**3D**POV  
**3D** **P**ersistence **o**f **V**ision Display

**Author(s)** : Philip D. Geramian, Alexander E. Julian, Lyle J. Moffit, Roman S. Olesh

**Date** : April 28, 2015

**Abstract**

**Introduction**

It is important to be able to visualize what you are working on especially in the 3D printing atmosphere. Seeing an object on a computer screen can be misleading since the image is not always 100 percent actual size, for this reason we have come up with the 3DPOV. With 3DPOV display you can accurately represent any object that is to be printed before you proceed with the print. Being able to see what the object looks like in real space before it is printed can save time and materials before printing something that may end up not being exactly what the user wanted. To be able to achieve a Persistence of Vision effect we build a system that rotates LEDs at 1800 rpm, during each rotation LEDs will light up at a preset position displaying an image. This build incorporated the use of brushless motor to drive the system, copper ring and carbon brush to transfer power to the rotating LEDs. Arduinos were built-in on each of the 8 layers which were used to control the LEDs through the use of LED drivers.

The concept of POV is that the human brain is capable of keeping an image in mind for 1/16th of a second. This allows for society to enjoy things like film, and television where (in the NTSC standard) a new frame is generated every 1/30th of a second, so it makes a sequence of images appear as a moving video. This allows us to take our system, spinning at 1800 RPM, to have an effective refresh rate equivalent to NTSC.

The Serial Peripheral Interface Bus, or more commonly SPI bus, is a communication protocol used to interface different systems together in a single master to one or many slave configuration mode. This allows for one to have a single master unit that may cost a lot of money, but is very good at processing data and have it interface with many, sometimes cheaper, slave units that are very good at doing a single task. In our system this is exemplified by the SRAM chips and the LED drivers that control our displays. In the SPI protocol every full duplex device has four wires going to it for just SPI purposes. These four wires are: Clock, Master out Slave in (MOSI), Master in Slave out (MISO), and Chip Select (/CS).



Figure 0: SPI Basic Functionality

It should be noted that the standard has chip select as an active low line. The basic functionaltiy is as follows, when the /CS pin is pulled low by the master it then generates a clock signal and on every clock cycle it shifts out one byte of data at a time over MOSI, while the slave device shifts out one byte at a time on MISO. Once the transfer is completed the clock stops and the chip select line is pulled high.

**Methods**

The 3DPOV display is a very complex system containing many interacting subsystems. The 3DPOV display can be broken into four distinct categories: mechanical design, fabrication, and verification, hardware design and verification, software design and verification, and PCB design and fabrication.

**Mechanical Design, Fabrication, & Verification**

We did extensive design and fabrication with our mechanical system because it is one of the most integral parts of the system. In order to achieve persistence of vision, we need to maintain (approximately) 1800 rpms and the mechanical vibrations need to be minimized.

Our design uses a DC brushless motor in order to drive the system. We chose a Turnigy d2386-11 750 KV DC brushless motor to drive the system. The KV measurement for DC brushless motors stands for RPM/volt, where a lower number generally indicates the motor outputs more torque, but at the expense of speed. The 750 KV DC brushless motor was a good middle ground between torque and speed. Additionally, DC brushless motors are generally very stable because they have internal sensors that communicate back to a controller.

In order to control the DC brushless motor, we needed to use an electronic speed controller [ESC] with which we could control the speed with a simple servo signal. The interaction looks as follows:

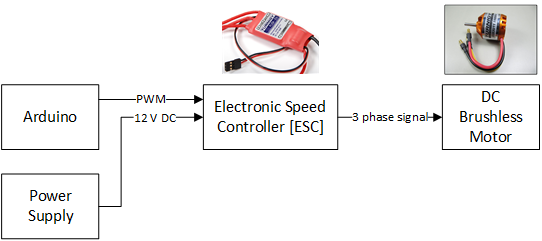


Figure 1: Motor Control Diagram

We realized quickly that the brushless motor wouldn’t be able to deliver enough torque. We also needed to consider how to build a very precise shaft that rotates completely on axis. To accomplish we decided on a timing belt pulley system that would drive a take apart DC motor. With the timing belt system we can use a 1:3 configuration that yields 3 times the torque at the output at the cost of 1/3 the speed. Therefore, the motor needs to spin at 5400 rpm so that the main shaft can spin at 1800 rpm.

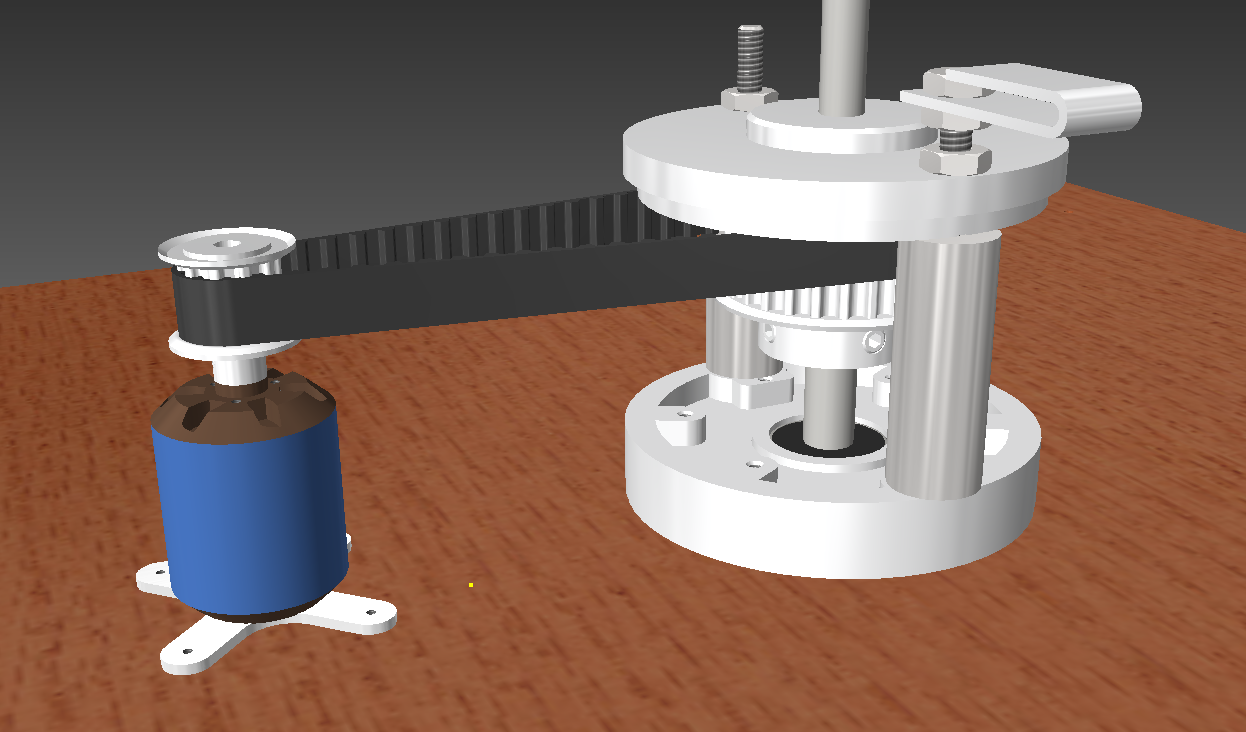


Figure 2: 3D Model of drive system

Pictured above is a 3D model of our drive system. We fabricated the main shaft (pictured to the right) by taking apart a DC brushed motor. We took the DC brushed motor and removed the casing by unscrewing the two through bolts. Next, we carefully removed the windings with a hack saw and then removed the internal wirings and the carbon graphite brushes. We made sure to keep the brushes intact so that they could be reused.

The final step was to machine the metal frame and shaft so that it set at the appropriate dimensions we needed. This involved utilizing our machine shop to cut off the extra lip on the base portion of the motor casing and using a lathe to machine the shaft to spec. Then, a timing belt gear is attached to the shaft so that it can couple to the motor.

Since we removed the casing that supports the motor in place, we needed to make support pieces to brace the motor. We used the machine shop to lathe to cylindrical support pieces (visible in the diagram above) to fully support the frame.

With the drive system integrated, we needed a method of attaching the wings to the shaft. After some debate and designing, we decided that we could couple a carriage bolt to the lower shaft using a custom machined coupling and then affix the wings to the blade using nuts and rubber washers.

Pictured below is the original shaft (left), the shaft coupling, a bearing coupling, and the carriage bolt. The carriage bolt is a 5/16” threaded carriage bolt that can be found at any hardware store. In order to fabricate the couplings, we used a lathe to bore out the inner diameter of the lower shaft to approximately 3/8” and the inner diameter of the upper portion was lathed to approximately 7mm.

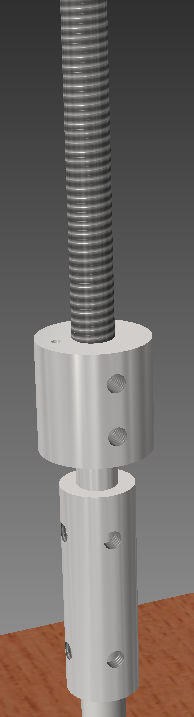


Figure 3: 3D Model of coupling system

The upper coupling sits on the carriage bolt and sits inside of a bearing that is attached to the upper layer. This bearing acts as a method of pinching off any imperfections in the lower shaft that would cause the shaft to rotate off axis, creating bad instability. This bearing is crucial in reducing noise and ensuring smooth rotations.

The following is a visualization of the bearing attached to the upper layer:

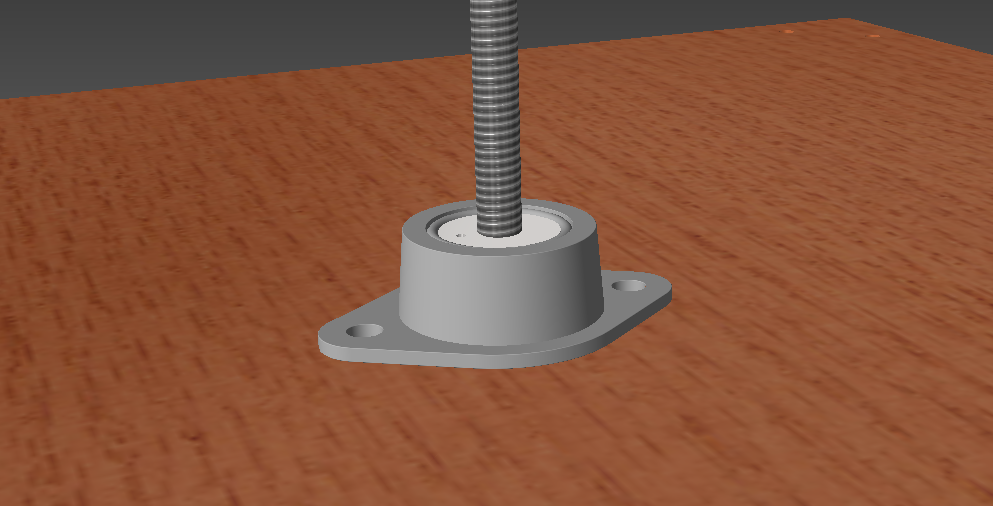


Figure 4: 3D Model of bearing system

As can be seen, the coupling sits firmly in the three quarter inch bearing. We needed to build a separate coupling for the carriage bolt because we needed a clever way of getting the power up to the boards. We had brainstormed methods of doing so and a standard slip ring didn’t seem to be a feasible solution both economically and for size constraint reasons. Therefore, we re-used the carbon graphite brushes to transfer power into a machined super conductive coper alloy ring. This ring would spin with the central shaft and a wire attached to the ring delivers power to the boards.

The copper ring system looks as follows:

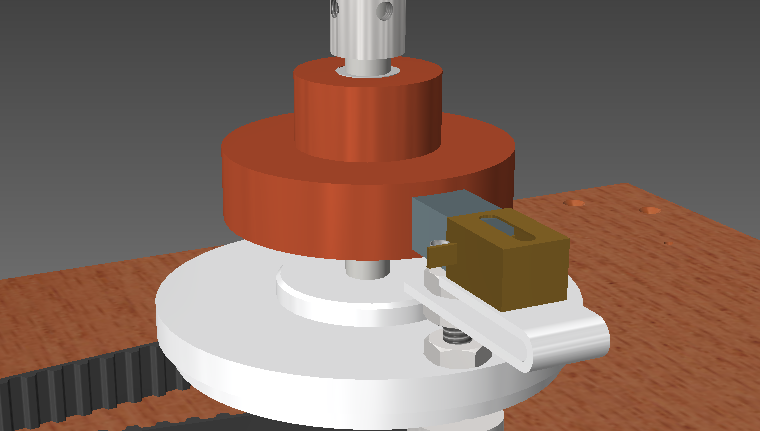


Figure 5: 3D Model of copper ring power transfer system

The coupling that sits inside the bearing freely spins with the shaft and an 18 gauge wire can comfortably fit inside of a whole drilled into the coupling allowing connection to be made. It might be difficult to see in the diagram, but there is a plastic sheathing inside the copper ring to isolate it from the shaft. We did this so that we could send ground up through the aluminum/steel chassis and the boards could then simply pull ground off the carriage bolt.

In the next round of fabrications, we focused on building a fully functional wooden platform with a detachable safety peripheral. The wooden based was constructed with ¾” thick solid oak wood. We cut two squares for the two layers. We made the top layer slightly smaller than the bottom layer. The bottom layer was cut to 17 in2 and the top layer was cut to approximately 16.625 in2 using a table saw.

Holes were drilled to countersink standard sheet rock screws into the boards so that we could attach legs to both layers. Additionally, holes were drilled in the center to allow the DC motor assembly to sit flush and for the bearing and coupling to comfortably fit. 7/8” angle brackets were used to affix the legs of the top layer to the bottom layer.

With the base layers put together, we needed a method of ensuring that the system wouldn’t slide all over the place when it is sitting on a plane table. To solve this problem, we took a sheet of gasket rubber from the hardware store and cut out squares that would fit on the bottom of the feet attached to the bottom board. This allows the board to get a really good grip on any table and stay firmly in place.

At this point, our base with the mechanical assembly looked as follows:

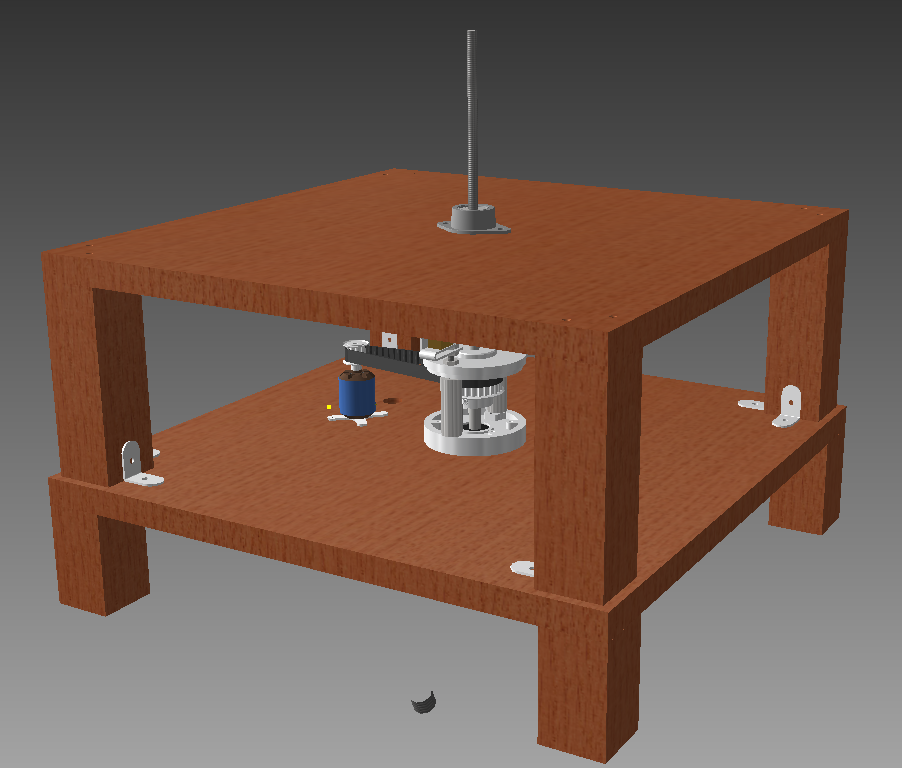


Figure 6: 3D Model of base assembly with mechanical pieces

The next piece that needed to be fabricated was our safety shield. We ordered a large quantity of 1/8” thick clear acrylic which we cut to size using a table saw with a special blade attached for cutting plastics. With the sides and top cut, we used a combination of super glue and silicon caulk to adjoin the pieces. Additionally, we cut the carriage bolt to size and fitted the electronics boards to the system.

Finally, we needed to fabricate a piece that would sit in front of the board that has all the control devices attached to it. We used left-over pieces from the build to make a nice looking control panel. The final result looked as follows:



Figure 7: 3D Model of complete assembly

**Hardware Design and Verification**

We started our hardware design with a set of criteria for the system. We quickly decided on logistics for the display such as how many layers (8), how many LEDs on a board (32), LED size (3 mm), and LED color (white).

With the logistics set out, we needed to determine the best solution for storing a bunch of data and creating a complete cohesive system. From our proof of concept, we knew we wanted to stick with the Maxim MAX6971 LED driver IC. This IC is a 16 output, 36V constant current LED driver that is interfaced to with SPI.

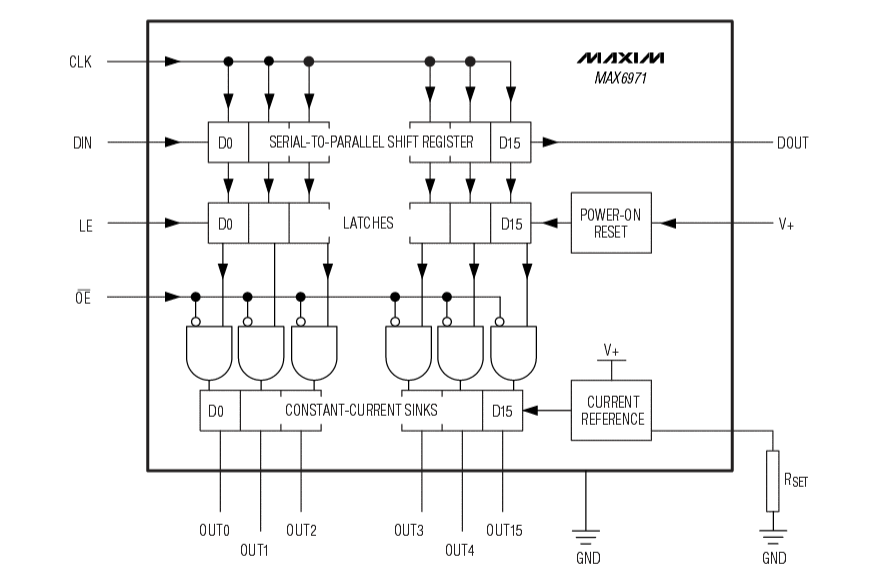


Figure 8: MAX6971 block diagram

The MAX6971 has a simple 16-bit shift register on its input that will always shift the next bit through regardless of the state of the latches (OE and LE). This means that you can’t feed back the data out to the master otherwise you’ll have noise on the data bus. By sending the latch enable (LE) high, the contents of the shift register are dumped out in parallel to the output latches. When the output enable (OE) is set low, the output drivers go from high impedance to being on, display whatever is in the output latches. Data is shifted into the shift registers MSB first and a resistor is needed to set the current reference for the constant-current sinks.

We chose a 680Ω resistor which gave us the following current to each LED:

In a worst case scenario, this means we would have to source of current. So we knew early on it would be important to have a power supply that could source enough current for us.

Before we addressed this issue, we sought to pick a suitable option for memory. We needed something we could talk to quickly with an 8 bit microcontroller and was easy enough to interface to. We had multiple options to test including: SD card (FAT file system, RAW binary data), EEPROM, and SRAM (volatile and non-volatile).

After doing testing on the SD card, we knew it wouldn’t be possible to get it to work quickly enough with an 8-bit microcontroller. Additionally, SD cards require 3.3V to run and the rest of our system was using 5V, so this was a consideration as well. An interesting thing we found though, was that the SD card was quick enough for entire revolution, but not for each individual theta position.

The EEPROM memory IC we were looking at was a MicroChip 25LC1024 1 Mbit SPI interface EEPROM chip. The following is the block diagram:

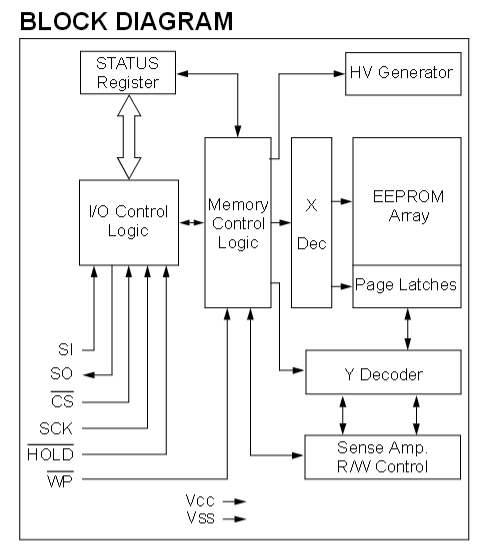


Figure 9: 25LC1024 block diagram

In short, to use the EEPROM memory chip, the user sends over an 8-bit instruction via SPI. Then depending on the instruction, additional SPI transfers are needed specify memory addresses and to send/receive data from the system. The great part about the EEPROM chip is that it is very fast. With 8-bit SPI we are able to read 32 bits of data in approximately 16 µs. The one drawback to EEPROM is that it has to go into a write cycle at the end of each “page” of data (where a page is every 256 bytes). This write cycle lasts several milliseconds which is not ideal.

Finally, the SRAM chips we tested were a MicroChip 23LC1024 1 Mbit SPI, SDI, and SQI SRAM chip. The block diagram for the SRAM is nearly identical to that of the EEPROM, however, since the SRAM is volatile we can get the same speed results, but we can write indefinitely without ever having to go into a read cycle.

To interface to SRAM chips, we needed the standard SPI connections (data in, data out, spi clock), as well as an active low chip select line and an active low hold line. The chip select allows you to put multiple memory chips on the same bus and then simply select which chip you want to talk to. The hold signal allows you to suspend the SPI transmission. This is actually incredibly useful because it will ignore any signal on the input/output until hold is raised back to a high logic level. If chip select is raised, however, the device does exit the transmission. This feature allows us to continuously read/write to the device and then pause and write to the LED drivers and then continue to read/write to the memory chips without having to resend the instruction and address.

The final piece of hardware that needed to be tested was the Hall Effect sensor. In our proof of concept, we used a Melexis MLX92241 Hall Effect sensor that would detect the magnetic field and through hysteresis, change at the output. The issue with this device is that it was a current output device, when we needed to have a voltage output device.

After ordering a bunch of different sensors, we ended choosing the Melexis MLX92212LSE-ABA-000 unipolar switching digital output Hall Effect sensor. The wiring diagram looks as follows:

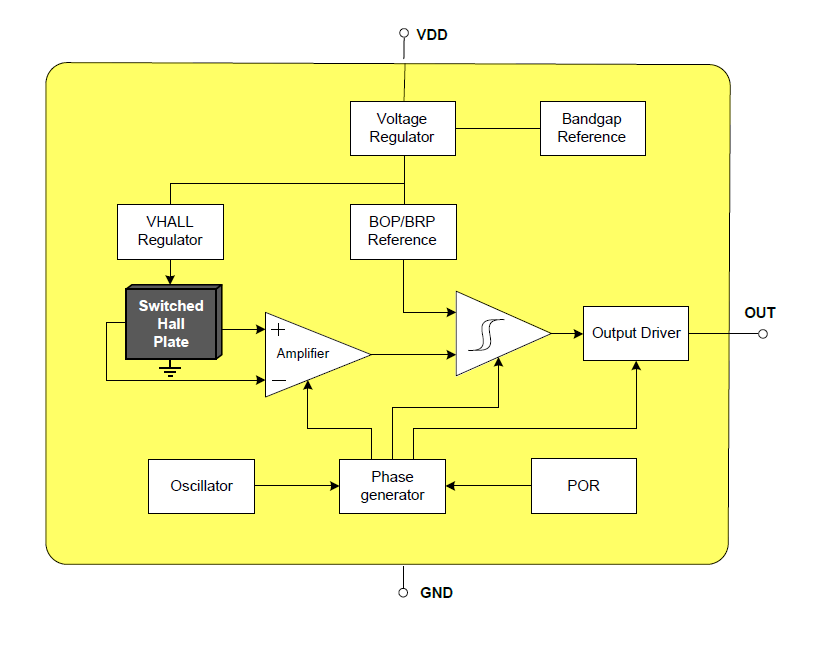


Figure 10: Hall Effect sensor functional diagram

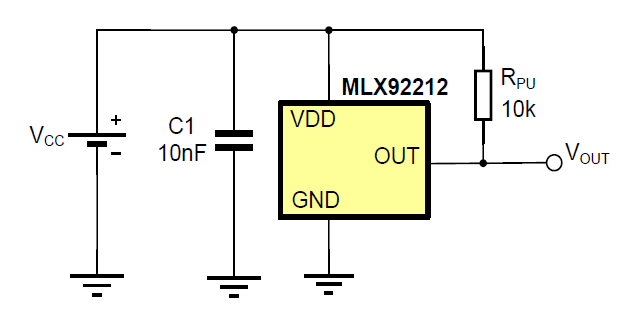


Figure 11: Hall Effect Sensor wiring diagram

The MLX92212 only needs a decoupling capacitor across power and ground and a pull-up resistor off the output to work. Additionally, it can switch on and off at up to 10 kHz which is much quicker than our requirements call for (30 Hz). Since the sensor is a unipolar switching device, it only detects one pole (South Pole in this case), so the direction of the magnet matters. Upon seeing a magnet field the output drops from 5V to approximately 10 mV and stays there until the magnetic field goes away. Thus, we can look for a falling edge on our input and interrupt on that condition.

With all the pieces in place, we could put together a functional diagram of the entire system:

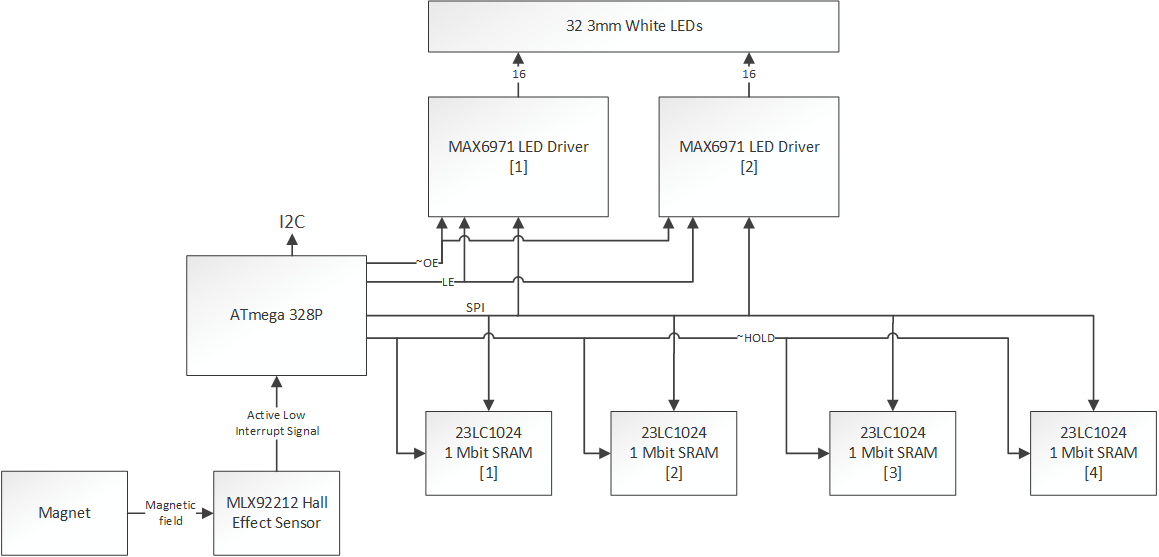


Figure 12: 3DPOV functional hardware block diagram

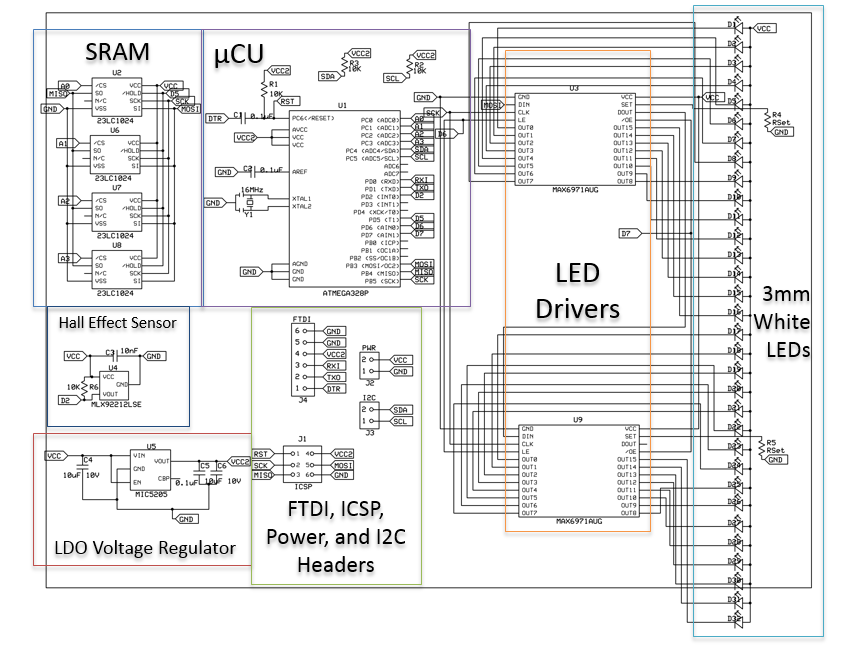


Figure 13: 3DPOV functional hardware schematic

The schematic is a synthesis of all of our components and has some new components that were added when we decided to have an Arduino built into our PCBs. The low drop off (LDO) voltage regulator is there to ensure that the ATmega 328p gets a solid 5V with an input of 5-12V. Additionally, the FTDI and ICSP headers were added in so we could program the Arduinos and burn the bootloader.

**PCB Design and Fabrication**

In the effort to make our end product look as professional as possible, we wanted to have our own custom PCBs printed. We used expressPCB to print our products because they have a very nice turnaround time and aren’t too expensive.

Our first revision of the boards had a lot of error, so we needed to go back through and fix everythingso we could have functional boards. The following is a picture of the board layout:

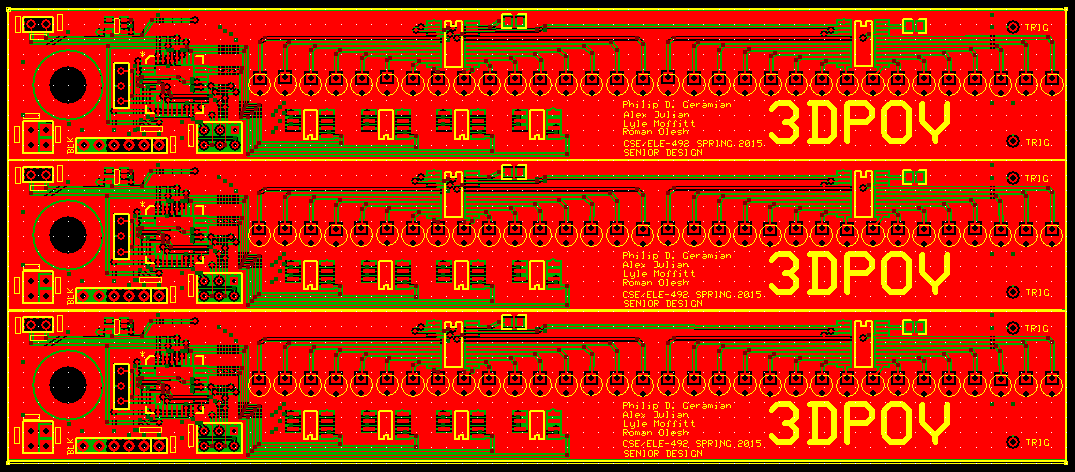


Figure 14: PCB layout as seen in expressPCB

With boards printed, we then had to populate the board with a combination of surface mount and through-hole components.

**Software Design and Verification**

**Results**

**Discussion**