 200 W load without issues. We have little to no progress on the Android application, nor do we have much in the way of Internet provisioning. We were able to get an HTTP server running on the device but we did not successfully integrate it with the project code.

Comparing our results to the goals, we believe we fully completed everything up to the the 'C' mark. We had an HTTP server but it did not do anything at all, and certainly did not help our project in any way since we did not flesh it out, so we do not believe we hit the 'B' mark. As

for the ‘A’ mark, if we had completed everything earlier, then it would have been possible to get it, but in hindsight, our ‘A’ goal set a very high bar.

Costs

Parts Used for the Low Power Board

CC3220 Launchpad [13]
NHD-C0220BIZ LCD Screen [14]
2x 10x2 Header Connector [15]
Bourns PEC12R Rotary Encoder [16]
ZXMS600 MOSFET [17]
6 Position Terminal Female Block Header [18]
4 Position Terminal Female Block Header [19]
3 Position Header Connector [20]
2x 1µf cap
6x 10k resistor [21]
2x 0.01µf cap

Total Cost: $ 33.92

Parts Used for the High Power Board

GSP1.8101 Power Entry Connector Receptacle [22]
4 Position Terminal Female Block Header [19]
2x 14k resistors [23]
FOD814A Optoisolator [24]
Optoisolator mount [25]
Littelfuse Fuse [26]
Fuse Mount [27]
2x 360 resistors [28]
330 resistor [29]
T3035H Triac [30]
Triac Heatsink [31]
MOC3043M Optoisolator Triac [32]
3 Position Terminal Female Combicon Block Plug [33]
VCE03 AC/DC Converter 3.3V [34]
1x 1µf cap

Total Cost: $33.92

Parts Used for the Temperature Sensor Board

ADT7310 SPI Temperature Sensor [35]
6 Position Terminal Female Block Header [18]
0.1 µf bypass capacitor

Total Cost: $ 5.96

Cables and Other Connectors
6 Conductor Multi-Conductor Cable [36]
4 Conductor Multi-Conductor Cable [37]
6 Position Terminal Male Block Plug [38]
4 Position Terminal Male Block Plug [39]
3 Position Terminal Male Combicon Block Header [40]
Power Cord [41]

Total Cost: $12.94

Total Project Costs

Total Cost of 1 unit including the cables and the CC3220 Launchpad: $ 152.34
Total Cost of 10,000 units: $ 1,164,532
See appendix for the whole breakdown of component costs.

We definitely believe that buying parts in bulk would reduce costs in general; we can see that from the 10,000 unit cost, the average price of a device dropped by ~$36. One of the costliest parts on this list is the Launchpad itself, and all of the accessories needed to make the Launchpad fit into the board. If we were able to lay out a PCB with only what we needed, then we could cut costs by about ~$63 since we would be dropping the $51 LaunchPad and most of the cables / connectors. We would need to invest just a little bit to get the other chips that the board has in order to take advantage of the same architecture however, so that would probably drop savings by a decent amount. Still, this would amount to probably something like a 35% unit cost reduction, which would set aside a lot of funds for automation. And since the parts would be easier to work with if we just laid out a PCB, we would be able to automate all parts of the process.

However, with all of these costs in mind, it probably is not likely that this device would come very cheap. The Inkbird thermostats are extremely cheap, but Amazon reviews indicate that these devices have also been cheaply made. Cost saving measures were made in all of the wrong areas with devices like that. Our device has been overengineered for extremely high wattage and uses parts from fairly reputable brands so we think that the price could be justifiable to some degree.

Future Work

The chief pitfalls with our current iteration of the project are a lack of smooth dimming, a lack of IoT support, and perhaps program efficiency / correctness. Attacking each problem one by one, some possible improvements are as follows:

1. Dimming:
    a. Frequency multiplier: The 60 Hz line input could be frequency multiplied to some large multiple like 720 Hz, then the PID algorithm and the integer cycle PWM timer could be used with a much higher resolution and a much faster update rate. This would result in the actual ‘dimming’ action. To invert the frequency multiplication, a low pass filter could be used to take the multiplied frequency back down to 60 Hz on the line. This multiplication factor would ideally have to be a power of two to allow our fixed point calculations to directly represent every possible duty cycle output with full accuracy.
    b. Rectifier: The input to the lamp could be rectified to DC with a full bridge rectifier so that the DC amplitude would scale linearly with the number of half cycles in the wave directly. The backwards conversion for this would probably be much more difficult, since it would be essentially a DC neutral voltage being converted back to 60 Hz AC.
2. IoT Support:
    a. Provisioning: We could use the TI Provisioning source code to create automatic network connection support for the device. This would employ some standard technology like WiFi Direct or TI SmartConfig. This would then hook into a compatible app.
    b. IoT App: We would have to set aside time for and develop the IoT app that provisions the device directly through the phone, provides a decent user interface, and functions. This would include registering the temperature, set point, user input to change the set point, in-app WiFi provisioning, the temperature scheduling system, alerts, and potentially even a database for reptiles that contains some ‘default’ schedules for different reptiles.
3. Program Efficiency / Correctness:
    a. Our project at present is relatively low frequency in terms of PID rates, temperature acquisition, and output resolution. Thus the performance hit from the RTOS performing lots of different tasks is negligible because of the spacing between the events.

b. With the IoT support and much higher frequency PID, temperature sensing, etc. it would be extremely important to properly assign priorities for the RTOS scheduler, and decide on a proper event loop in order to ensure consistent performance.
c. With the solutions above and the things below, it would also be possible to improve program correctness:
    i. More sensors for better temperature sensing, and correcting for heat diffusion into the air in order to get accurate temperature predictions and better response times on the control algorithms. The temperature sensor is not directly under the light, so it could potentially report a different temperature.
    ii. Including some sort of auto-tuning heuristic to improve the quality of the PID algorithms over time.

References

[1] "Amazon.com : iPower 40-108°F Digital Heat Mat Thermostat Controller for Seed Germination, Reptiles and Brewing : Pet Supplies." [Online]. Available: https://www.amazon.com/iPower-40-108%C2%B0F-Thermostat-Controller-Germination/dp/B01E9IO6N0. [Accessed: 17-Dec-2018].

[2] "Temperature Controller ITC-308 | INKBIRD." [Online]. Available: https://www.ink-bird.com/products-temperature-controller-itc308.html. [Accessed: 17-Dec-2018].

[3] "Google Sheets: Free Online Spreadsheets for Personal Use." [Online]. Available: https://www.google.com/sheets/about/. [Accessed: 16-Dec-2018].

[4] "What Is Multisim™ for Education - National Instruments." [Online]. Available: http://www.ni.com/en-us/shop/electronic-test-instrumentation/application-software-for-electronic-test-and-instrumentation-category/what-is-multisim/multisim-education.html. [Accessed: 16-Dec-2018].

[5] "Ultiboard™ Software - National Instruments." [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/en/nid/203428. [Accessed: 16-Dec-2018].

[6] "What Is VirtualBench? - National Instruments." [Online]. Available: http://www.ni.com/en-us/shop/electronic-test-instrumentation/virtualbench/what-is-virtualbench.html. [Accessed: 16-Dec-2018].

[7] "CCSTUDIO Code Composer Studio (CCS) Integrated Development Environment (IDE) | TI.com." [Online]. Available: http://www.ti.com/tool/CCSTUDIO. [Accessed: 16-Dec-2018].

[8] "PINMUXTOOL Pin Mux Tool | TI.com." [Online]. Available: http://www.ti.com/tool/PINMUXTOOL. [Accessed: 16-Dec-2018].

[9] “UNIFLASH Uniflash Standalone Flash Tool for TI Microcontrollers (MCU), Sitara Processors & SimpleLink devices | TI.com.” [Online]. Available: http://www.ti.com/tool/UNIFLASH. [Accessed: 16-Dec-2018].
[10] A. Tubi and E. Nuriel, “Method and device for WiFi controlled electric switch,” EP2860717A1, 15-Apr-2015.
[11] “US Patent Application for TEMPERATURE CONTROL DEVICE Patent Application (Application #20150096734 issued April 9, 2015) - Justia Patents Search.” [Online]. Available: https://patents.justia.com/patent/20150096734. [Accessed: 16-Dec-2018].
[12] “US Patent for Control system for thermoelectric devices Patent (Patent # 9,739,512 issued August 22, 2017) - Justia Patents Search.” [Online]. Available: https://patents.justia.com/patent/9739512. [Accessed: 17-Dec-2018].
[13] “CC3220SF-LAUNCHXL Texas Instruments | RF/IF and RFID | DigiKey.” [Online]. Available: https://www.digikey.com/products/en/rf-if-and-rfid/rf-evaluation-and-development-kits-boards/859?k=CC3220SF. [Accessed: 16-Dec-2018].
[14] “NHD-C0220BIZ-FSW-FBW-3V3M Newhaven Display Intl | Optoelectronics | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/newhaven-display-intl/NHD-C0220BIZ-FSW-FBW-3V3M/NHD-C0220BIZ-FSW-FBW-3V3M-ND/2626407. [Accessed: 16-Dec-2018].
[15] “PPPC102LFBN-RC Sullins Connector Solutions | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.ca/product-detail/en/sullins-connector-solutions/PPPC102LFBN-RC/S6106-ND/807245. [Accessed: 16-Dec-2018].
[16] “PEC12R-4030F-S0024 Bourns Inc. | Sensors, Transducers | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/bourns-inc/PEC12R-4030F-S0024/PEC12R-4030F-S0024-ND/6153385. [Accessed: 16-Dec-2018].
[17] “ZXMS6004DGTA Diodes Incorporated | Integrated Circuits (ICs) | DigiKey.” [Online]. Available: https://www.digikey.ca/product-detail/en/diodes-incorporated/ZXMS6004DGTA/ZXMS6004DGTACT-ND/2192654. [Accessed: 16-Dec-2018].
[18] “1778803 Phoenix Contact | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/phoenix-contact/1778803/277-2319-1-ND/2625589. [Accessed: 17-Dec-2018].
[19] “1778780 Phoenix Contact | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/phoenix-contact/1778780/277-2317-1-ND/2625587. [Accessed: 17-Dec-2018].
[20] “PPPC031LFBN-RC Sullins Connector Solutions | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.ca/product-detail/en/sullins-connector-solutions/PPPC031LFBN-RC/S7036-ND/810175. [Accessed: 17-Dec-2018].
[21] “CFR-25JB-52-10K Yageo | Resistors | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/yageo/CFR-25JB-52-10K/10KQBK-ND/338. [Accessed: 17-Dec-2018].

[22] “GSP1.8101.1 Schurter Inc. | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/schurter-inc/GSP1.8101.1/486-1007-ND/569907. [Accessed: 17-Dec-2018].
[23] “MRS25000C1402FCT00 Vishay BC Components | Resistors | DigiKey.” [Online]. Available: https://www.digikey.ca/products/en/resistors/through-hole-resistors/53?k=&pkeyword=14k+resistor&umin2085=13.9&umax2085=14.1&rfu2085=kOhms&pv2=191&pv2=41&pv2=181&pv2=193&pv2=6&pv2=68&pv2=143&pv2=53&pv2=561&pv2=7&pv2=838&pv2=542&pv2=8&pv2=858&pv2=827&pv2=543&pv2=161&pv2=828&pv2=128&pv2=105&pv2=1399&pv2=9&pv2=1400&pv2=173&pv2=675&pv2=103&pv2=602&pv2=188&pv2=98&pv2=10&pv2=928&pv2=106&pv2=77&pv2=104&pv2=554&pv2=76&pv2=16&pv2=117&pv2=109&pv2=555&pv2=11&pv2=12&pv2=17&pv2=544&pv2=239&pv2=81&pv2=13&pv2=1199&pv2=14&pv2=127&sf=1&FV=ffe00035&quantity=&ColumnSort=0&page=1&stock=1&pageSize=25. [Accessed: 17-Dec-2018].
[24] “FOD814A ON Semiconductor | Isolators | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/on-semiconductor/FOD814A/FOD814A-ND/1143906. [Accessed: 17-Dec-2018].
[25] “110-87-304-41-001101 Preci-Dip | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/preci-dip/110-87-304-41-001101/1212-1001-ND/3757251. [Accessed: 17-Dec-2018].
[26] “0332009.HXP Littelfuse Inc. | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.ca/products/en?keywords=Littelfuse%20Inc.%200332009.HXP. [Accessed: 17-Dec-2018].
[27] “03540101ZXGY Littelfuse Inc. | Circuit Protection | DigiKey.” [Online]. Available: https://www.digikey.com/products/en/circuit-protection/fuseholders/140?k=&pkeyword=&pv454=158&sf=1&FV=fffc0012%2Cc40054%2C1af00002%2Cmu100A%7C2088%2Cmu10A%7C2088%2Cmu15A%7C2088%2Cmu200A%7C2088%2Cmu20A%7C2088%2Cmu30A%7C2088%2Cmu400A%7C2088%2Cmu600A%7C2088%2Cmu60A%7C2088%2Cffe0008c%2C114007b&quantity=&ColumnSort=0&page=1&pageSize=25. [Accessed: 17-Dec-2018].
[28] “RSF200JB-73-360R Yageo | Resistors | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/yageo/RSF200JB-73-360R/360W-2-ND/18279. [Accessed: 17-Dec-2018].
[29] “CFR-25JB-52-330R Yageo | Resistors | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/yageo/CFR-25JB-52-330R/330QBK-ND/1636. [Accessed: 17-Dec-2018].
[30] “T3035H-6T STMicroelectronics | Discrete Semiconductor Products | DigiKey.” [Online]. Available: https://www.digikey.ca/product-detail/en/stmicroelectronics/T3035H-6T/497-10583-5-ND/2294981. [Accessed: 17-Dec-2018].
[31] “581002B00000 Aavid, Thermal Division of Boyd Corporation | Fans, Thermal Management | DigiKey.” [Online]. Available: https://www.digikey.ca/product-detail/en/aavid-thermal-division-of-boyd-corporation/581002B00000/HS300-ND/373776. [Accessed: 17-Dec-2018].

[32] “MOC3043M ON Semiconductor | Isolators | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/on-semiconductor/MOC3043M/MOC3043M-ND/281232. [Accessed: 17-Dec-2018].
[33] “1757022 Phoenix Contact | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/phoenix-contact/1757022/277-1012-ND/260380. [Accessed: 17-Dec-2018].
[34] “VCE03US03 XP Power | Power Supplies - Board Mount | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/xp-power/VCE03US03/1470-4517-ND/8021445. [Accessed: 17-Dec-2018].
[35] “ADT7310TRZ-REEL7 Analog Devices Inc. | Sensors, Transducers | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/analog-devices-inc/ADT7310TRZ-REEL7/ADT7310TRZ-REEL7CT-ND/3674808. [Accessed: 17-Dec-2018].
[36] “30-00510 Tensility International Corp | Cables, Wires | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/tensility-international-corp/30-00510/T1347-1-ND/6245003. [Accessed: 17-Dec-2018].
[37] “1778874 Phoenix Contact | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/phoenix-contact/1778874/277-2326-ND/2625560. [Accessed: 17-Dec-2018].
[38] “30-00386 Tensility International Corp | Cables, Wires | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/tensility-international-corp/30-00386/T1286-1-ND/5270260. [Accessed: 17-Dec-2018].
[39] “1757255 Phoenix Contact | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/phoenix-contact/1757255/277-1107-ND/260475. [Accessed: 17-Dec-2018].
[40] “1778858 Phoenix Contact | Connectors, Interconnects | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/phoenix-contact/1778858/277-2324-ND/2625558. [Accessed: 17-Dec-2018].
[41] “3021451F5701(R) GlobTek, Inc. | Cable Assemblies | DigiKey.” [Online]. Available: https://www.digikey.com/product-detail/en/globtek-inc/3021451F5701-R/1939-1013-ND/8597816. [Accessed: 17-Dec-2018].
[42] “US Patent for Programmable smart thermostat Patent (Patent # 9,857,090 issued January 2, 2018) - Justia Patents Search.” [Online]. Available: https://patents.justia.com/patent/9857090. [Accessed: 17-Dec-2018].
[43] “Seizure triggers for people with photosensitive epilepsy | Epilepsy Action.” [Online]. Available: https://www.epilepsy.org.uk/info/photosensitive-epilepsy/triggers. [Accessed: 17-Dec-2018].
[44] “About NEMA Standards - NEMA.” [Online]. Available: https://www.nema.org/Standards/About-Standards/pages/default.aspx. [Accessed: 17-Dec-2018].
[45] “NFPA - About the National Electrical Code®.” [Online]. Available: https://www.nfpa.org/NEC. [Accessed: 17-Dec-2018].
[46] “I2C - What’s That?,” *I2C Bus*. [Online]. Available: /http://www.i2c-bus.org/i2c-bus/. [Accessed: 17-Dec-2018].

[47] “Standards | IPC.” [Online]. Available: http://www.ipc.org/ContentPage.aspx?pageid=Standards. [Accessed: 17-Dec-2018].
[48] “SPI software coding | EDN.” [Online]. Available: https://www.edn.com/design/systems-design/4327989/SPI-software-coding. [Accessed: 17-Dec-2018].
[49] “filters - Exponential moving average cut-off frequency,” *Signal Processing Stack Exchange*. [Online]. Available: https://dsp.stackexchange.com/questions/40462/exponential-moving-average-cut-off-frequency. [Accessed: 17-Dec-2018].
[50] “Desmos graph,” *Desmos Graphing Calculator*. [Online]. Available: https://www.desmos.com/calculator/z0bb3peysu. [Accessed: 17-Dec-2018].
[51] “final paper costs,” *Google Docs*. [Online]. Available: https://docs.google.com/spreadsheets/d/1-n_BWXe8eIlCOvU-A_hyroLQGd1-HWhrPnwhMvpnM30/edit?usp=embed_facebook. [Accessed: 17-Dec-2018].

Appendix

Item	Price per unit	Number of units	Total Unit Cost	Price per unit WHOLESALE	Number of units WHOLESALE	Total WHOLESALE	Part Number	Link	Distributor	Datasheet	Size	Diameter
CC3220SF Launchpad	51.91000	1	51.91000	51.91000	10000	519,100.00000	296-46388-ND	https://www.digikey.com/products/en/rf-if-and-rfid/rf	TI	http://www.ti.com	NA	NA
LCD Display	10.40000	1	10.40000	7.92000	10000	79,200.00000	NHD-C0220BIZ-FSW-FBW-3V3M-ND	https://www.digikey.com/product-detail/en/newhaven	Newhaven Display	http://www.newh	60mmx15.1mm	NA
FOD	0.59000	1	0.59000	0.16703	10000	1,670.30000	FOD814A-ND	https://www.digikey.com/product-detail/en/on-semi	ON	http://www.onsen	NA	NA
FOD Mount	0.27000	1	0.27000	0.10280	10000	1,028.00000	1212-1061-ND	https://www.digikey.com/product-detail/en/preci-dip	Preci-Dip	http://www.preci	NA	NA
14k Resistors	0.41000	2	0.82000	0.05685	20000	1,137.00000	BC3694CT-ND	https://www.digikey.ca/products/en/resistors/through	Vishay	http://www.vishay	NA	NA
330 Resistor	0.35000	1	0.35000	0.00876	10000	87.60000	330ZCT-ND	https://www.digikey.com/product-detail/en/yageo/CF	Yageo	http://www.yageo	NA	NA
360 Resistor	0.44000	2	0.88000	0.05243	20000	1,048.60000	360W-2-ND	https://www.digikey.com/product-detail/en/yageo/RS	Yageo	http://www.yageo	NA	NA
tmp sensor ADT7310TRZ	3.93000	1	3.93000	2.46212	10000	24,621.20000	ADT7310TRZ-REELTCT-ND	https://www.digikey.com/product-detail/en/analog-d	Analog Devices	https://www.analo	8.00IC	NA
MOSFET	1.43000	1	1.43000	1.11248	10000	11,124.80000	ZXM5406GTACT-ND	https://www.digikey.ca/product-detail/en/diodes-inc/	Diodes Inc.	https://www.diod	SOT-223	NA
Rotary Encoder	1.25000	1	1.25000	0.60200	10000	6,020.00000	PEC12R-4030F-S0024-ND	https://www.digikey.com/product-detail/en/bourns-in	Bourns Inc.	https://www.bour	13.4mmx12.4mm	NA
Opto-Triac-Isolator	1.08000	1	1.08000	0.35871	10000	3,587.10000	MOC3043M-ND	https://www.digikey.com/product-detail/en/on-semic	ON semiconductor	https://www.fairc	6.1mmx6.13mm	NA
Triac	4.48000	1	4.48000	2.42772	10000	24,277.20000	497-10583-5-ND	https://www.digikey.ca/product-detail/en/stmicroelec	STMicroelectronics	https://www.st.co	15.2mmx10mm	NA
Triac Heatsink	1.97000	1	1.97000	1.36201	10000	13,620.10000	HS300-ND	https://www.digikey.ca/product-detail/en/aavid-therm	Aavid	https://media.digi	1"x0.94"	NA
Fuse	7.42000	1	7.42000	3.60617	10000	36,061.70000	F7959-ND	https://www.digikey.ca/products/en?keywords=Littel	Littelfuse Inc.	http://www.littelfu	1/4"x1 1/4"	NA
Fuse holder	1.25000	1	1.25000	0.90700	10000	9,070.00000	F1468-ND	https://www.digikey.com/products/en/circuit-protect	Littelfuse Inc.	https://www.littel	1/4"x1 1/4"	NA
AC/DC Converter 120VAC-3.3VDC	9.75000	1	9.75000	8.16900	10000	81,690.00000	1470-4017-ND	https://www.digikey.com/product-detail/en/xp-power	XP Power	https://www.xpp	1.6"x0.75"x0.75"	NA
NEMA Enclosure	28.10000	1	28.10000	19.08550	10000	190,855.00000	377-1544-ND	https://www.digikey.com/product-detail/en/bud-indu	Bud Industries	https://www.budi	8.740" x 5.740" x 2.	NA
10k resistors	0.10000	6	0.60000	0.00876	60000	525.60000	10KQBK-ND	https://www.digikey.com/product-detail/en/yageo/CF	Yageo	http://www.yageo	NA	NA
6 pin cable	2.81000	1	2.81000	1.67500	10000	16,750.00000	T1M7-1-ND	https://www.digikey.com/product-detail/en/tensility	Tensility International	http://www.tensility.com/pdffiles/30-00510.pdf		
10x2 connector	1.81000	2	3.62000	0.96853	20000	19,370.60000	S6106-ND	https://www.digikey.ca/product-detail/en/sullins-con	Sullins Connector So	https://drawings-pdf.s3.amazonaws.com/10492.pdf		
4 pin cable	3.07000	1	3.07000	2.04100	10000	20,410.00000	277-2317-1-ND	https://www.digikey.com/product-detail/en/tensility-i	Tensility International	http://www.tensility.com/pdffiles/30-00386.pdf		
3x1 pin holder	0.53000	1	0.53000	0.24260	10000	2,426.00000	57036-ND	https://www.digikey.ca/product-detail/en/sullins-con	Sullins Connector So	https://drawings-pdf.s3.amazonaws.com/10492.pdf		
power cable	2.40000	1	2.40000	1.27972	10000	12,797.20000	1956-1013-ND	https://www.digikey.com/product-detail/en/globtek-i	GlobTek Inc	https://www.globt	9.28"	0.185"
female socket	2.41000	1	2.41000	0.91080	10000	9,108.00000	486-1007-ND	https://www.digikey.com/product-detail/en/schurter-	Schurter Inc	https://us.schurte	30.30mm x 22.50m	NA
combicon female	2.98000	1	2.98000	1.97107	10000	19,710.70000	277-1012-ND	https://www.digikey.com/product-detail/en/phoenix-	Phoenix Contact	https://www.digikey.com/product-detail/en/phoenix-contact/1757022/277-1012-ND/260380		
combicon male	1.03000	1	1.03000	0.65436	10000	6,543.60000	277-1107-ND	https://www.digikey.com/product-detail/en/phoenix-	Phoenix Contact	https://media.digikey.com/pdf/Data%20Sheets/Phoenix%20Contact%20PDFs/1757255.pdf		
gpio transfer male 4 pin	1.49000	1	1.49000	1.05313	10000	10,531.30000	277-2324-ND	https://www.digikey.com/product-detail/en/phoenix-	Phoenix Contact	https://media.digikey.com/PDF/Data%20Sheets/Phoenix%20Contact%20PDFs/1778858.pdf		
gpio transfer female 4 pin	1.35000	1	1.35000	1.11226	10000	11,122.60000	277-2317-1-ND	https://www.digikey.com/product-detail/en/phoenix-	Phoenix Contact	https://media.digikey.com/PDF/Data%20Sheets/Phoenix%20Contact%20PDFs/1778769.pdf		
gpio transfer male 6 pin	2.14000	1	2.14000	1.57976	10000	15,797.60000	277-2326-ND	https://www.digikey.com/product-detail/en/phoenix-	Phoenix Contact	https://media.digikey.com/PDF/Data%20Sheets/Phoenix%20Contact%20PDFs/1778874.pdf		
gpio transfer female 6 pin	2.03000	1	2.03000	1.67402	10000	16,740.20000	277-2319-1-ND	https://www.digikey.com/product-detail/en/phoenix-	Phoenix Contact	https://media.digikey.com/PDF/Data%20Sheets/Phoenix%20Contact%20PDFs/1778803.pdf		
		Low Power Board:	75.12000									
		High Power Board:	33.92000		Total	1,164,532						
		Temp Sensor Board:	5.96000		Unit Count	10,000						
		Total Cost:	152.34000									
		Cable Costs and other:	12.94000									
0.1 uf capacitor		1	0.00000									
1 uf capacitor		3	0.00000									
0.01 uf capacitor		2	0.00000									

Figure 37: Comprehensive Breakdown of Component Cost [51]

-3 dB frequency calculations [49]

Frequency response analysis for magnitude and phase of EMA [50]

Important Notes

1. There is a defect with the screen that causes it to not load characters when the device is plugged into the wall about half of the time. The rest of the board functions the same but the screen displays nothing. If this happens during testing the device, unplugging it and plugging it back in usually fixes the problem.
2. We decided to go with using half-cycles instead of full cycles to increase the resolution of our PWM. In a future revision of the device we would definitely do away with this as we understand that it introduces a DC voltage on the line, negatively affecting the performance of other devices on the same circuit.
3. As you will see in the testing images below, there was a glaring lack of test points on our circuit boards; it is safe to say that we have learned why their use was so stressed in our earlier classes. We found ways around this issue with these boards but it doesn't change the fact that it was bad design practice to leave them off. This is something we don't plan to repeat.
4. Our rotary encoder came with pushbutton functionality that is connected to a pin on our board, however, we didn't have time to implement a function for this button. Our idea was to maybe use it to switch between celsius and fahrenheit for displaying the temperature, ut there are doubtless more useful things that the button could be used, especially if we designed a more comprehensive user interface than the one we ended up using.

Test Results

MOSFET+MOC+Triac+FOD on Breadboard Testing

- We wanted to determine whether our general setup for switching the AC line would work, so we set up a breadboard with all of the necessary components on it and set about debugging it.
- We used the virtual bench function generator to put a 24VAC voltage on the triac and hooked up the DC supply to where the voltage regulator would be supplying current in our circuit.

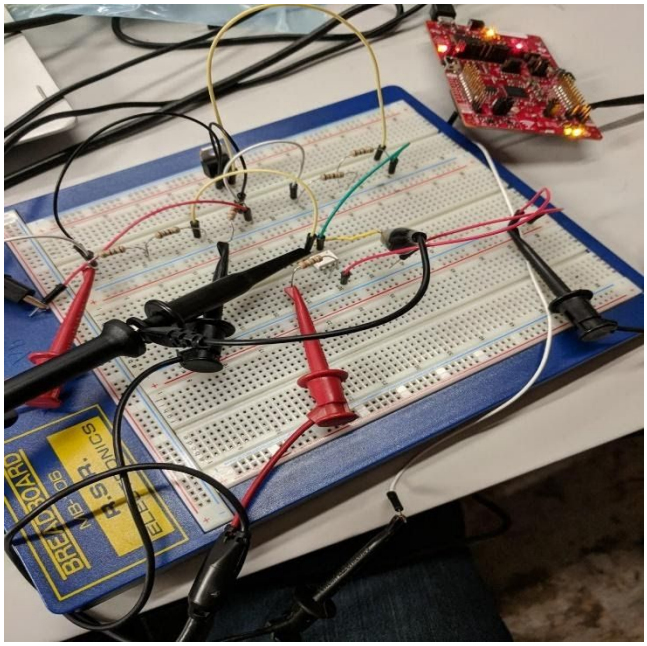


Figure 38: Testing the AC Line using 24 VAC from the VirtualBench

- Our first attempt had no chance of working as we had not implemented the MOSFET yet. This meant we did not get beyond testing if the MOC switched. We gave it a few days and then installed the MOSFET when we got it in the mail. Sadly, the MOSFET was a surface mount part so we had to solder jumper wires to its pins to make the setup feasible.

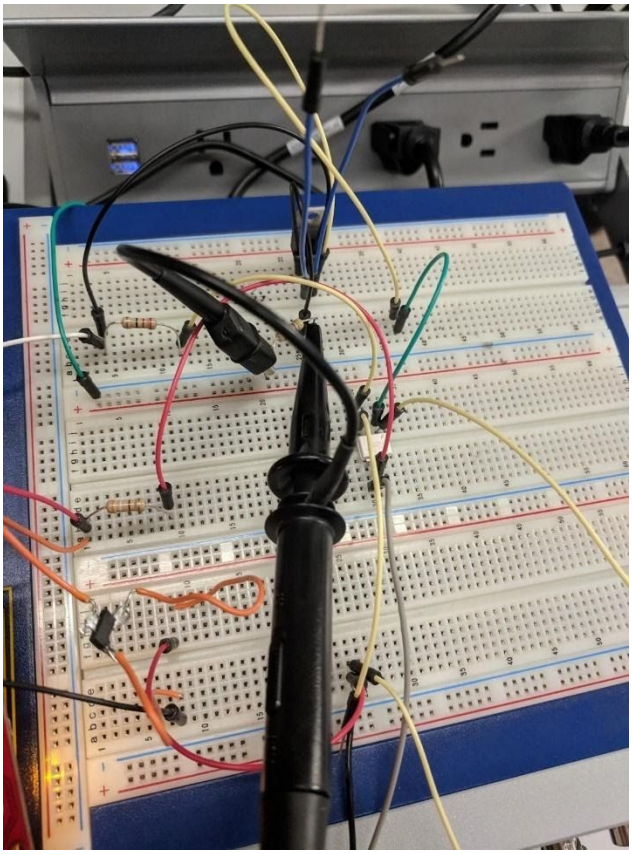


Figure 39: Debugging the Triac

- After connecting the MOSFET, we could get the MOC to switch and we saw that our MOC resistor calculation was putting 6mA through, however, we could not get the Triac to switch. This took a while to figure out but we had the Triac backwards. We had assumed that A1 and A2 were reversible due to the schematic for the device, but it turned out that was not the case.

Figure 40: FOD Zero Crossing

- With the triac switching working, we moved onto setting up the FOD zero crossing. The circuitry for this was simple as described in the test plan, but we had difficulty making the interrupts that it generated actually sync the PWM up.
- We also had trouble making the interrupts not trigger several times during the peak where the zero crossing occured. This was solved using a pullup resistor to make the peak more sharp and by effectively software debouncing the pin so that it could only be interrupted once per second.

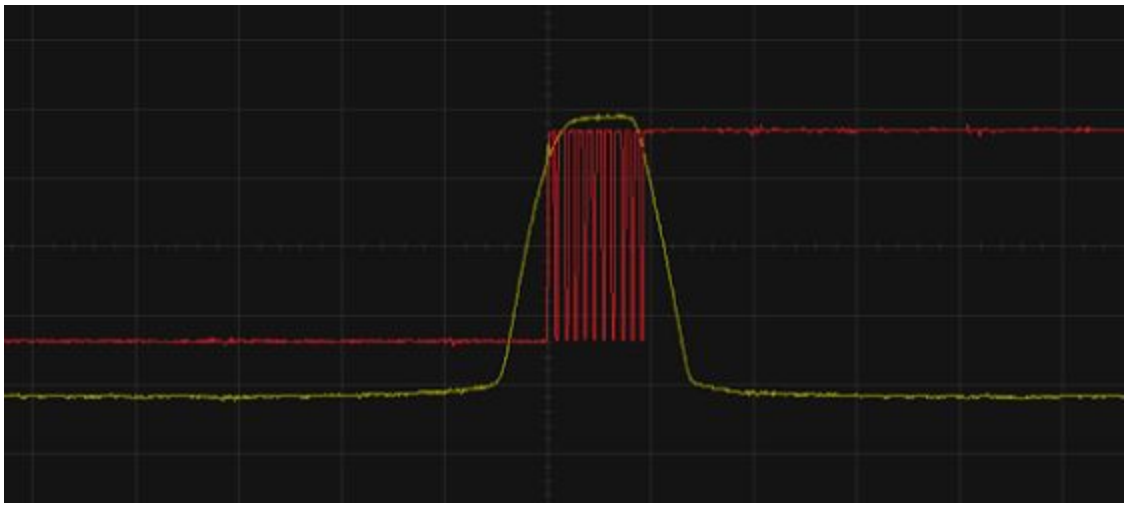


Figure 41: Interrupts Triggering Several Times During Zero Crossing Peaks

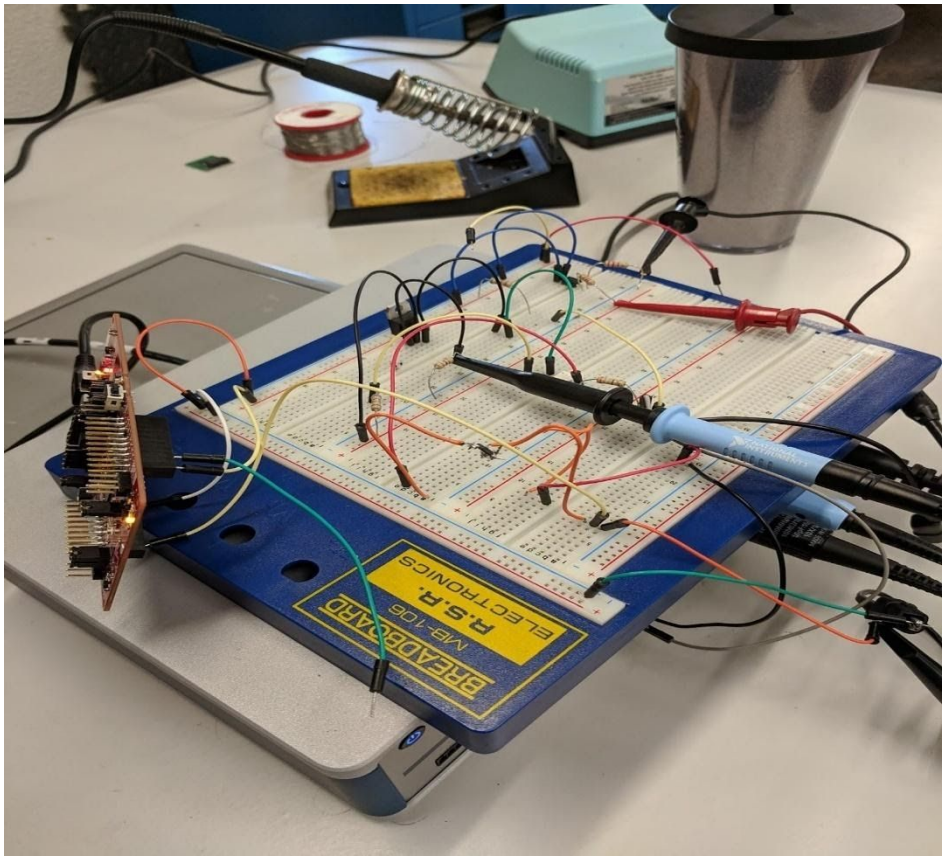


Figure 42: Setup of PWM Zero Crossing Testing

- After several hours of messing with this, we realized that turning on and off the PWM did not reset its clock. We changed our code to set the Duty to 0 and back to its full value and then we saw the expected behavior.

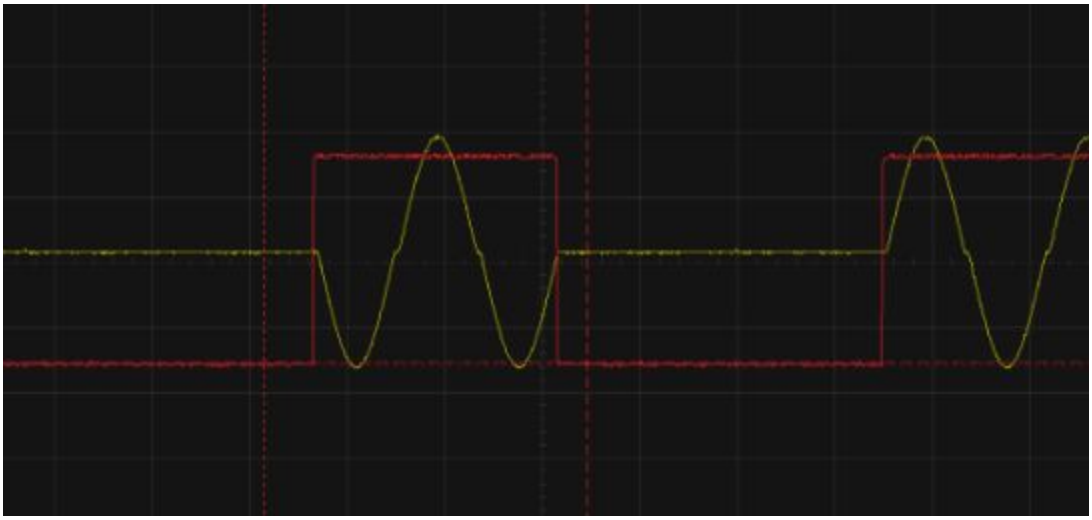


Figure 43: PWM Working Properly

HPB Connectivity Test

- First we tested that the input voltage at the 120VAC input was propagating through the fuse using the virtual bench connected with 12VAC @ 60Hz.

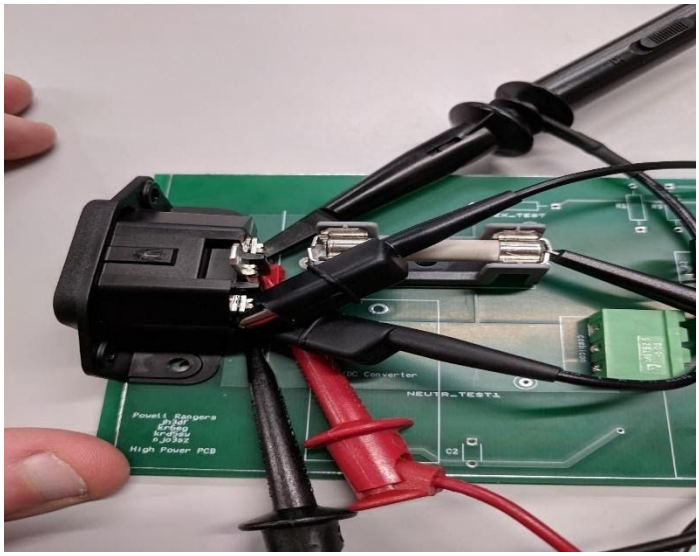


Figure 44: Testing Voltage Propagation Through the Fuse Using 12 Vpp at 60 Hz

Figure 45: Verification of Voltage Propagation Through the Fuse

- Next, we connected the triac and made sure that the voltage was reaching its input pin, pin A2.
- We also connected the male combicon to the board at this point to have another pin that we could connect to that referenced the same ground as the virtual bench. This was because one side of the combicon was connected to neutral which was connected to the ground of the virtual bench.

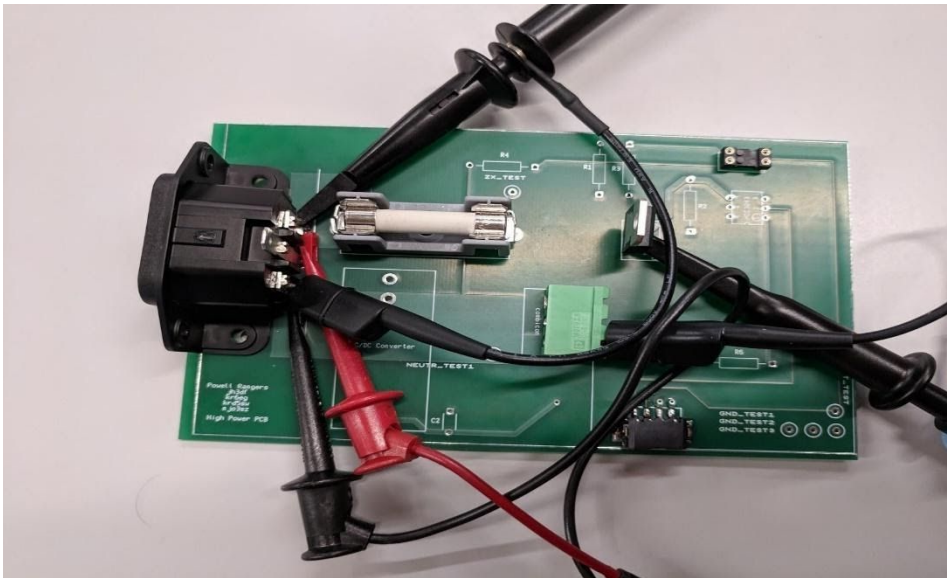


Figure 46: Set Up for Voltage Propagation Through the Triac

Figure 47: Verification of Voltage Propagation Through the Triac

<u>MOC + Triac Testing HPB #3 at 24VAC via Virtual Bench:</u>

- R1 calculation: We don't want more than 1A through the MOC as that is its absolute maximum rating. We provide 24vpp and through Ohm's law, V=IR so 24/R < 1. As such, we use a 30 ohm resistor.
- When switching to high voltage, we use 360ohms i.e. 120v/360 = 333mA < 1amp.
- Individual resistor value: R1 = 33.004 ohms.

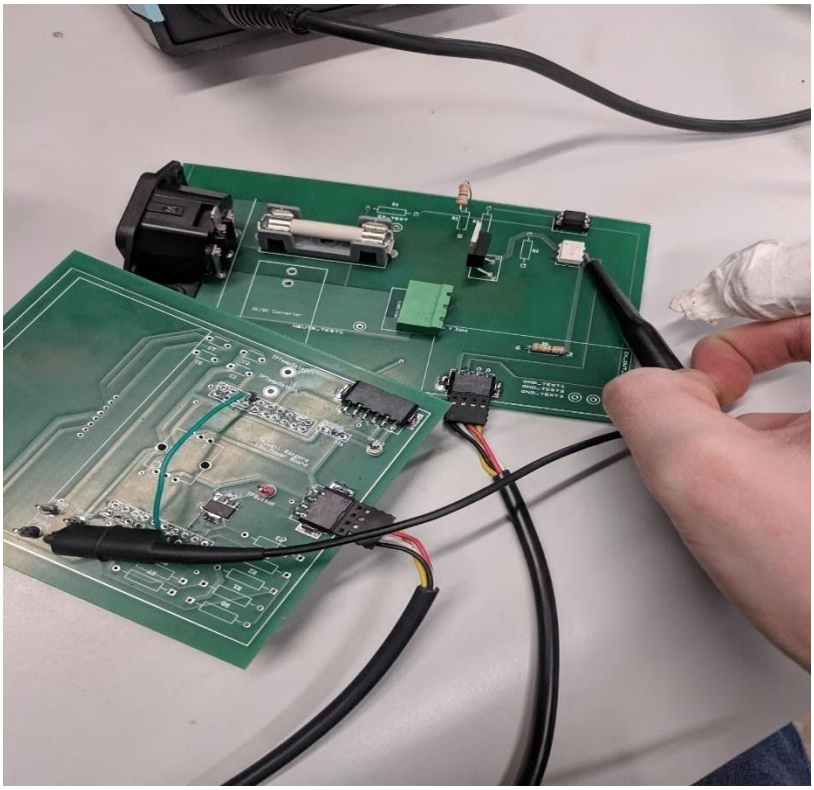


Figure 48: Set Up for MOC Functionality Testing

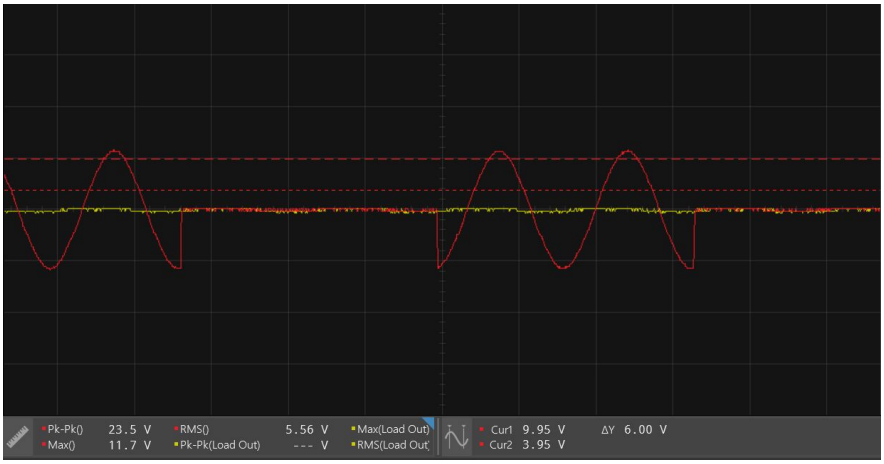


Figure 49: Voltage Across the Triac at 24 Vpp at 4/8 Half-Cycles

- Next we tested with 24Vpp across a 180kOhm load to be sure that the proper voltages were propagating. At this point, the zero crossing circuitry was not in use, as such we see fractions of half-cycles being sent.

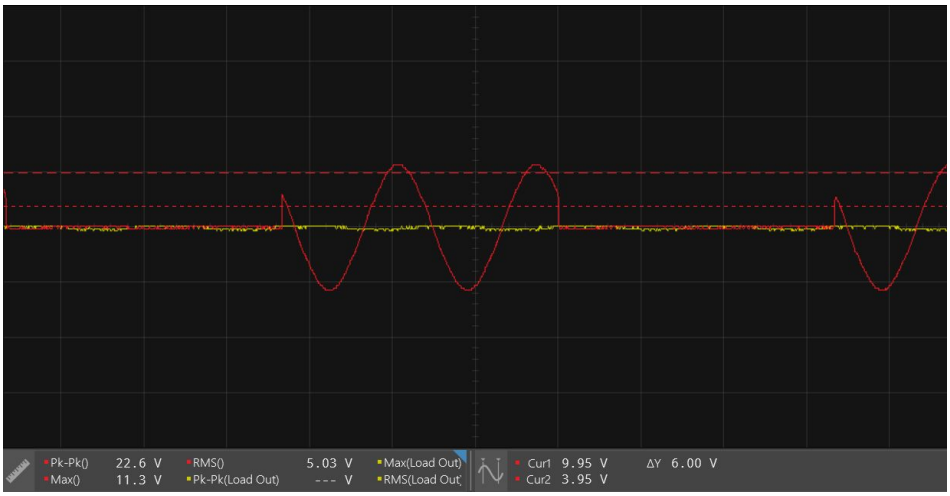


Figure 50: Voltage Across Load at 24 Vpp at 4/8 Half-Cycles

FOD Zero Crossing Testing HPB #3 at 24VAC via Virtual Bench:

- We began by connecting 2.2k resistor for R3 and R4 to limit the current through the diode in the FOD.
- R3 = 2.168k
- R4 = 2.1733k
- We then connected a 5.6kOhm pullup resistor to P15 to increase the height of the peaks experienced on zero crossings.

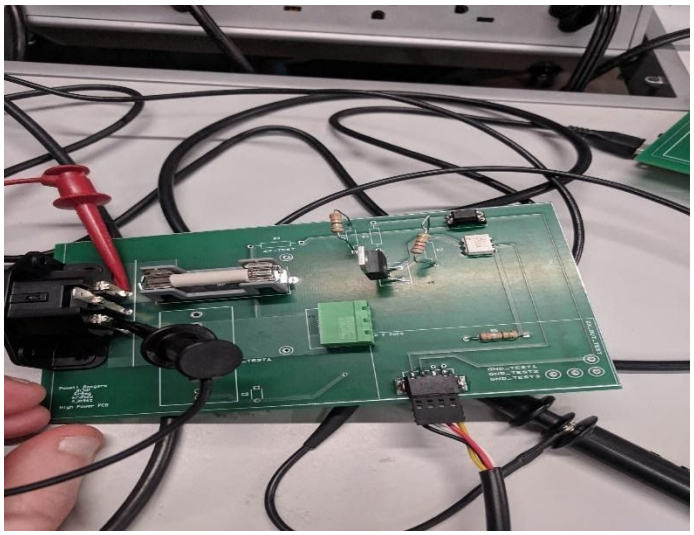


Figure 51: Set Up for Zero Crossing Testing on the HPB

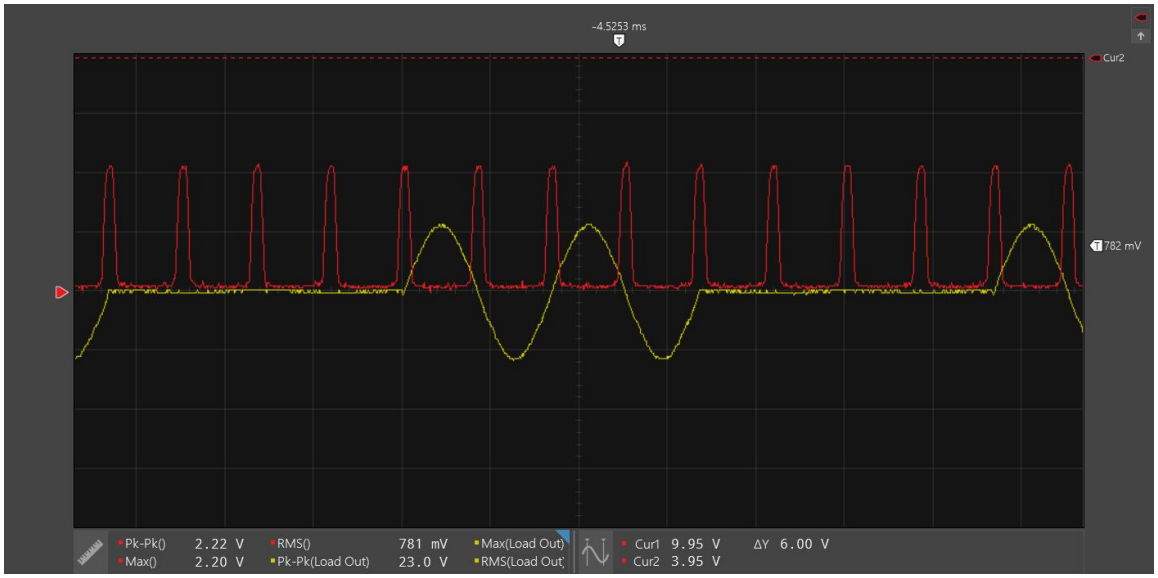


Figure 52: Verification of Zero Crossing on the HPB

High Voltage Testing on HPB:

- Having assured ourselves that our HPB was working properly at low voltages, we swapped out the current-limiting resistors for ones that would work for 120VAC, these were our measured values:
- R1 = 353.7
- R2 = 325.15
- R3 = 13.967k
- R4 = 13.940k
- All of these resistors are rated for at least 0.5W and at least 200V.

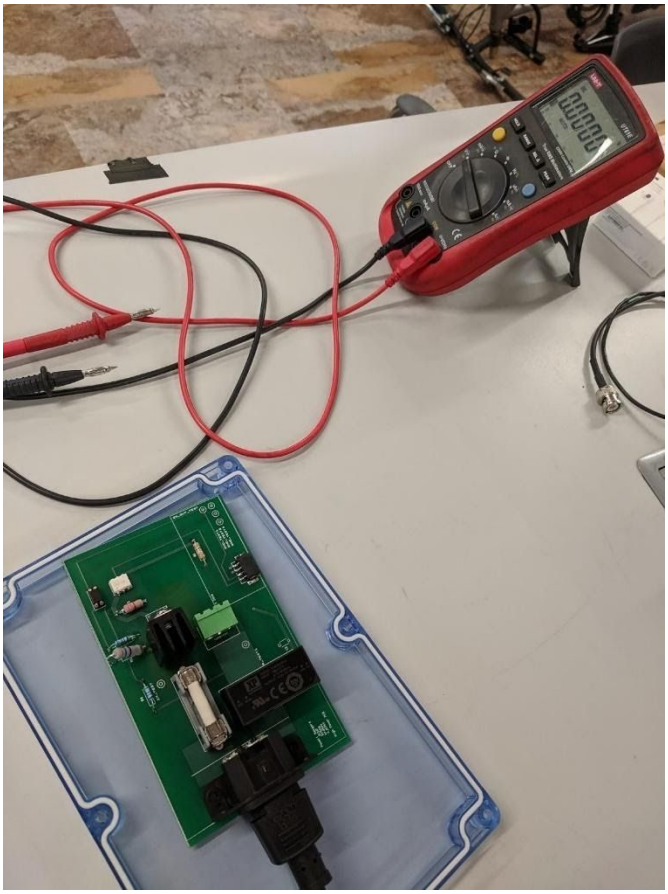


Figure 53: Testing HPB with AC Mains

- With the guidance of our TA, we carefully placed the board on the lid of its enclosure and stayed a safe distance away as we plugged it in. After being plugged in for 30 seconds, we saw no melting pieces and smelled nothing out of the ordinary so we assumed things were safe to test with a multimeter.
- We then connected the 120VAC - 3.3VDC voltage regulator and saw that it was making 3.27VDC on its terminal end, very close to the projected 3.3VDC.
- To make sure this was enough to power our board, we connected our cable from the HPB to the LPB and saw the Launchpad LEDs turn on.

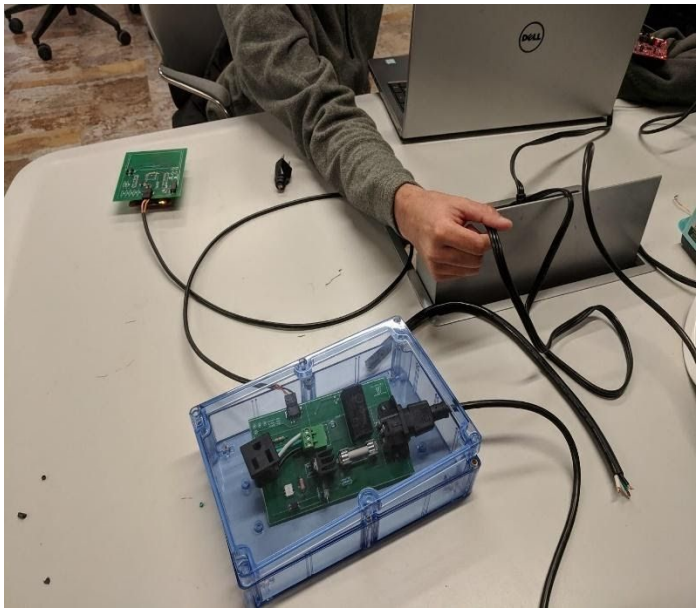


Figure 54: CC3220 Receiving 3.3V from the Voltage Regulator

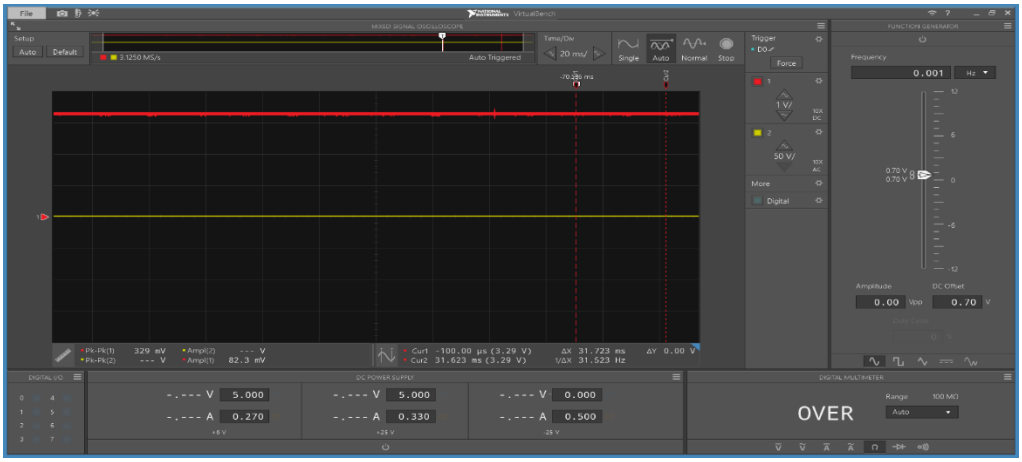


Figure 55: Verification of CC3220 Receiving 3.3V

- We then used a 300V rated oscilloscope to test the voltage across the pins of the triac to be sure that the voltage was reaching as far as it was earlier and to be sure that the pins of the triac weren't too close to each other, causing arcing. We saw about 180Vpk, signifying to us that the separation was wide enough.

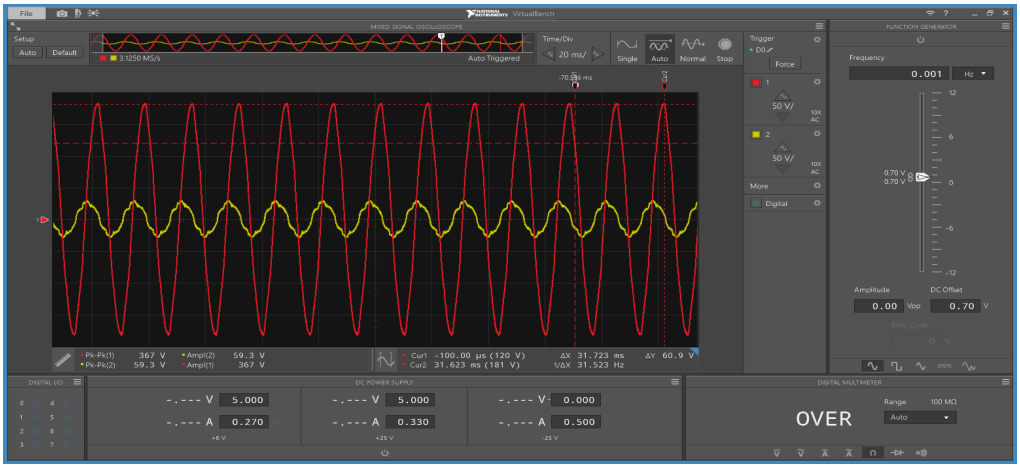


Figure 56: Voltage Across Triac A1 and A2 using AC Mains

- For our next test, we connected the lamp to the VAC out plug and saw that the lamp turned on. This was proof that our board could at least propagate the 120VAC through all of the components, send that to a 50W load and take the current back through to neutral, all without melting.
- Next, we tested sending a PWM signal to the lamp via the LPB - HPB connection and saw that the lamp turned on and off at the same frequency as our LED that we were using to signify the PWM. This was proof that we could switch the lamp fast enough to meet our 15Hz goal.

Figure 57: Testing LPB and HPB Using Mains with a 50W Bulb

Temperature Sensor Testing:

- We decided to not go home for thanksgiving break and, thus, had a ton of time to work on this part. The sensor testing went largely according to plan.
- We began by connecting the wires using a pair of tweezers to open up the wire opening, sticking the wire in, then removing the tweezers. This was enough to get the wires in, but we had to adjust them several times to get reliable results.

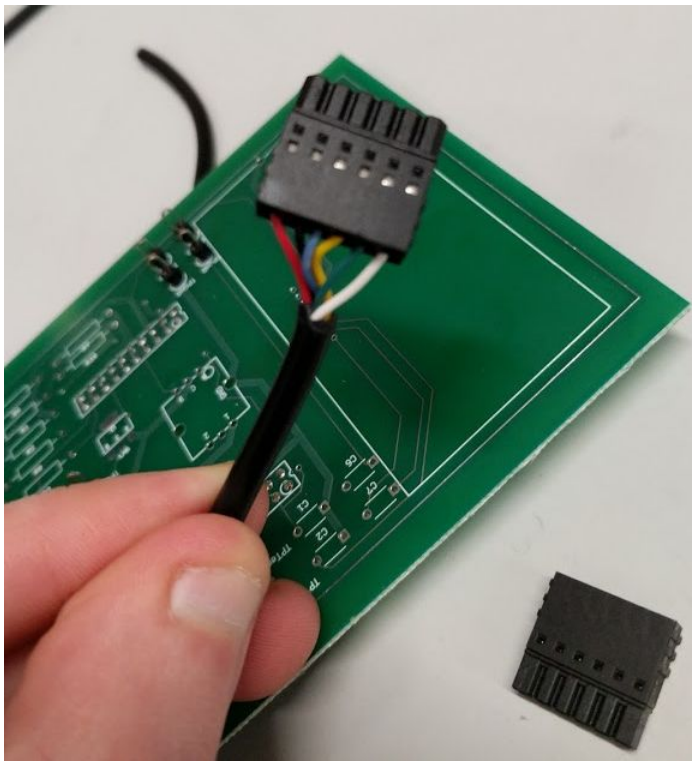


Figure 58: Temperature Sensor Cable Single End

- After soldering the surface-mounted connector, surface-mounted temp sensor and capacitor to the TB, we hooked up the wires, connected the logic analyzer, and tried to send a basic string of bytes to the sensor over MOSI.
- We spent a while fidgeting with this, but eventually realized that our MOSI and MISO had been backwards on one side of the cable, so we swapped those.

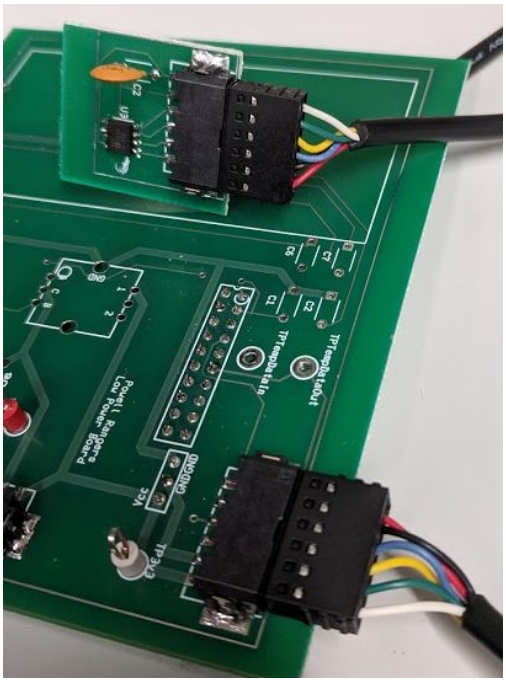


Figure 59: Temperature Sensor Cable with Both Sides Attached

- Now that the cables were connected in the right orientation, we thought everything would work, but alas it did not.
- Our temperature sensor was not connected properly, so we had to resolder several of its pins.
- Now we thought the connection would work, but still no luck.
- We realized that the protocol was not acting correctly so we had to change our CS to be software controlled.
- Finally, we got values back but they were negative. We thought this was strange, but when we unplugged the logic analyzer they went to realistic values like 23.
- We later realized that the cable was poorly connected and readjusting it fixed the possibility of getting these faulty negative values.

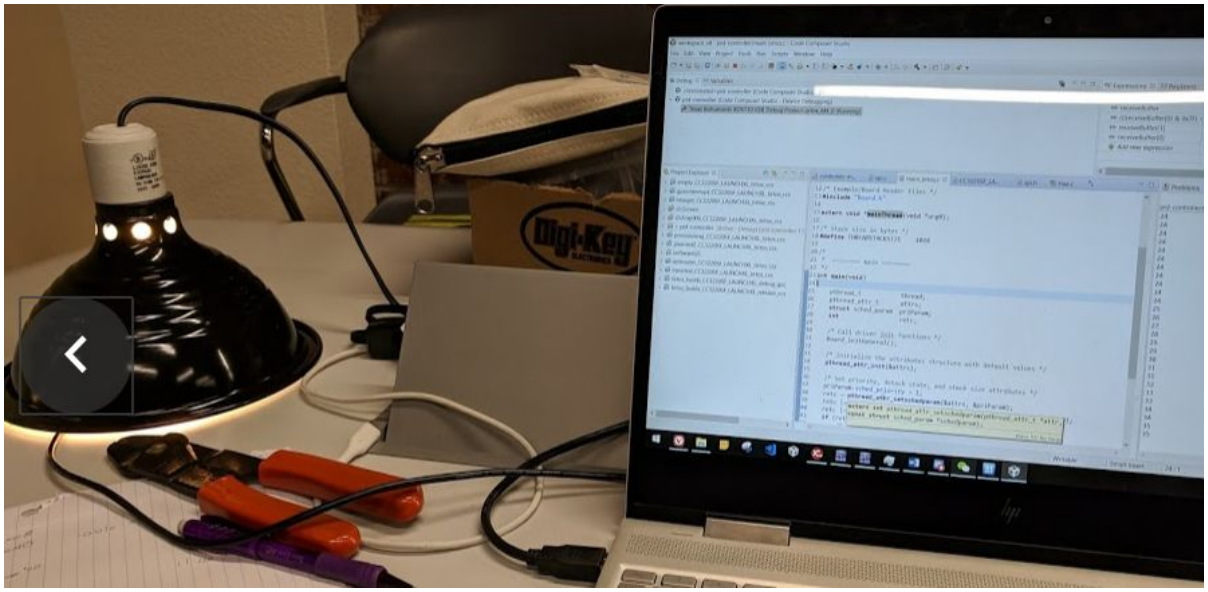


Figure 60: Temperature Sensor Working Properly

Rotary Encoder Testing:

- We began by connecting the necessary resistors and capacitors for the rotary encoder on a breadboard, as seen below

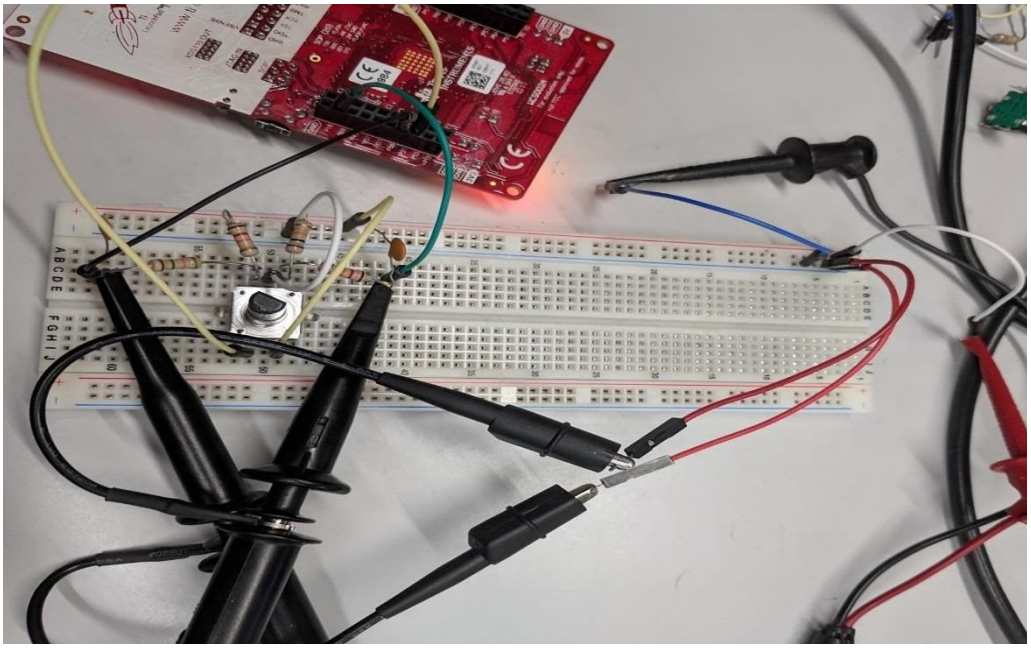


Figure 61: Rotary Encoder Set Up

- We ended up using resistors in the 10kOhm - 15kOhm range as we had no 10kOhm resistors left and saw that these values weren't extremely important and had a very small amount of time left to get the circuit up and running.
- R1 = 15.014k Ω
- R7 = 12.060k Ω
- R8 = 9.954k Ω

- R9 = 12.020k Ω
- These were the same resistors that we ultimately used on our PCB.
- As the rotary encoder was an input to the software rather than an output from it, we connected the oscilloscopes as seen above and turned the encoder to be sure that the signals we saw lined up with our expectations of offset square waves.

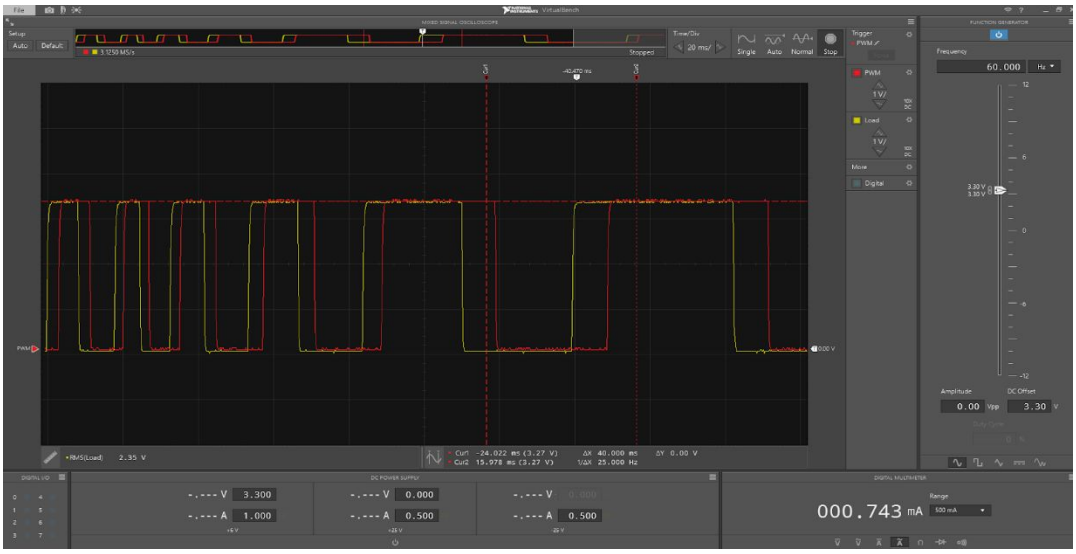


Figure 62: Verification of Functionality of the Rotary Encoder

- Next, we set up individual interrupts for terminals A and B and saw that the interrupts were both triggered. Finally, we altered the code to modify a global variable based upon the clockwise or counterclockwise turning of the encoder.
- We then moved our parts to the PCB. We didn't have much time left, so we soldered all of the parts at once and hoped that the connections would work out.

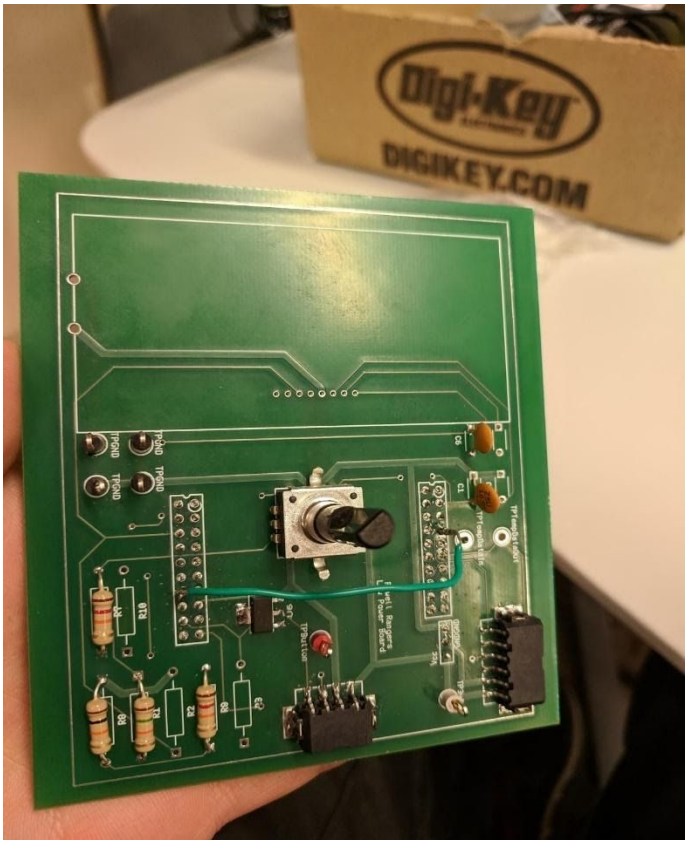


Figure 63: LPB After Rotary Encoder Assembly

- After soldering everything on, we tested it with the code and everything worked as expected.

LCD Screen Testing

- Our first attempt at hooking up the LCD screen involved soldering all of the pins to jumper wires and plugging these into a breadboard.


Figure 64: LCD Screen with Jumper Cables Set Up

- The problem with our first approaches were that we didn't realize that all of the pins were connected backwards, so none of the software signals that we sent were being interpreted and we were likely damaging the screen.
- Luckily, we had another screen that we had ordered a while prior specifically with this dire scenario in mind.
- Once we figured out that our pins were all backwards, it was already afternoon on the day before our presentation, as such, we didn't even have time to get the screen working on a breadboard before moving it to the PCB. We immediately cut all of the traces related to the screen, soldered the screen in, and began soldering jumper wires to connect the pins to the correct locations.
- It turns out that we had cut the trace that sent power to the backlight so we had to add an unnecessary jumper wire to make that work again.
- We ran into an issue where p03 and p04 would not send I2C signals, so we switched them to p01 and p02. The problem here was that p01 was being used for our PWM to the MOSFET, so we switched that to p64.
- At this point we miraculously had data being sent to the screen that we could see via the virtual bench logic analyzer, but nothing showed up on the screen.
- We then realized that we were using a ceramic capacitor between pins 07 and 08 instead of an electrolytic one, when we switched that, our screen popped up our "hello world" text.

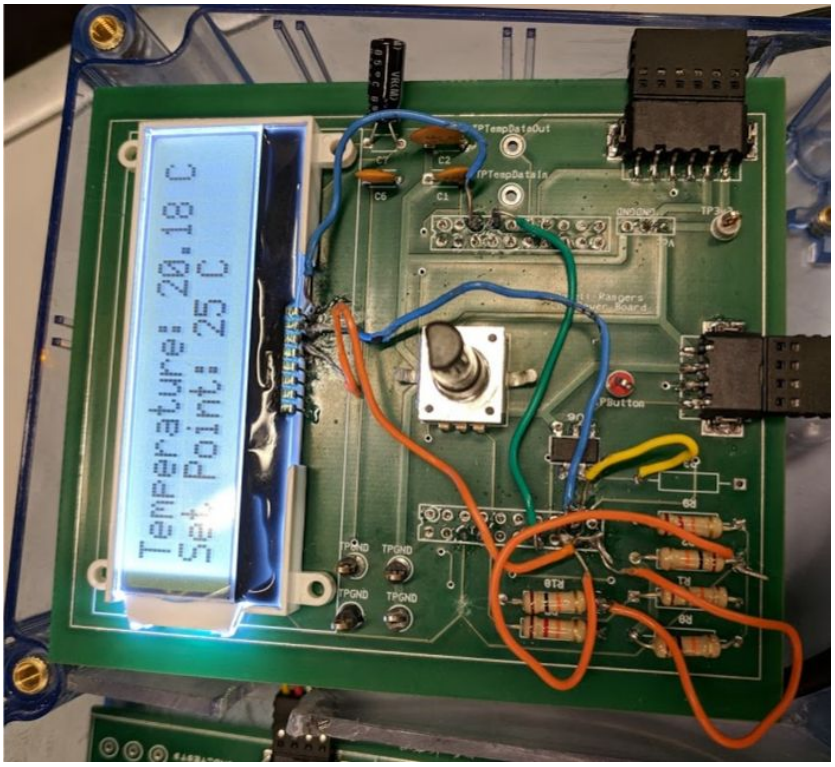


Figure 65: Functioning LCD Screen

- The testing for this screen was not done the right way and ideally we would have figured out that the pins were backwards before sending out our final board but, sadly, we live in the real world. We are just satisfied that it works at all.

SPI Command Byte Layout

Table 15. Command Byte

C7	C6	C5	C4	C3	C2	C1	C0
0	R/$\overline{W}$	Register address			Continuous read	0	0

Figure 66: ADT7310 Command Byte Layout

Table 6. ADT7310 Registers

Register Address	Description	Power-On Default
0x00	Status	0x80
0x01	Configuration	0x00
0x02	Temperature value	0x0000
0x03	ID	0xCX
0x04	T_{CRIT} setpoint	0x4980 (147°C)
0x05	T_{HYST} setpoint	0x05 (5°C)
0x06	T_{HIGH} setpoint	0x2000 (64°C)
0x07	T_{LOW} setpoint	0x0500 (10°C)

Figure 67: ADT7310 Registers

For further detail, see the data sheet for the ADT7310 SPI Temperature Sensor.