**Abstract**

The 3DPOV display is a three dimensional display that enhances 3D printing workflows by providing an organic representation of any 3D model. The system exploits the *persistence of vision effect* to form a perceived static image from layered motion (similar to how a spinning fan blade appears as a flat disc).

Layered segments of pulsing LEDs spin around a central shaft at 1800 rpm. The LEDs’ illumination is synchronized with specific points of the blades’ rotation to create the appearance of a 3D image.

**Discussion**

Over the course of building the 3DPOV display, we ran into a vast variety of issues and were exposed to issues we had previously never thought of.

Through-out the build process, getting a precise mechanical build was a huge struggle. We found that any small errors in our build process could cause error that would be too significant to compensate for. We knew from the beginning that we wanted to use a dual-helix formation because the pattern balances well (each opposing blade is 180 degree offset and the boards are all relatively identical). We found that even small errors in the alignment of the blades could cause issues.

To address this, we took a piece of foam board and cut a 45, 45, 90 triangle that we could use to ensure perfect alignment. In a fit of Murphy’s Law, we discovered that it is detrimental to rotate the system clockwise. What ends up happening when you spin clockwise is that any vibrations in the shaft will cause the nuts holding the blades in place to loosen. In a test run, the nuts loosened to the point that all blades came loose and snapped into place. Luckily, we had our safety shield at the time and additionally, the motor has a built in safety shut-off that turns off the motor when too much torque is applied to it.

Since the speed controller sends out a 3 phase signal to the motor, we discovered that just swapping two of the wires would change the direction that the motor travels.

Another place of great tribulations was the shaft coupling. Mr. Timothy Breen from the workshop helped us machine probably 4 or 5 shaft couplings before we got it right because we found that we needed much more precision in our coupling than initially planned. However, even with our final coupling we were able to measure .006” of play in the lower shaft. What must’ve happened is that the lower shaft slowly bent out of shape as we continued to test the system. The issue with having this amount of play in the shaft is that when we coupled the 8 inch carriage bolt to the lower shaft, that .006 inches of play translated into over .060 inches of play. This caused the system to be extremely unstable.

We were able to address this, however, by incorporating the bearing on the upper layer. We initially tested this by drilling a whole in a rectangular piece of plastic and fitting it to the carriage bolt. We then spun the shaft while holding the stock piece firmly to the top board. We observed that our play in the shaft disappeared. We then knew we needed to get a bearing to place on the top layer.

We discovered after planning the bearing system out that we would need to machine a coupling to sit on the carriage bolt that would have room to feed up the power cable without causing any interference. Upon installing the bearing and the coupling, the play wasn’t completely eliminated. The initial bearing coupling was made out of plastic and we had concerns that the bearing would melt the plastic and cause a bad situation to happen, so we meticulously crafted an aluminum coupling to work for us.

Even with all these improvements, the system still needed fine tuning. We found that we could use small nuts to counterbalance the system properly. The clear acrylic safety shield also acted as a nice sound buffer. All of the mechanical precision work that was done gave us a very good appreciation for mechanical systems we never thought of. For instance, the huge hard drivers you would find in banks were precision instruments that had a similar rotational system, but those had no more than a couple of micrometers of play.

From a hardware perspective, we also ran into various issues. In our shift from the old Melexis MLX92241 Hall Effect sensor that was a current output device, we needed to find a device that would give us a digital voltage output that would eliminate the external Schmitt trigger.

Initially, we had chosen the Toshiba TCS20DPR Hall Effect sensor. Upon doing testing, we discovered it had one singular issue that would cripple our system from working. An inspection of the datasheet revealed that the sensor operated at 20 Hz, a rate much slower than our rotational speed of 30 Hz. We then had to go back to the drawing board and find a new sensor.

We ordered 5 or 6 different Hall Effect sensors and the one we chose out of the bunch was the Melexis MLX92212LSE-ABA-000 unipolar switching digital output Hall Effect sensor. This sensor has a switching frequency of 10 kHz, so it was more than capable of handling our work capacity.

We also spent a good deal of time choosing the right Memory device. As we described earlier, we had to do extensive testing and benchmarking to determine which device would suit our needs the best. The benchmarking on the SD card always proved to be mixed. On one hand, we could get an SD card with enormous amounts of storage, however, the 8-bit Arduino proved to be too weak to take advantage of the speed available with SD cards (additionally, we didn’t have the funds to license the SD card specification for our project).

The EEPROM memory chips were very promising initially, but they had the write cycle issue that was a bit of a bummer. However, if we never had to write to them after an initial start-up, they would be an ideal solution because they can read indefinitely.

The SRAM chips were also very promising. They offered the same performance as the SRAM, but without the write cycle dependencies. As a down side, the SRAM chips are volatile memory, meaning that any loss in power erases the data on the system almost immediately. This trade-off is manageable because the ATmega chips can be programmed while the board is powered.

Our decision on the SRAM chips came down to read and write performance. The lack of having a write cycle was critical for us in implementing our ideas.

Another issue we ran into, one that we don’t fully understand, was again related to the Hall Effect sensor. Whenever we ran the system before moving to the 8 layer 3D system, we didn’t have to worry about any form of interference on the interrupt line. However we found that once we connected all the layers together, the boards would start randomly triggering at spots. The distance between the layers wasn’t great enough to have to consider a distributed model (nor was the signal at a high enough frequency).

One theory we had to the random triggering had to do with the Hall Effect sensor picking up spurious electromagnetic emissions. Since we are sending a large amount of current up the wire and said wires are spinning very quickly, we know from Maxwell’s equations that a time-varying or space-varying current will induce Magnetic or Electric fields. We think the sensor was possibly sensitive enough to pick this up.

We solved our issue by doing an extra check in software when our interrupt is called. Basically, the first thing the interrupt service routine does is it checks if the signal is still low. If this conditional is false, we exit from the ISR and move on.

We also ran into issues in our PCB layout phase. Our initial board layout was far too aggressive and our spot-checking proved to be very poor. Our boards were so bad that we decided it would be easier to just redo them. The second revisions only had two minor errors in them, so we were able to use them.

We also ran into a minor road bump in trying to cut the PCBs. ExpressPCB gave us 4 copies of our board, each of which had 3 copies of the individual boards on them. We first tried to use the guillotine cutter in lab to chop up the boards, but it proved futile to attempt to use it. Dr. Brandon Choi was gracious enough to let us use his PCB cutter to slice up our boards.

Another interesting error we had in the boards was that in order to burn the bootloader, we needed to supply power to the boards through the external power system even though the sparkfun AVR programmer has a built in power supply system. We were very concerned that the reason we couldn’t program the boards was because the SPI lines were being loaded by the LED drivers and the memory chips. Fortunately, this wasn’t the case.

A final hardware error we had related to the SRAMS. It turned out that there was no guarantee that the memory chips would start in SPI mode and in order to send the reset command you had to do so in SDI or SQI. We had to use software SPI to solve this issue and send the reset command if necessary.