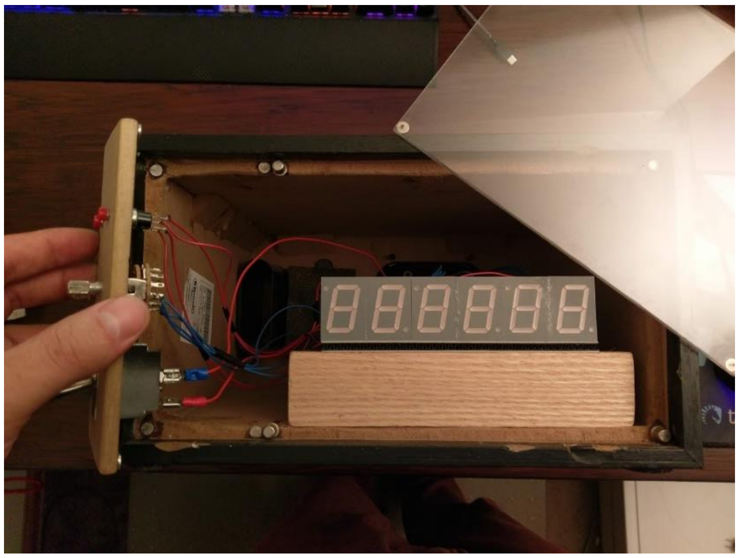
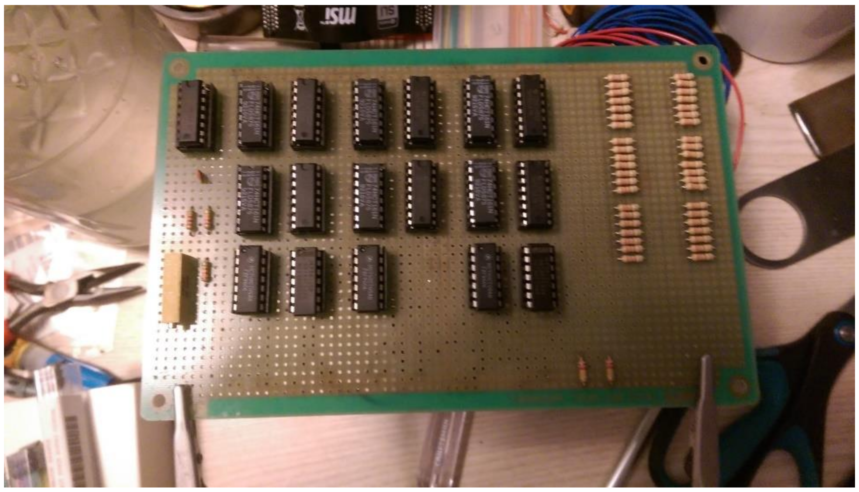
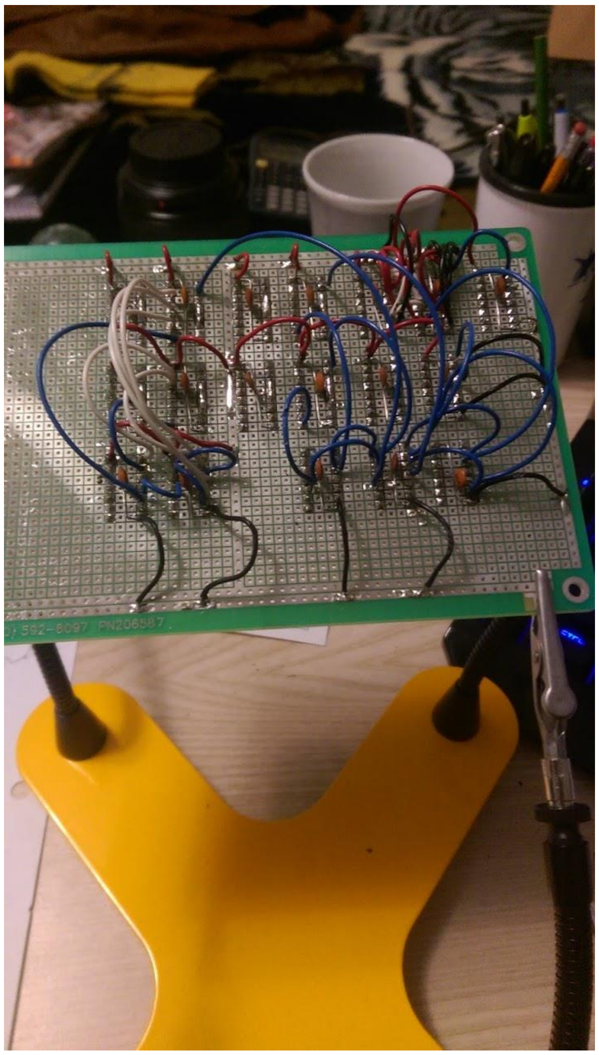
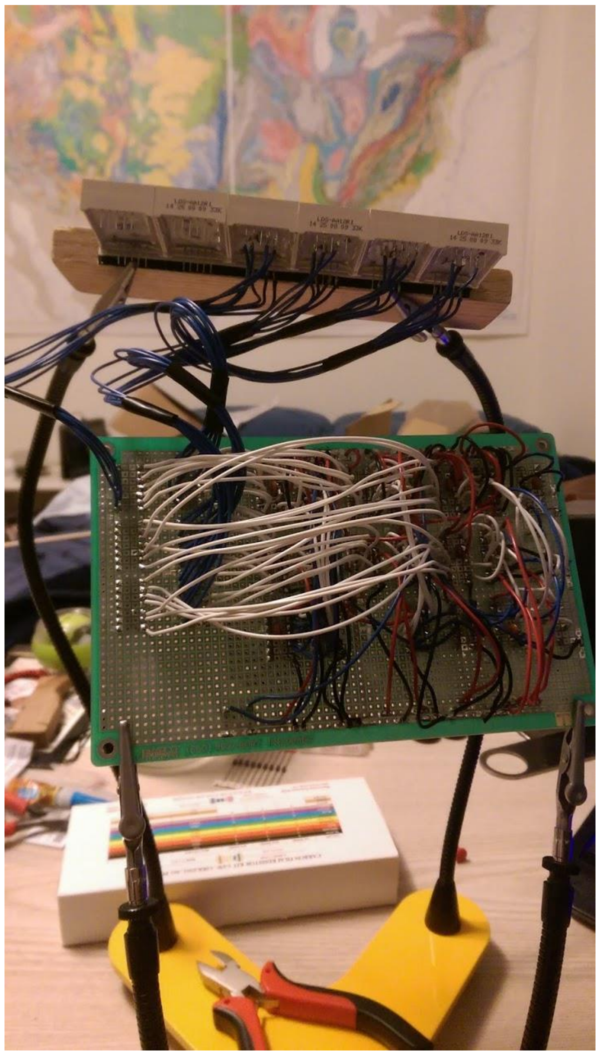
Freshman Projects Clock

This clock was made for a freshman projects class as an introduction to electrical engineering. It was implemented with a resonator, tuned by a trim pot, feeding an array of 74-series NAND/NOR logic that in turn fed decoders to drive (sink, really) the 7-segment displays.

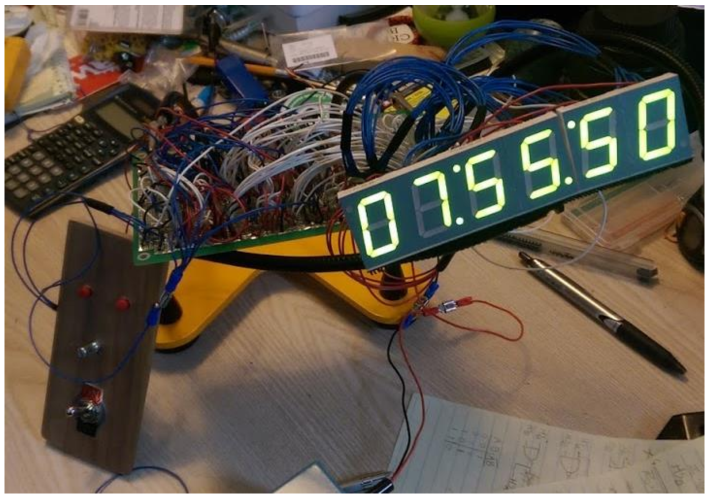
This was all placed into a hollowed-out speaker enclosure with a magnet-snap screen and control panel to adjust the clock.


Finished product – Front screen and control panel


Top of the PTH Board after assembly

Humble soldering beginnings on the left, miracle on the right



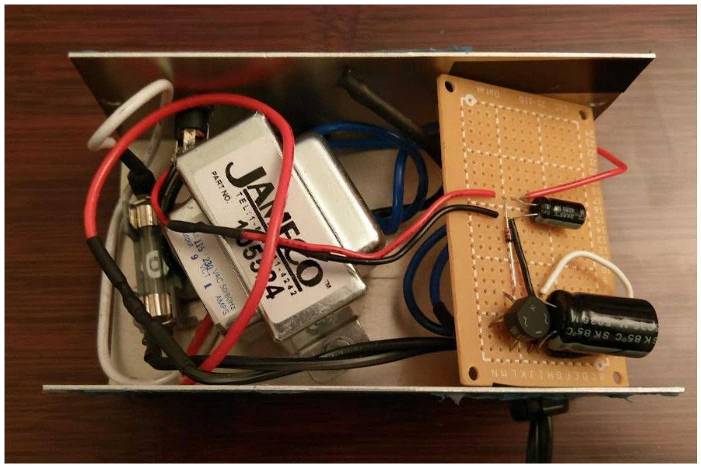
Fully functional, pre enclosure.

A few things learned as this was so long ago:

- Aside from being a good exercise, this method of building a clock is expensive, nearly impossible to maintain, upgrade or debug, takes a long time to manufacture, and takes up excessive space
- Any basic microcontroller could implement this in an afternoon
- The smallest CPLD on the market could directly replace this and even implement the same gate-level interactions if required
- Layout is important but at some point you have to admit that you don't have enough space
- 74-series logic is still very satisfying to work with.

5V$_{DC}$ Power Supply

This power supply was made to supply the power to my clock project. It is a transformer-based 5V power supply that plugs directly into the wall. It only makes 5VDC but it does it very well with nearly no ripple under load.

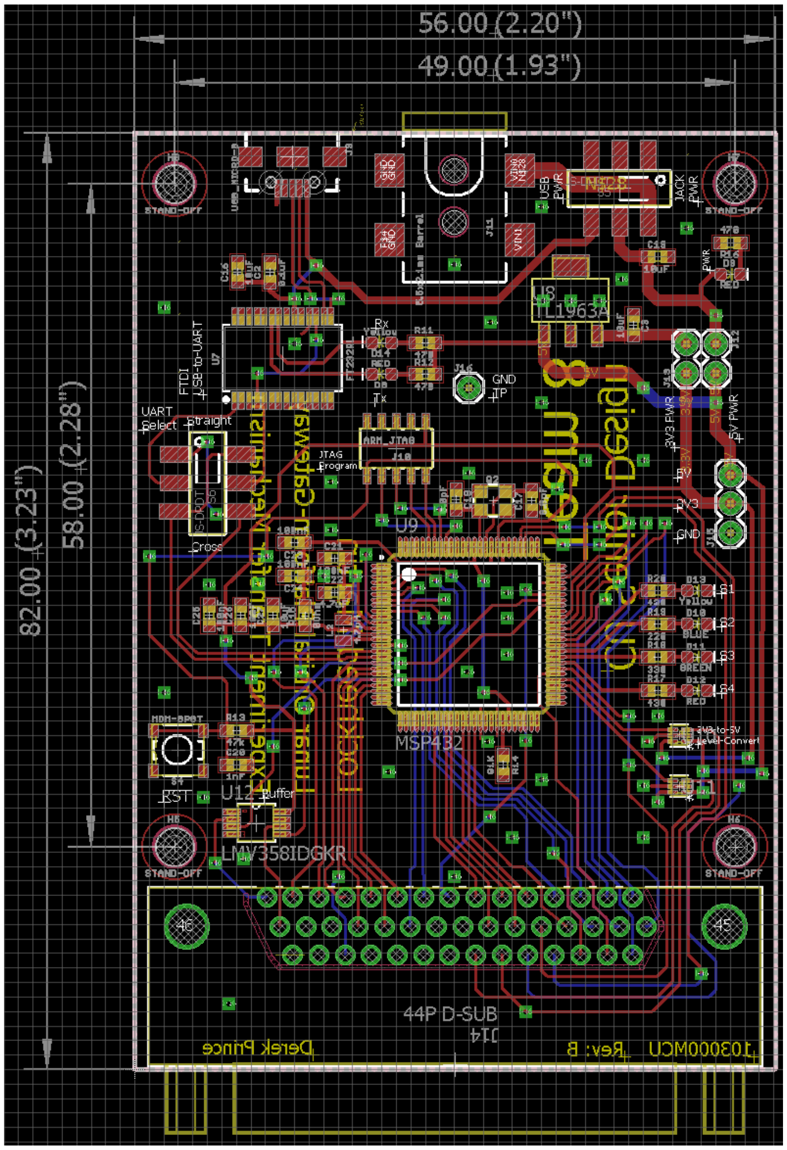


Closed box (top), internals (left)

Senior Design MSP432 Microcontroller Board

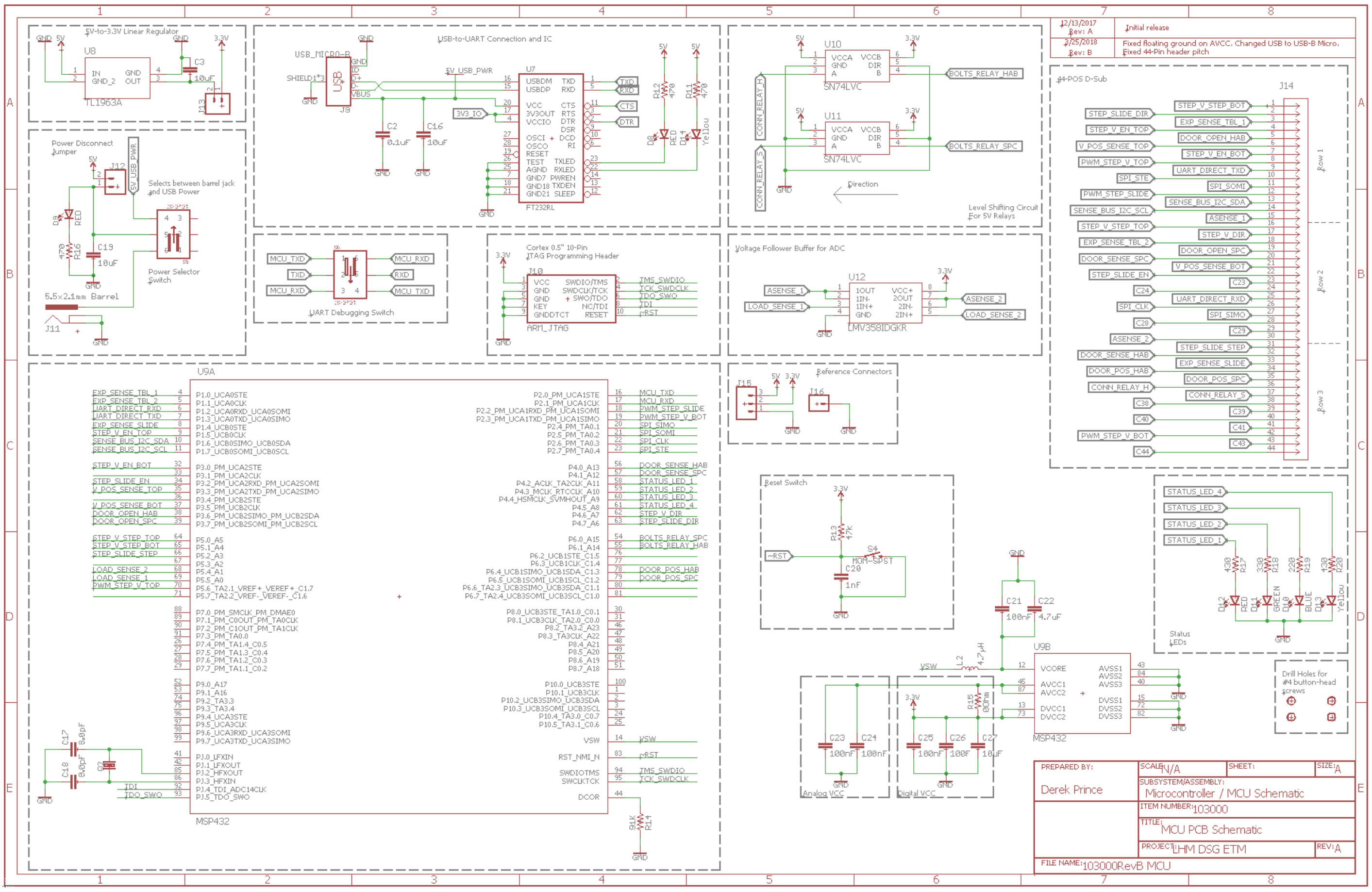
This PCB was designed as the main interface to all the sensors and drivers for my senior design project with Lockheed Martin. The board was designed to support a MSP432 microcontroller (MCU) with a footprint compatible with the mounting posts on a Raspberry Pi and support UART over USB to accept control and telemetry request commands.

The largest problem that arose was the two-layer restriction on the board that arose from cost and design tools available. This had a strong effect on the signal integrity and prevalence of inductive loops and broken return paths on the MCU routing as can be seen below. The board did feature two ground planes, they just make the design hard to digest so they have been suppressed for the image.

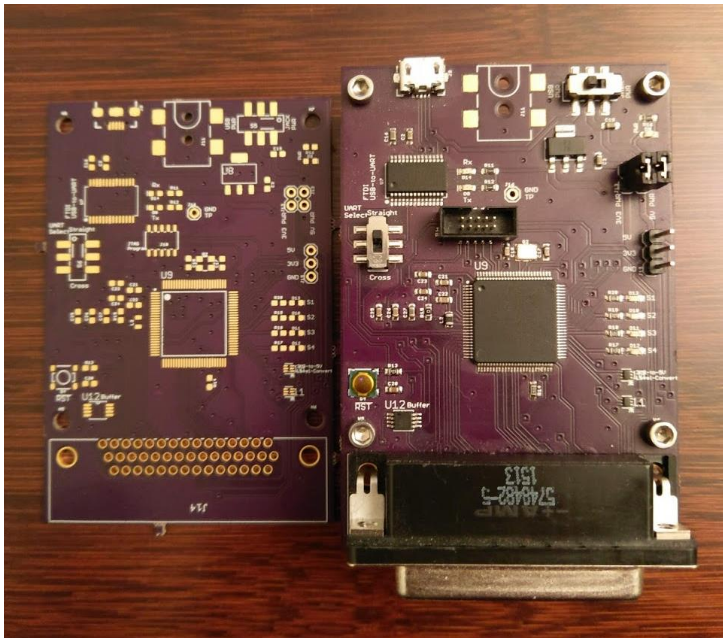


Board schematic for the microcontroller board

Here is the accompanying schematic, better viewed alone and out of a document:


5V-to-3.3V Linear Regulator
TL1963A
USB-to-UART Connection and IC
USB_MICRO-B
FT232RL
SN74LVC
Level Shifting Circuit For 5V Relays
Direction
44-POS D-Sub
J14
Row 1
Row 2
Row 3
Power Disconnect Jumper
Selects between barrel jack and USB Power
Power Selector Switch
5.5x2.1mm Barrel
UART Debugging Switch
Cortex 0.5" 10-Pin JTAG Programming Header
ARM_JTAG
Voltage Follower Buffer for ADC
LMV358IDGKR
Reference Connectors
Reset Switch
MOM-SPST
Status LEDs
Drill Holes for #4 button-head screws
Analog VCC
Digital VCC
MSP432
12/13/2017 Rev: A Initial release
3/25/2018 Rev: B Fixed floating ground on AVCC. Changed USB to USB-B Micro. Fixed 44-Pin header pitch
PREPARED BY: Derek Prince
SCALE: N/A
SHEET:
SIZE: A
SUBSYSTEM/ASSEMBLY: Microcontroller / MCU Schematic
ITEM NUMBER: 103000
TITLE: MCU PCB Schematic
PROJECT: LHM DSG ETM
REV: A
FILE NAME: 103000RevB MCU

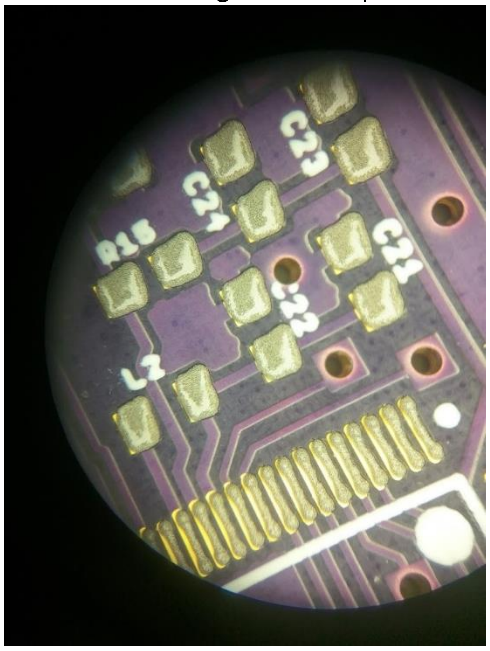
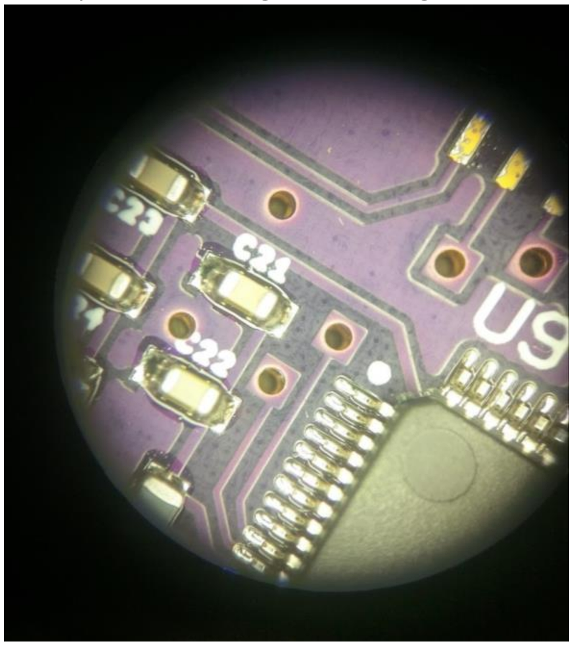
The resulting boards were manufactured by OSHPark and came out more or less beautiful.



Assembled board top next to raw PCB print top and bottom

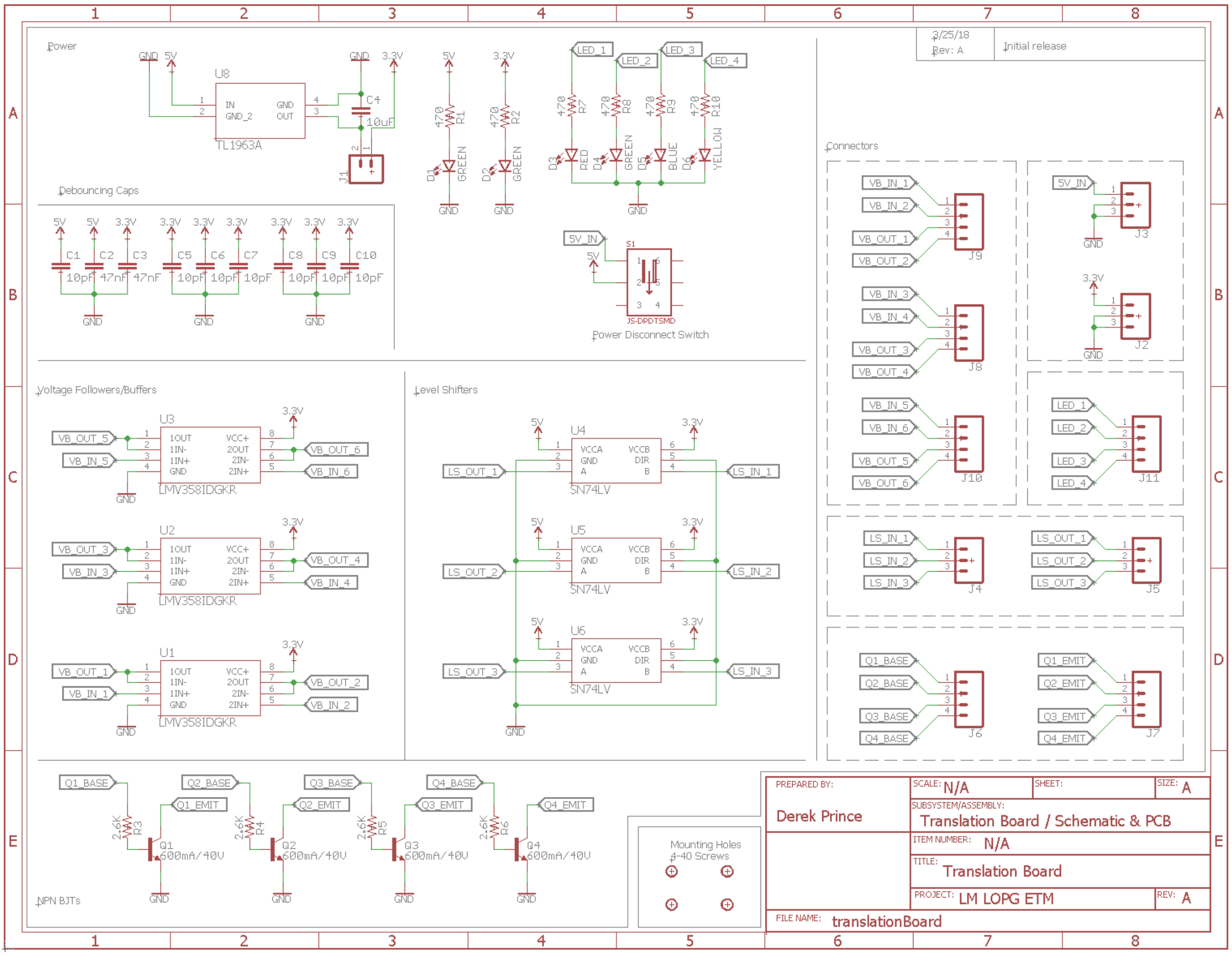
With the first run of the board, there were a few problems that arose, including an incorrect pitch for the 44P connector and insufficient clearance from decoupling capacitors on the 10-pin programming header. The microcontroller we received also had ever so slightly off-pitch pins that were only caught after reflow, causing the pins to wander onto the neighboring pads during reflow. This can be seen in the images below:

Left I am chcking the solder paste laid down by the Voltera, right is checking the reflow. The MCU pins can be seen wandering on the right side.

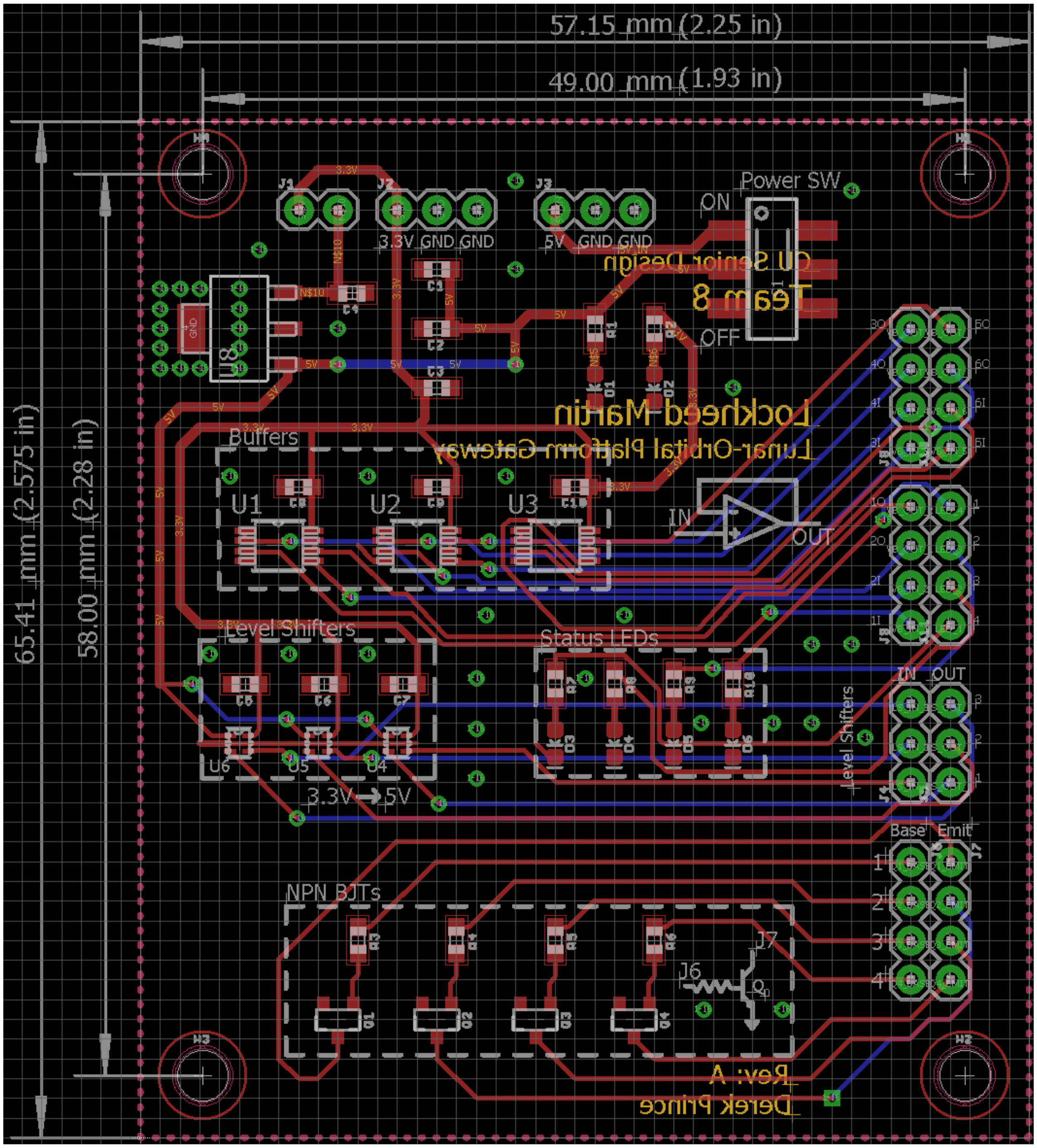



Development Conversion Board

This board was designed to convert the GPIO from the MSP432 TI Launchpad board to our special purpose board. The drive behind this board was to solve the problems of our design moving forward faster than the PCB fab shop could cheaply deliver so this allowed us to use the cheap (much cheaper than the custom board) dev board at home and in the lab to further development. There was also extra circuitry for unforeseen expansions like extra level converters and high-power switching drivers for relay control.



Conversion Board Schematic

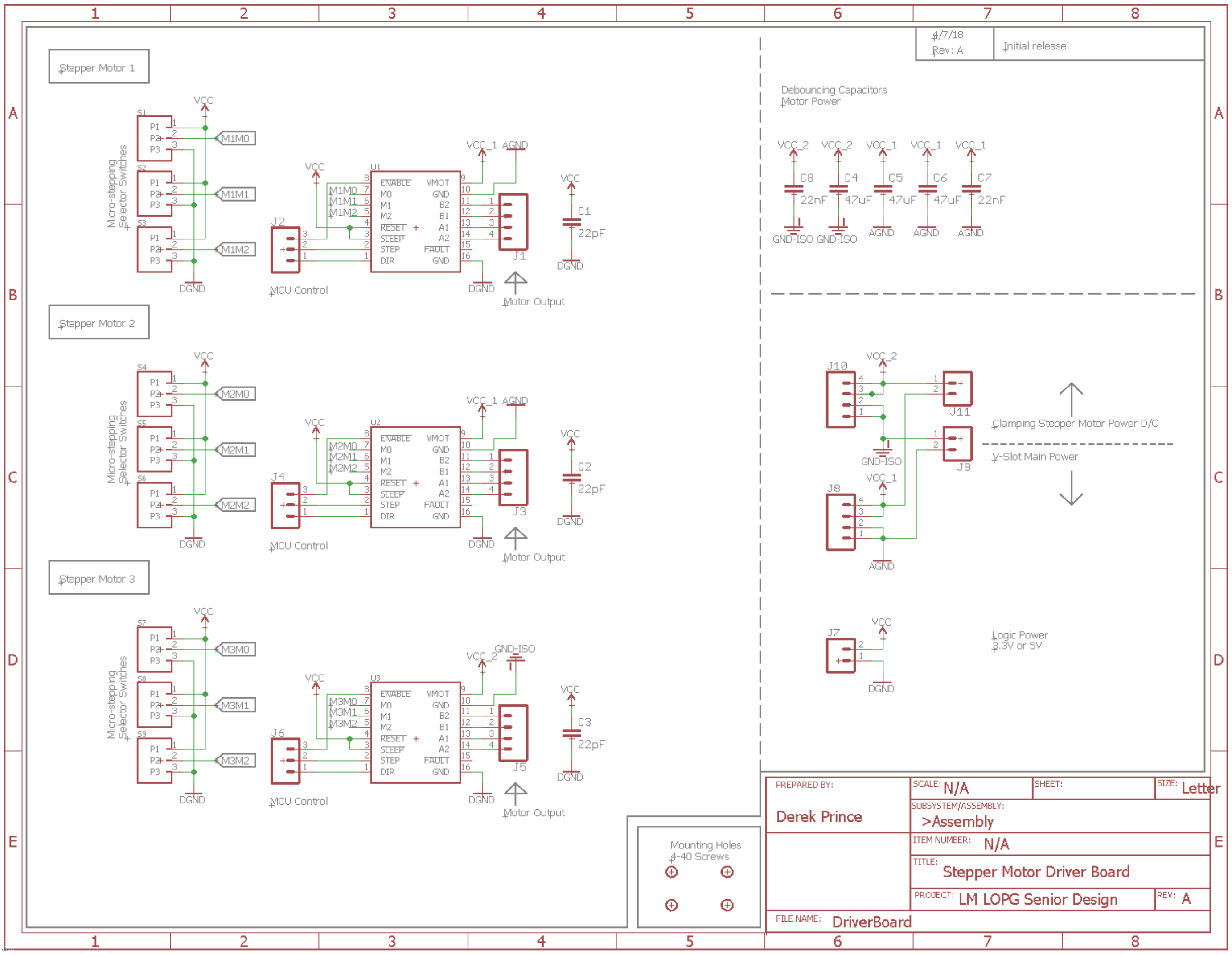


Board Routing Schematic. (ground planes not shown)

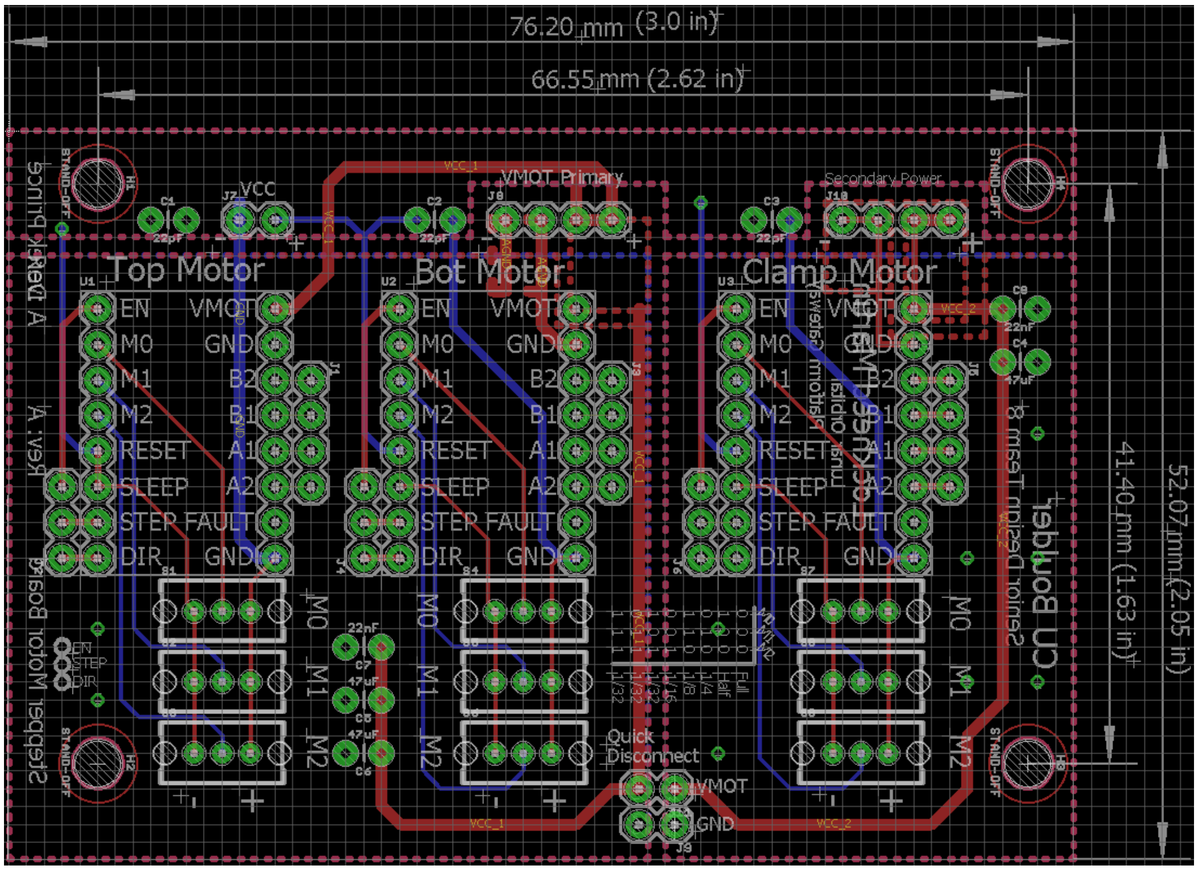
Somehow no known pictures exist of this board assembled. Likely because it was mounted underneath the main MCU board and never moved.

Stepper Motor Driver Socket Board

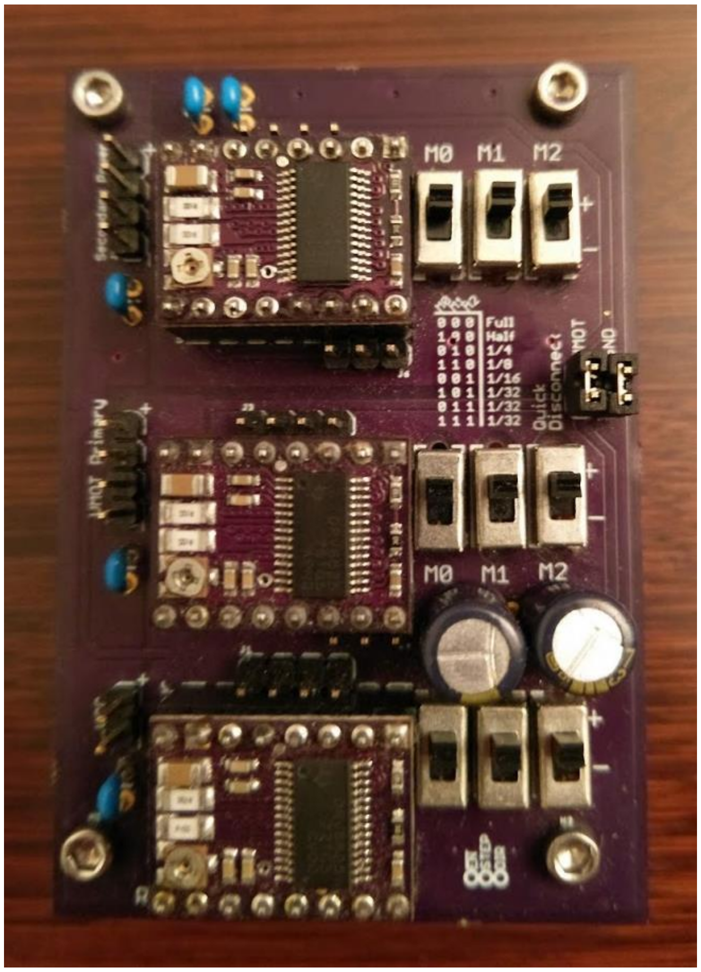
This board was designed to move the senior design team out of the breadboard-and-plated through hole phase while dramatically simplifying our testing setup. Designed for the popular Pololu high power stepper driver designs, it featured micro step selection switches and an option to isolate one driver to run it on a higher power level depending on what setup best fit our application load.

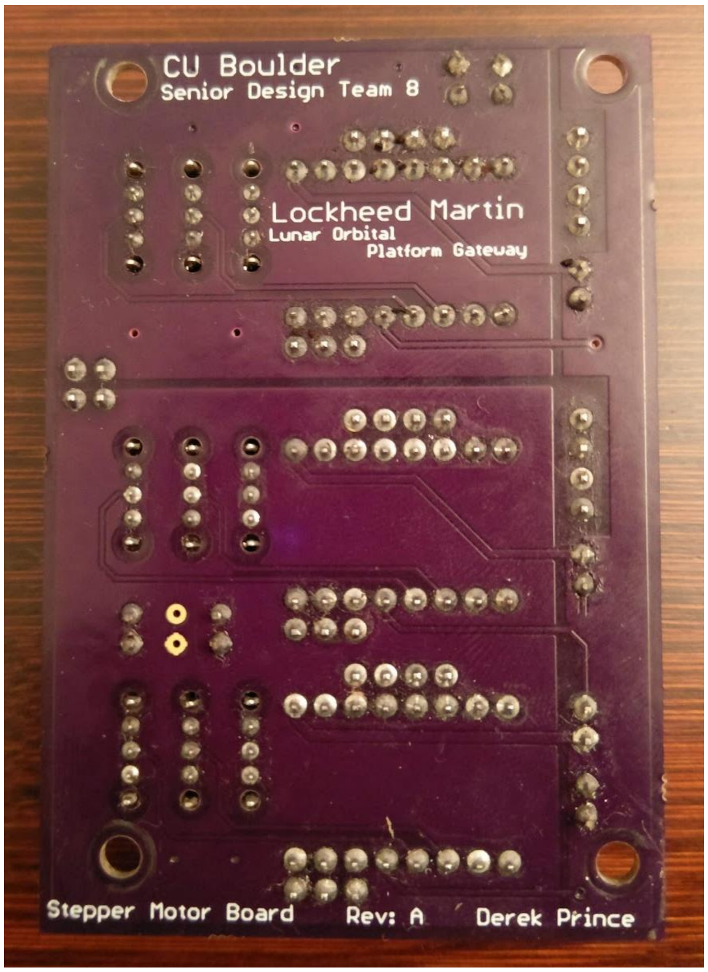

Stepper Motor 1
Stepper Motor 2
Stepper Motor 3
Micro-stepping Selector Switches
MCU Control
Motor Output
Debouncing Capacitors Motor Power
Clamping Stepper Motor Power D/C
V-Slot Main Power
Logic Power 3.3V or 5V
Mounting Holes 4-40 Screws
4/7/18 Rev: A
Initial release
PREPARED BY: Derek Prince
SCALE: N/A
SIZE: Letter
SUBSYSTEM/ASSEMBLY: >Assembly
ITEM NUMBER: N/A
TITLE: Stepper Motor Driver Board
PROJECT: LM LOPG Senior Design
REV: A
FILE NAME: DriverBoard

Stepper Motor Driver Socket Board Schematic



Stepper Motor Driver Socket Board Schematic

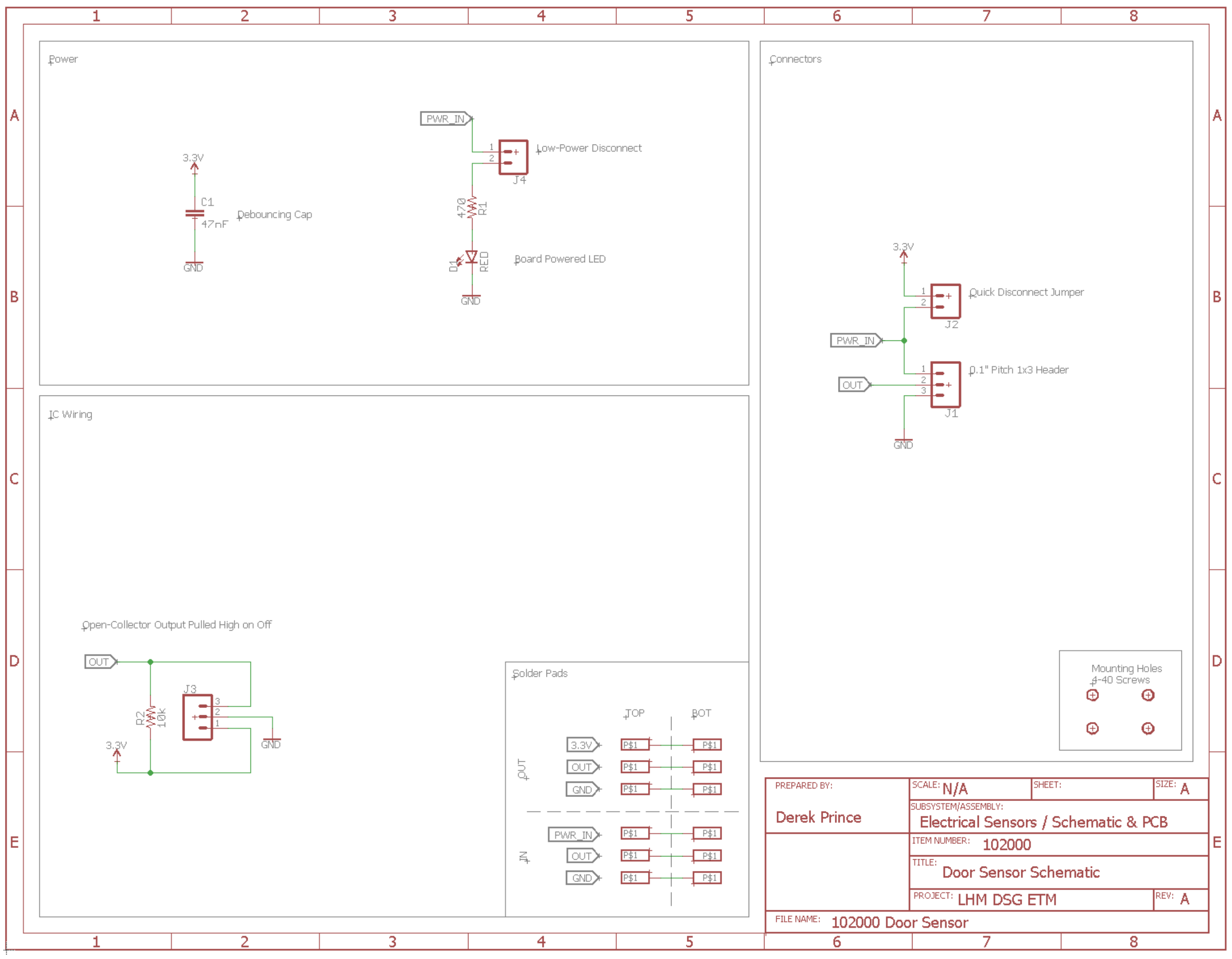




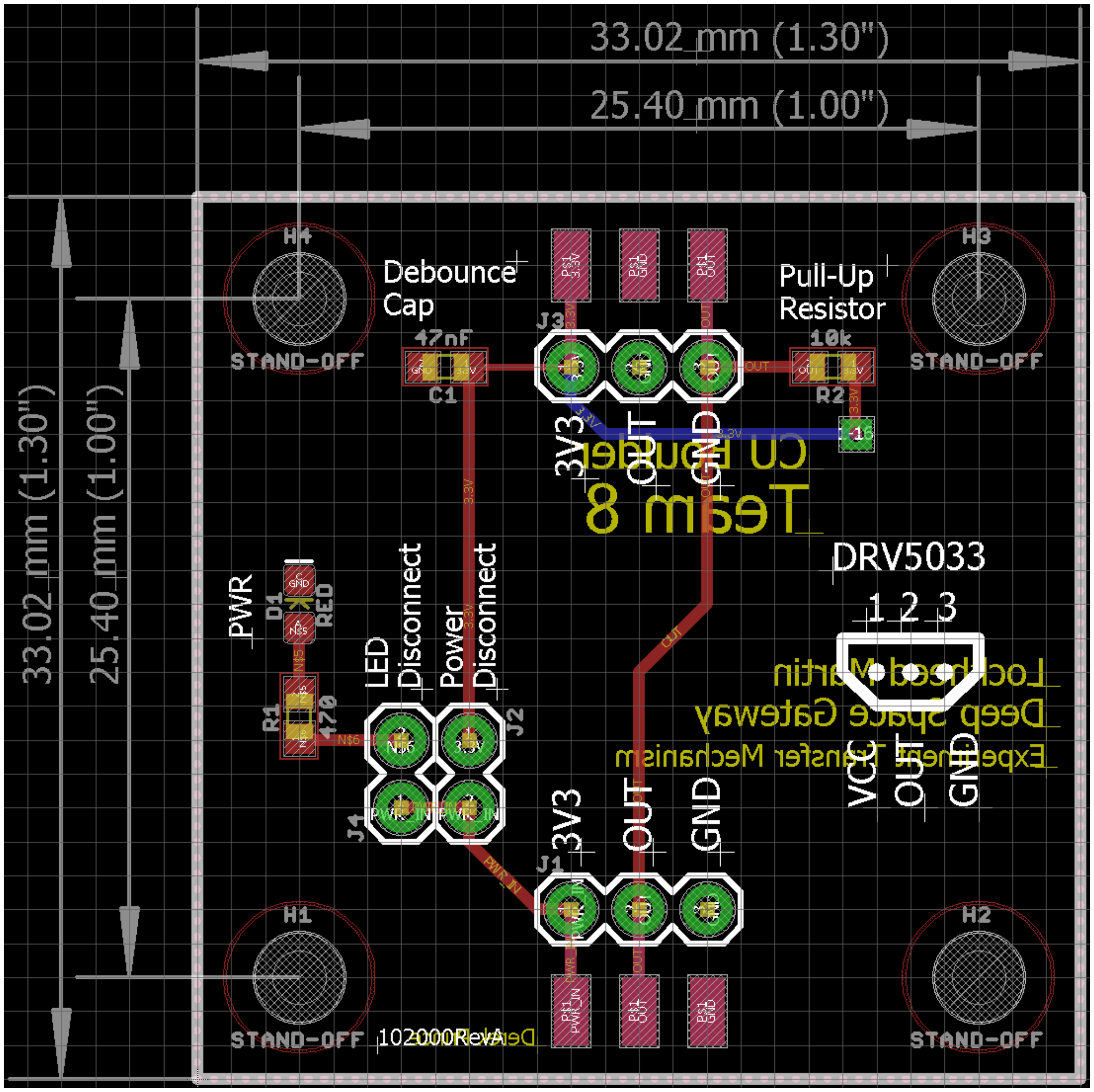
Assembled board top (left) with drivers installed and bottom (right)

Hall Effect Sensor Mount Board

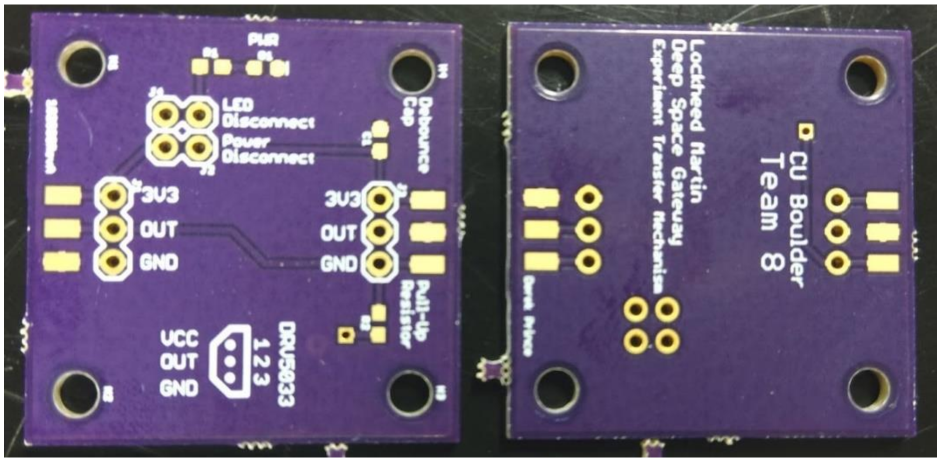
This board was designed to structurally support our various hall-effect boards. With the number we required, the cheap cost of components, and the minimum buy quantity required from OSHPark, this was an effective solution to mounting small components. They're simple boards with isolation jumpers, optional power indication LEDs, pull-up output resistors, and debouncing caps on a 1" squared mounting pattern.



Hall Effect Mount Schematic



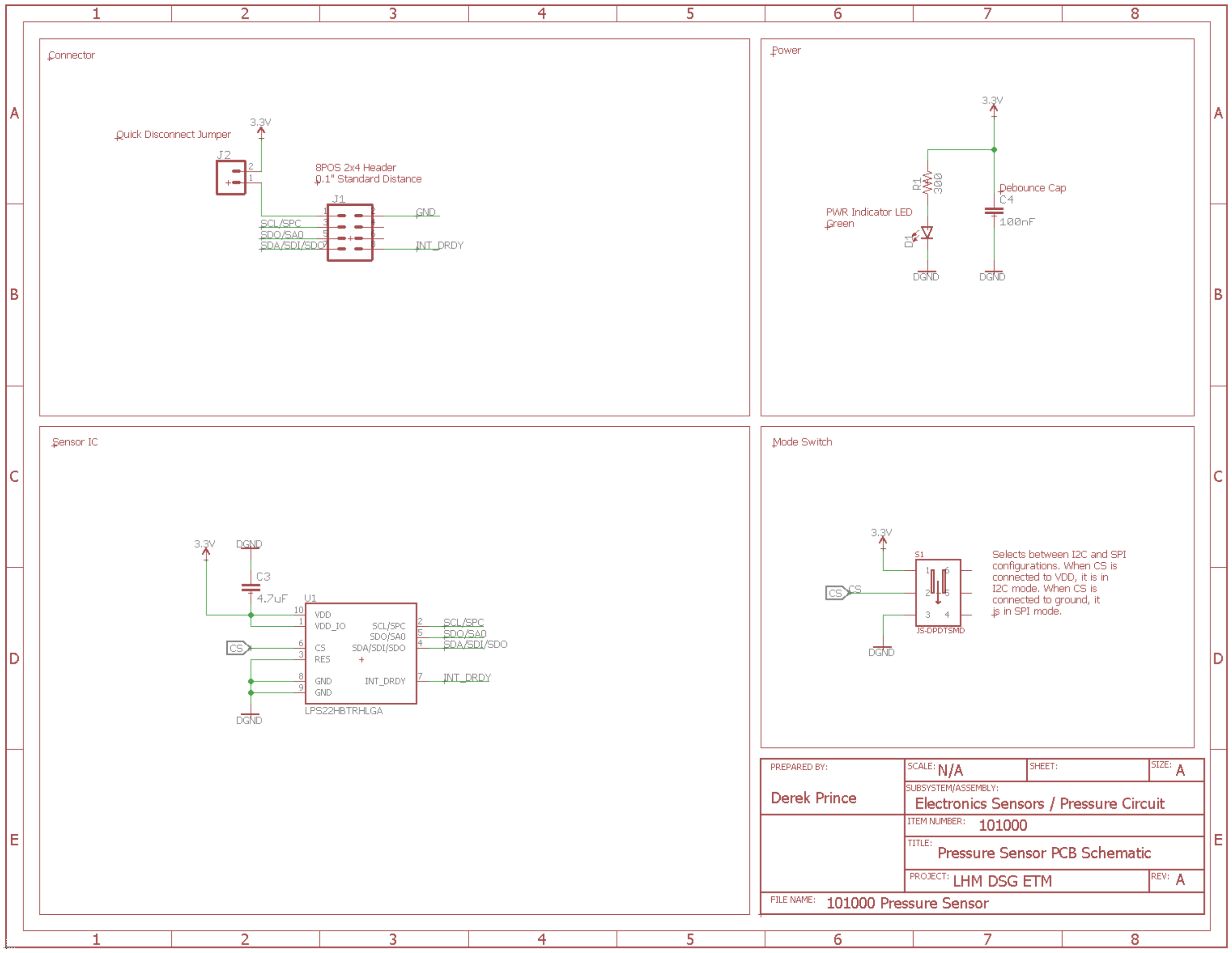
Hall Effect Board Layout



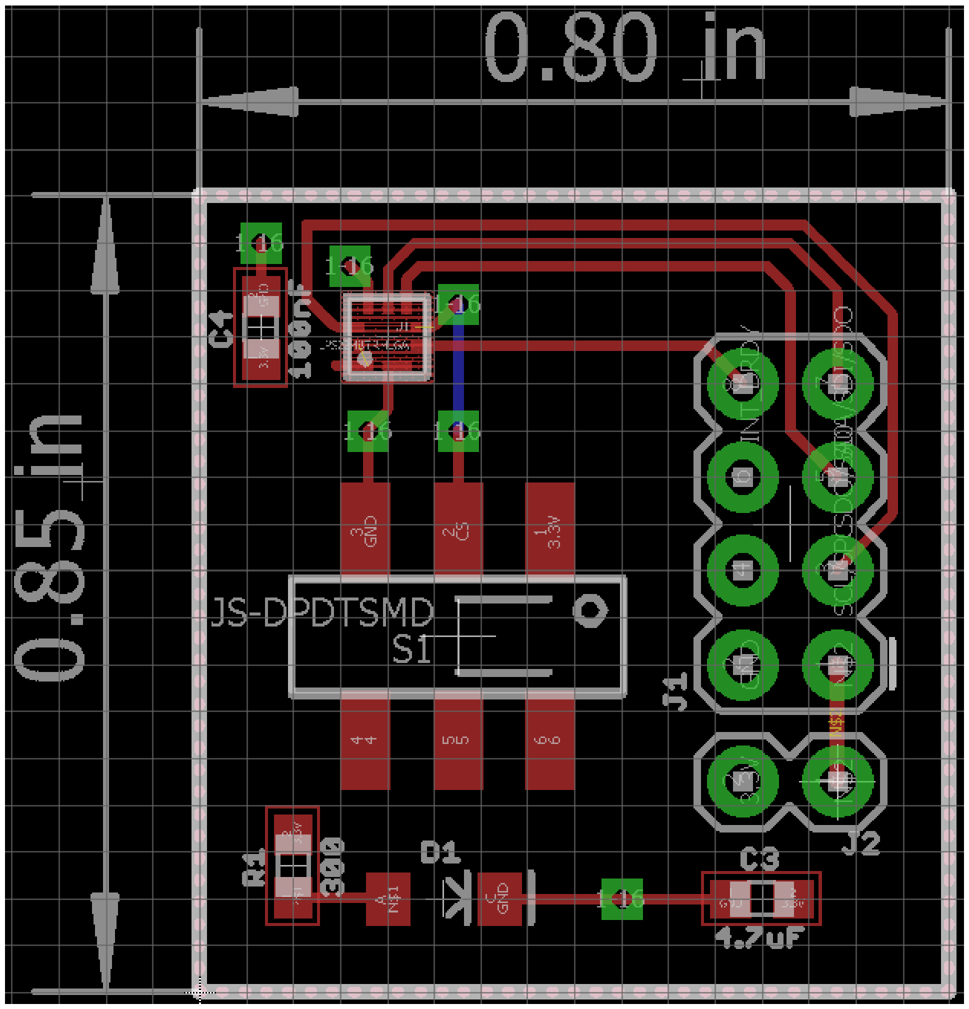
Pre-assembly and fresh from the fab house

Pressure Sensor Board

This board is very similar to the hall effect board where it’s main function was to provide good mounting and required passive components in a small space. Pressure regulation was dropped from our requirements so the board was never sent out to fab but it is pretty none-the-less. The board also features a SPI/I2C selector switch, which occupies about 30% of the board space.



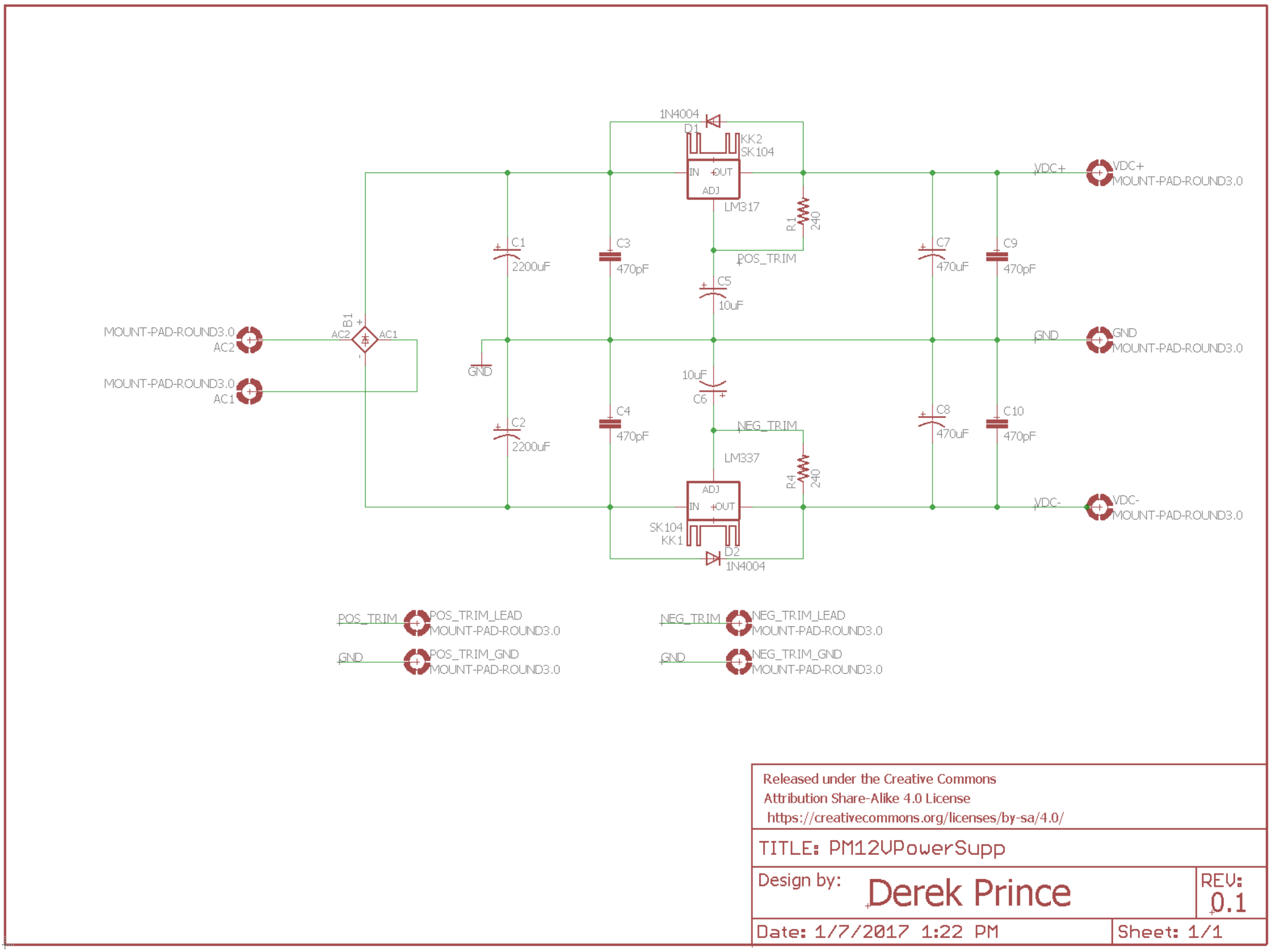
Pressure sensor schematic



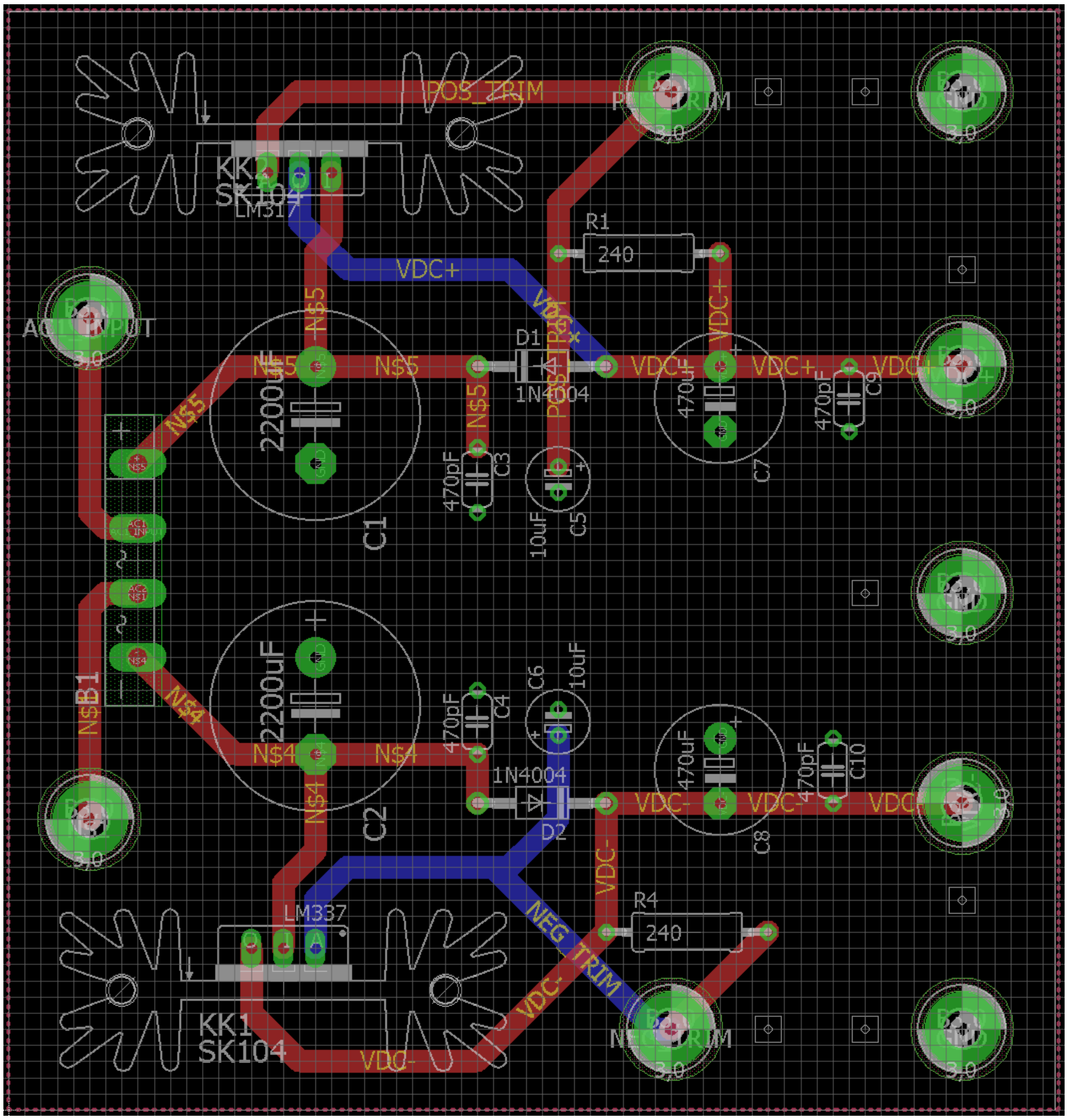
Pressure sensor board layout (ground planes suppressed)

Variable Power Supply

This supply was designed as a successor to the 5V TTL power supply I made in my freshman year but put on hold after pricing out the final design and seeing how the price compares to commercially available power supplies. I would still like to revisit this project and include a low power MCU to set limits and display outputs, while utilizing a fly-back design (or similar) over the expensive power transformer. The board was designed to have connectors placing fine and course trim pots in line to vary the output voltage. Design inspired by all relevant chapters in *The Art of Electronics Third Edition* and *Practical Electronics for Inventors*.



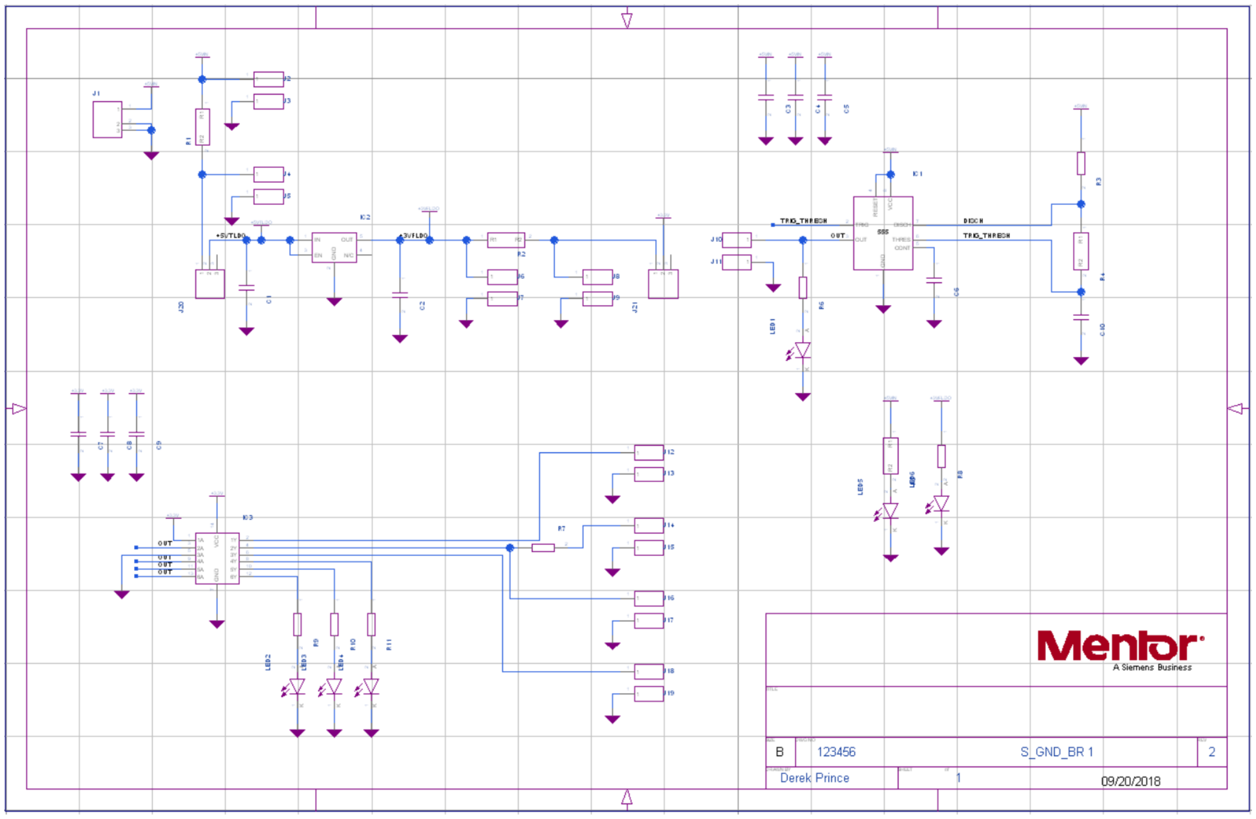
Variable Power Supply Schematic



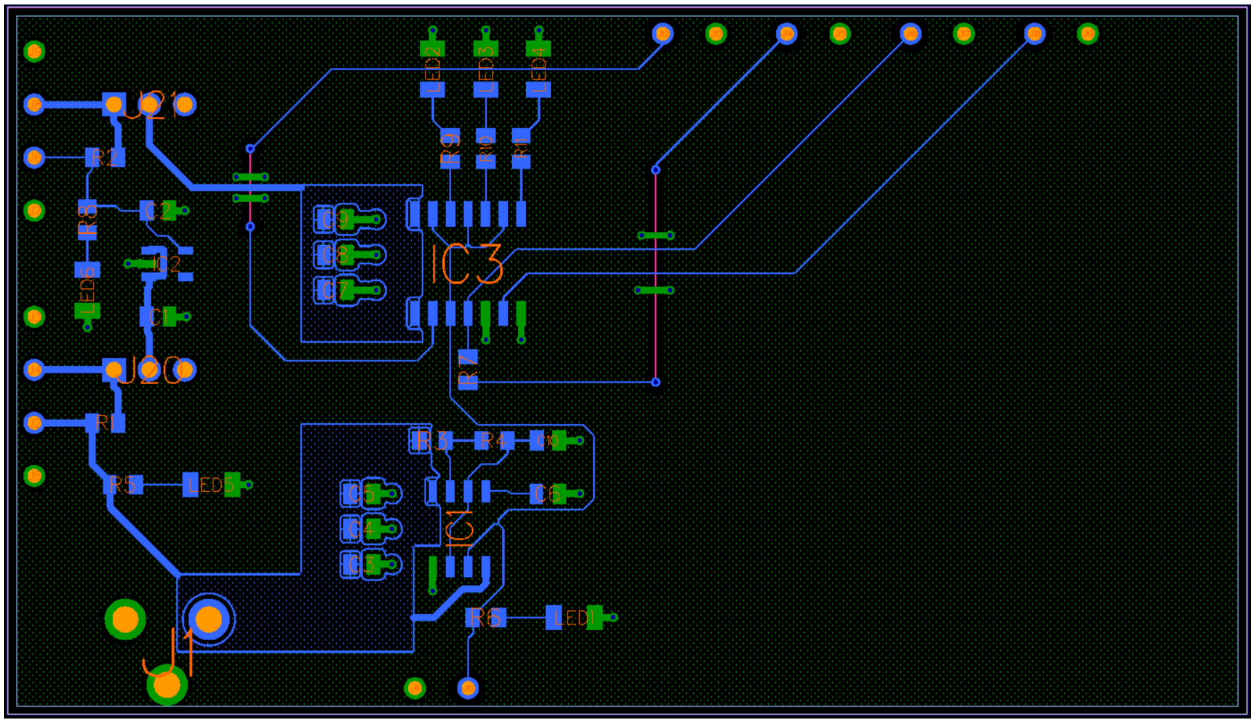
Variable power supply board layout

An Exercise in Ground Bounce – The 555 Timer PCB

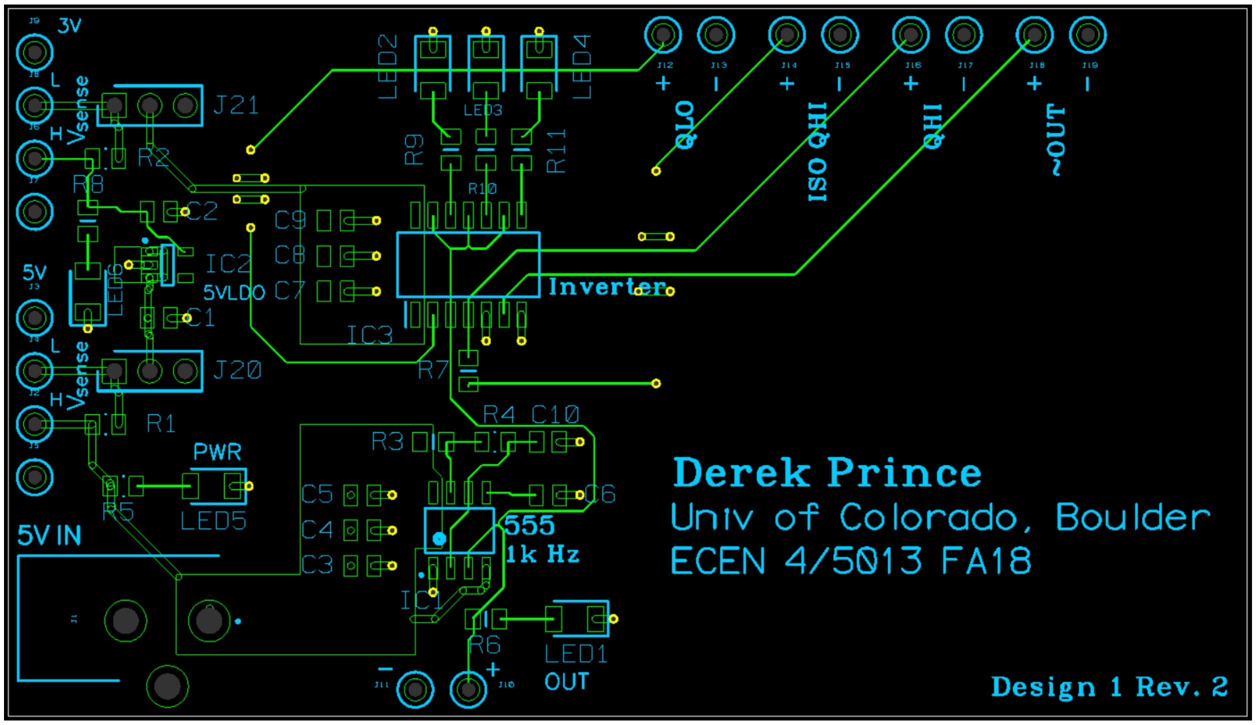
This board was the first assignment for a PCB design class. Initially we had to design a board with no ground planes with cross-routing to exacerbate the noise in the on-die ground from the timer and inverter switching, and then design a board with loop inductance in mind to compare the results in lab. As I had a partner for this lab, I do not have the physical copy of the good board I designed (though the design was entirely my own). This was also the first design in PADs Pro as opposed to Eagle.



Schematic of 555 Timer and Inverter Test Circuit

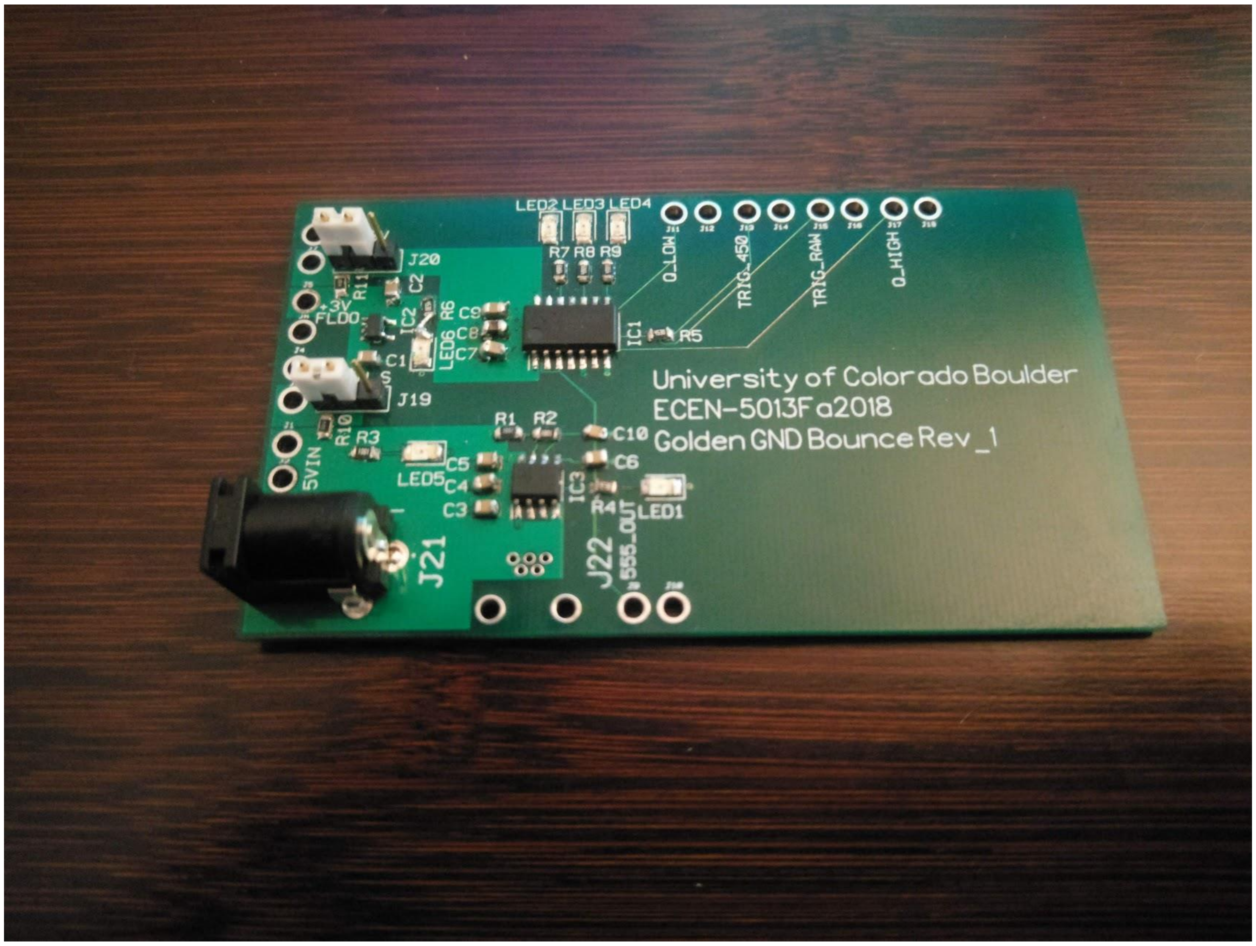


Board Routing – Hatched areas (green and blue) are planes.



Silk screen layer with superimposed traces and plane outlines

This is the assembled “low-ground-bounce board”. Due to time constraints that had to align to the class schedule, the well-designed boards were not manufactured, and my partner has the “poorly designed” ground-bounce board. This board was also assembled by hand (0806 parts) as opposed to solder paste and reflow.

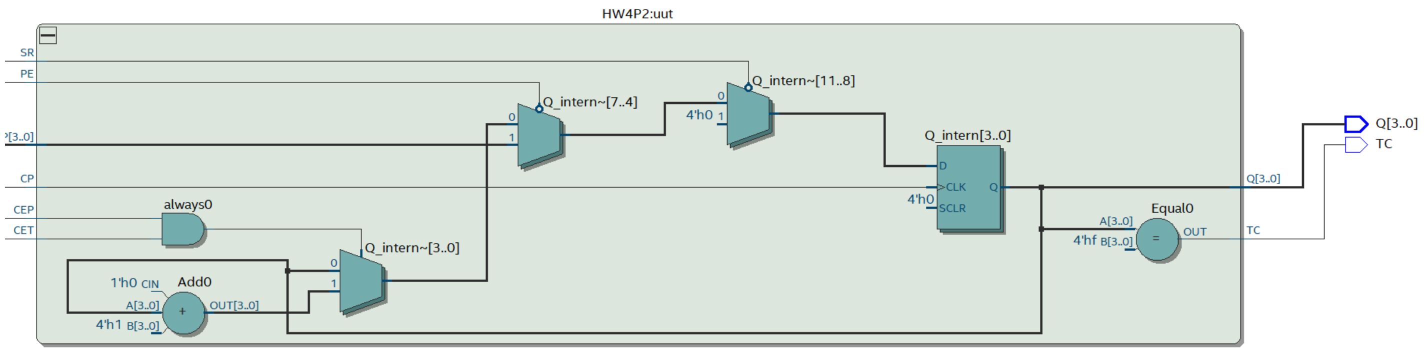


Hand-assembled timer board

Binary Counter Replacement Chip

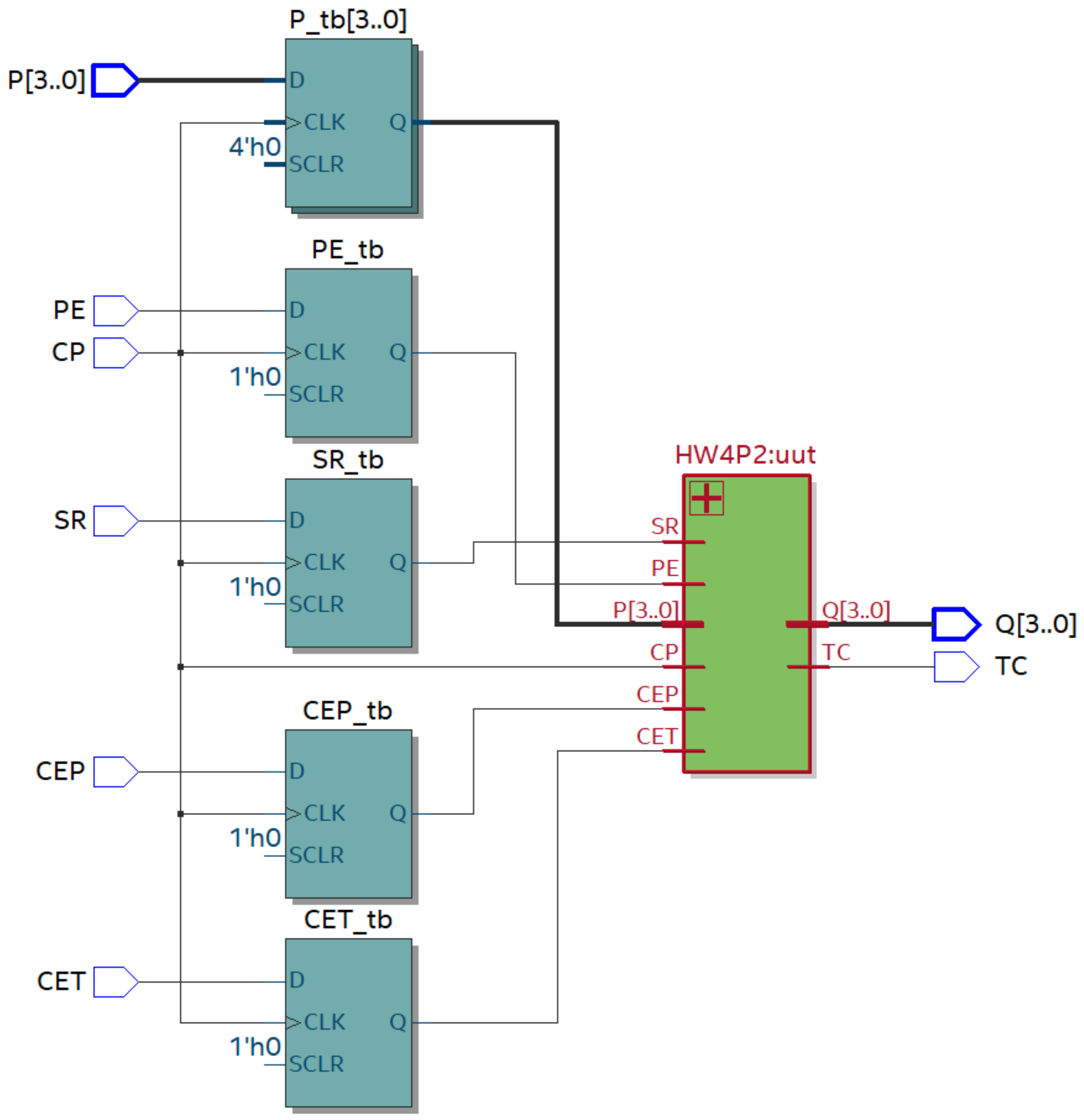
This was a short project to replace the SN74LS163A binary counter chip from the 80's with a CPLD/FPGA functionally equivalent part. The idea was to show how you can implement a design that has since been discontinued by the manufacturer by replacing it with a small logic chip that is functionally equivalent. Exact timing closure can also be achieved but in this case, only the I/O functionality and synchronous behavior needed to be modeled. This project was first done in VHDL and then later again in Verilog.

I started by defining the internal core module that defines synchronous operations and implements a control hierarchy based on external inputs. This core unit generated an RTL block that shows the hierarchy of inputs between count (CEP and CET), load (PE), and reset (SR) by the mux chain in series. These are held by a d flip flop for stability and then output with an overflow bit as the counter outputs.



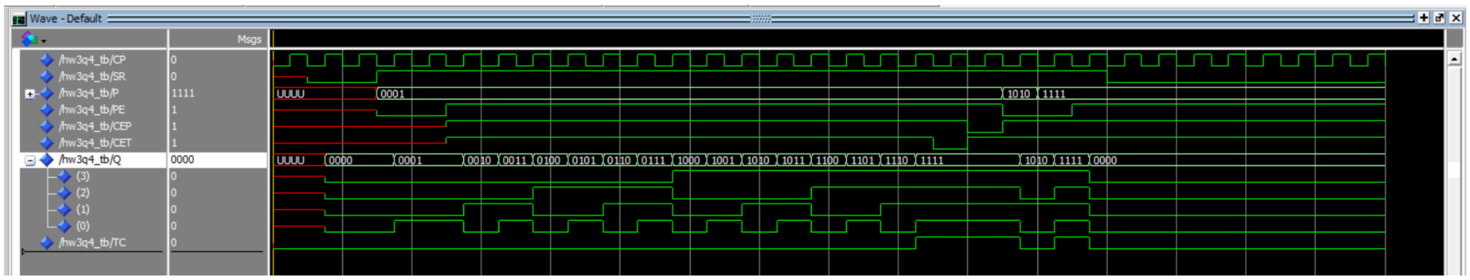
Core function module

Afterwards, I wrote a small wrapper module that clocked the inputs so I could write the timing constraints for the paths and tune the timing data (in a Max10 Flash-based FPGA, the timing in slow, uncooled silicon was approx. 109MHz without aggressive optimizations, almost 4x as fast as the original chip).



Timing test bench (output internally clocked)

Finally, after this I wrote a small wrapper that just brings the inputs and outputs out to the synthesizer so I could use commands that cannot fully synthesize in hardware while still testing the hardware-synthesizable core. The output waveform can be seen here:

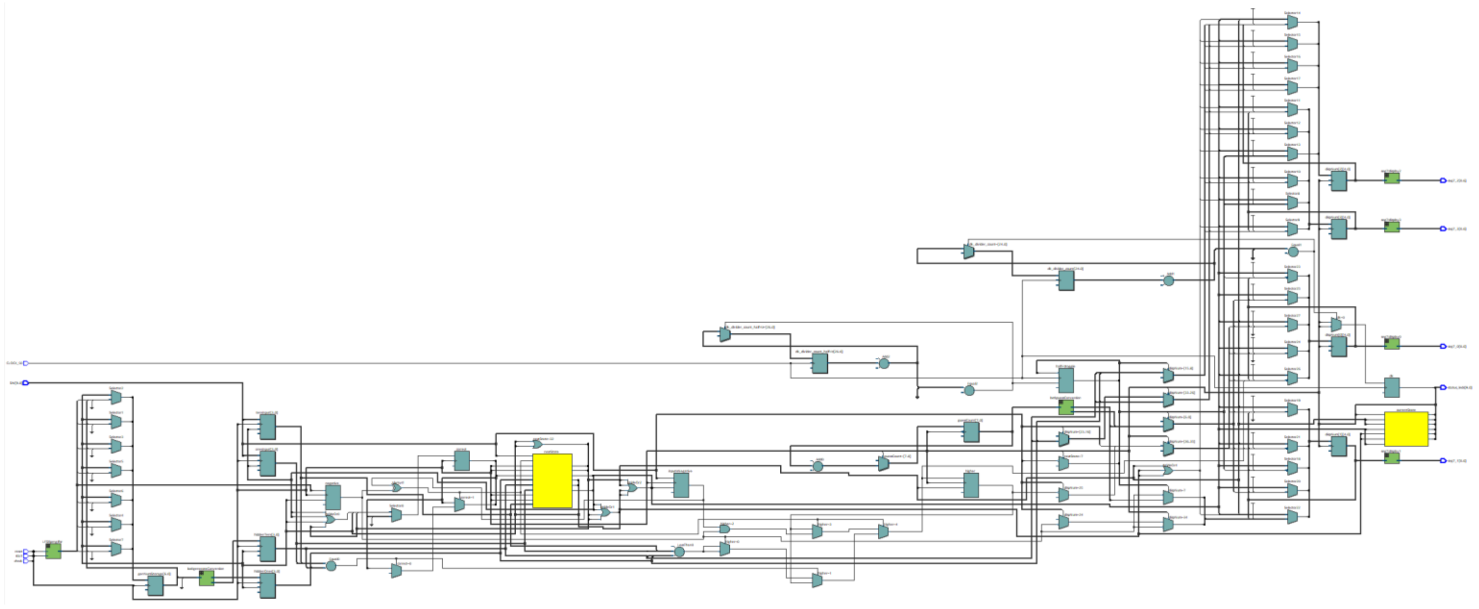


Testbench waveform

The gist of the test is simply that the core was slowly brought up to operational state by clocking it, resetting to default, loading a value, counting until rollover, and testing the hold states.

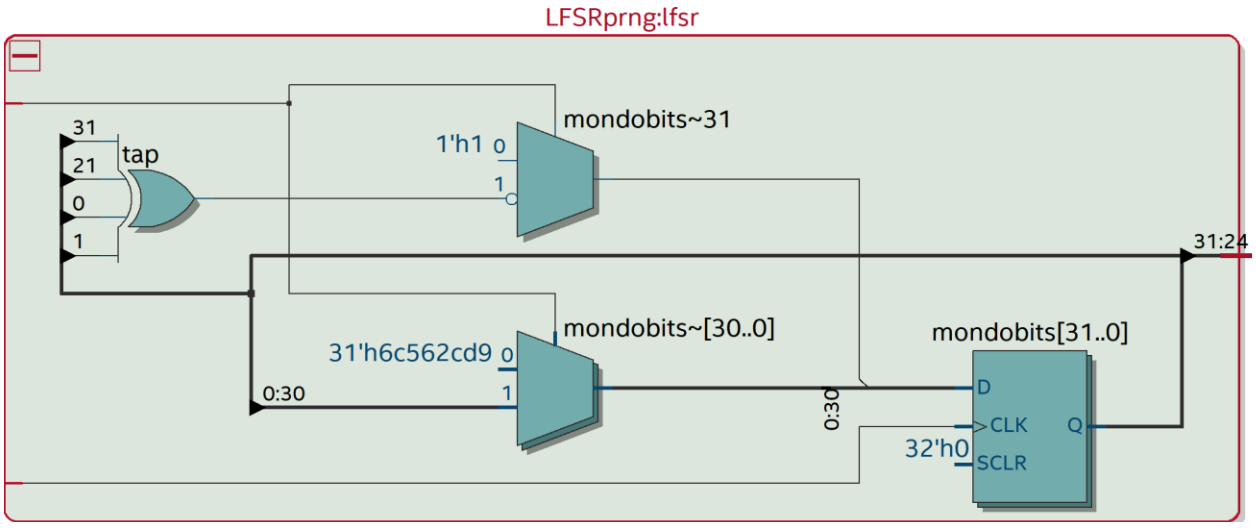
FPGA-Based Guessing Game

This project was implemented in Verilog and realized on the DE0's CycloneII FPGA, utilizing the pushbuttons, slide switches, and 7-segment displays for user feedback. The core of this design is the 32-bit linear feedback shift register that generates a pseudo-random number sequence of length 2^{32}. This shift register set had taps at positions 0, 1, 21, and 31 to ensure maximum length. While I've included the dull synthesis RTL, it is much easier to break the design into parts (design can be seen on my Github). The last time the CycloneII was supported in Quartus was 13.1 I believe so be wary of that. There is nothing special about the CycloneII hardware with respect to this project but the pinout was designed to target it's peripherals.

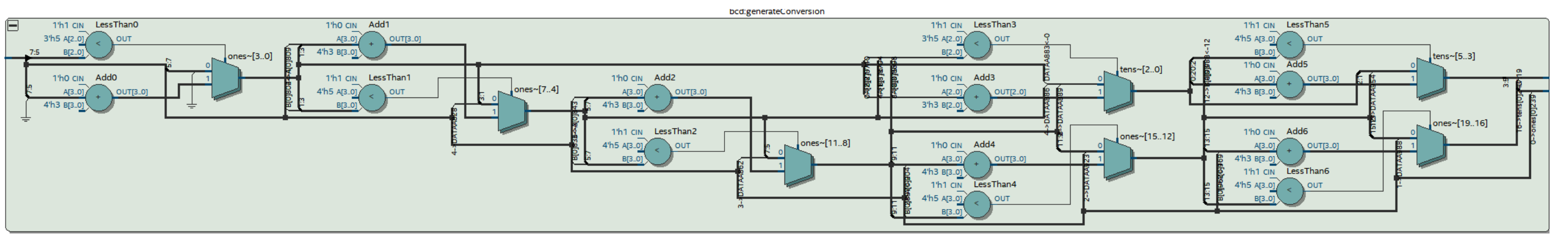


Full synthesis output

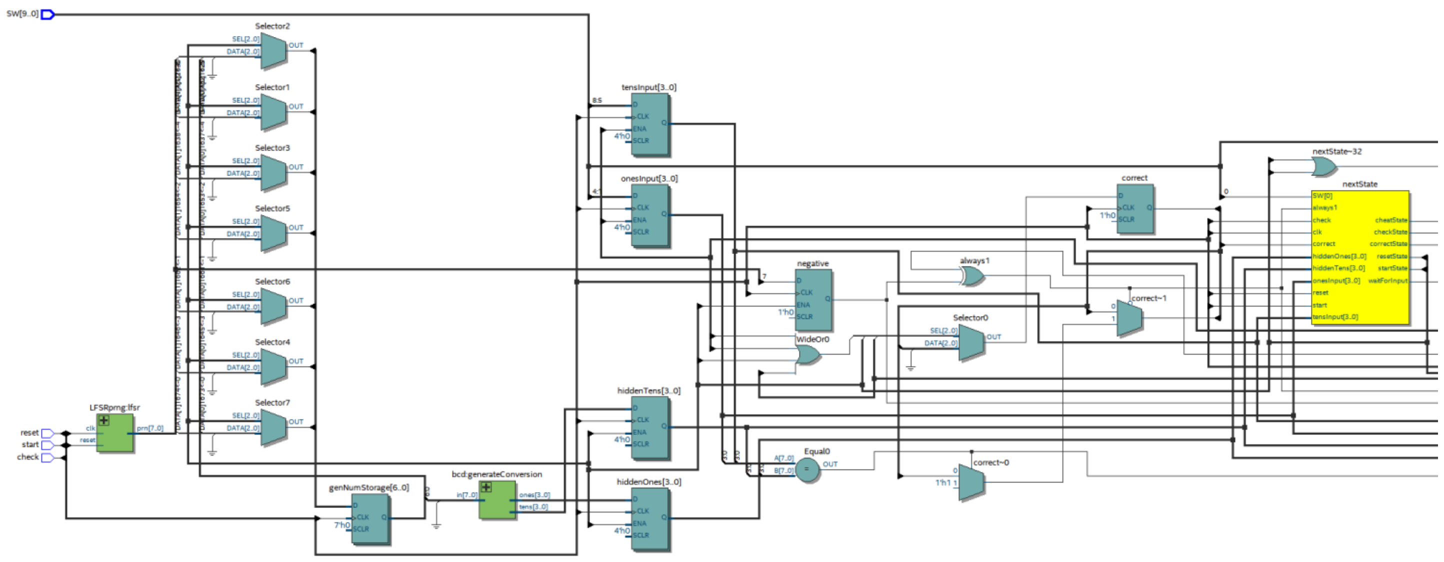
From left to right, this first generates the random number, takes the users guess via slide switches, converts the number decimal for display, and checks the two numbers against each other, sending the result into the state machine.



LFSR PRNG generator

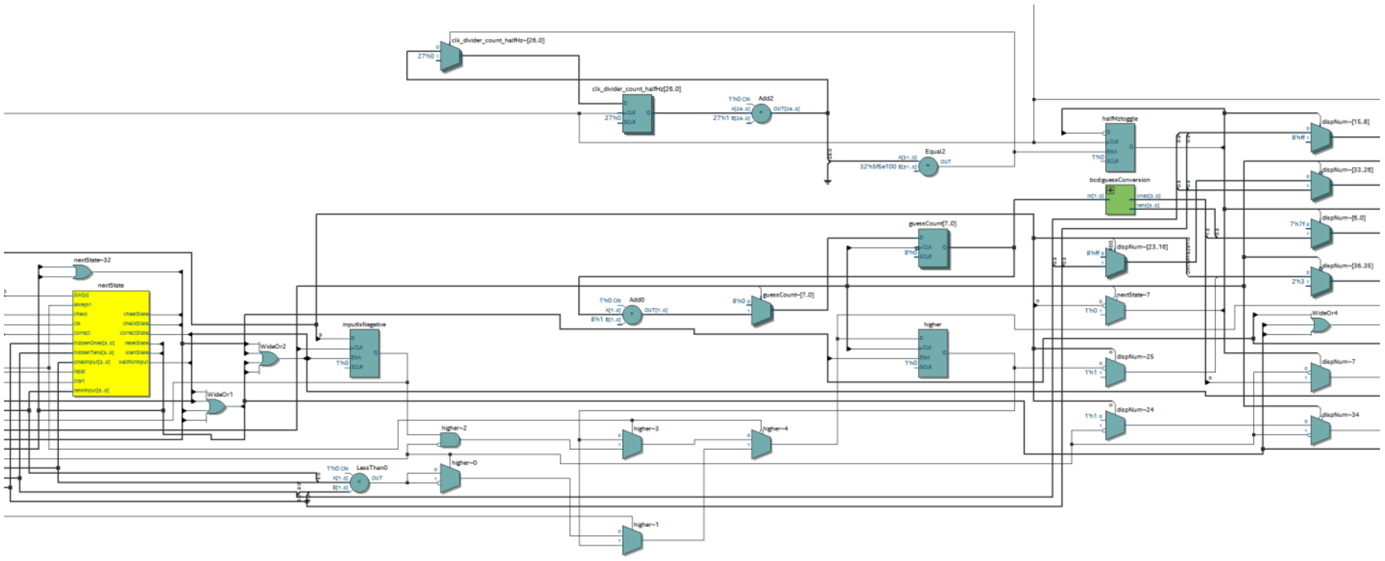


Unit conversion unit

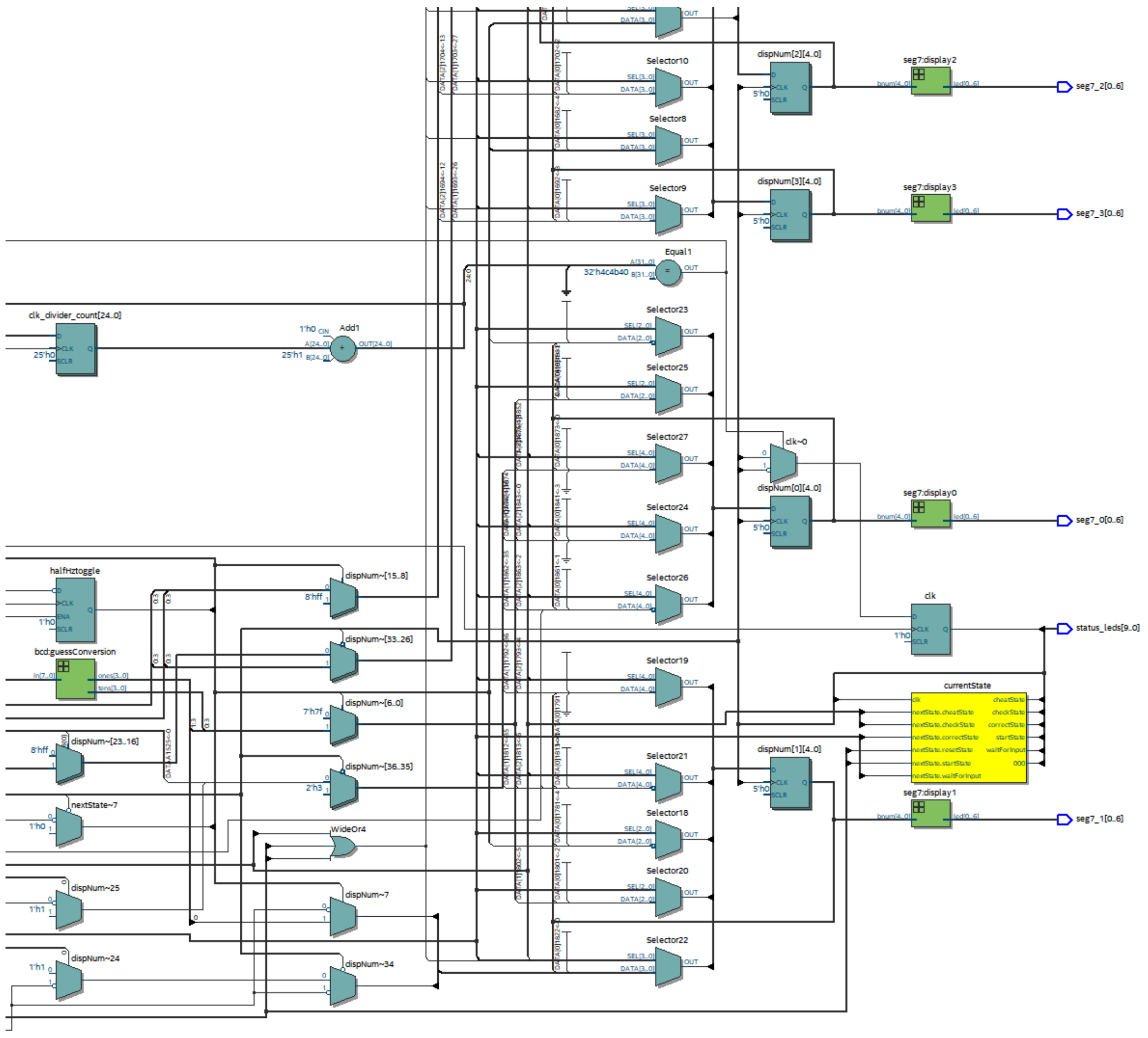


Board input to state machine

After going through the state machine, the hardware has determined what path of the game to go down (over, under, equal) and prepares the data to be displayed on the 7-segments, going through another instance of the unit converter above to prep the data.


State machine output

Immediately after this, the data flows into the 7-segment controllers and increments the guess counter before returning control to the inputs via the “current state” block.



7-segment controller

The last additions to this would be my work at LASP that I do not have pictures of as a result of the ITAR restrictions we had. The most notable of the work I did at LASP was the 1000+ signal interface responsible for routing all of the satellites test signals into a comprehensive test rack. Other notable work was my portion of the Spacecraft Bus Monitor programming and hardware debugging, various small circuit test fixtures for things like thruster heater overheat shutoff, test rack netlists and schematics, writing the GSE communication packet structure libraries, and designing/assembling the comprehensive test rack power module.

Other notable programming works include writing a TI-RTOS based microcontroller program that accepts Bluetooth commands from a phone to probe the temperature of the environment and return feedback, a program that places Colorado's 14ers in a map structure, with the edges weighted by cartesian distances between the peaks and hash-table addressed to boost look-up times, a program to control the movements of a wheeled robot designed and realized in class, and the entire 'slave' program in our semi-automated airlock design responsible for all telemetry and controls.