Persistence of Vision LED Sphere

By

Jiale Quan

Lunan Li

Michael Ling

Final Report for ECE 445, Senior Design, Spring 2016

TA: Vivian Hou

4 May 2016

Project 26

Abstract

Our project features a spinning ring of 8-bit RGB Light Emitting Diodes (LEDs) capable of producing a Persistence of Vision (POV) effect. This effect refers to the optical illusion that occurs when multiple discrete images blend into a single image or animation. The ring of LEDs is able to show images at a rate of 24 frames per second (FPS), which produces cinema-like smoothness and clarity. Users can “draw” the picture they want to display using the graphical user interface (GUI) we developed, which we have used to create animations including Super Mario and a pixelated globe.

Table of Contents

1.0 Introduction

Our group plans to create a persistence of vision (POV) display. Our POV display will create a 3D hologram by utilizing a single string of LEDs rotating with two degrees of freedom. Consumer electronic companies have created one degree of freedom POV displays and usually sell them as clocks. Engineers have created 3D holograms using devices that utilize only one degree of freedom. Others have created two degree of freedom POV display, but they have not been able to create a smooth holographic image. We plan to take these POV displays to the next level of complexity. Our design will include a spinning ring of radially-facing-out LEDs that will spin about an axis parallel to the XY-plane. We will then attach this spinning ring to another motor that will spin about the Z-axis. The ring will have a spinning frequency of at least 30Hz to convey the persistence of vision concept. In order for the POV to take on full effect, a considerable amount of timing will have to be taken into consideration when coding how and when the LEDs will light up. Creating precise spinning frequencies poses mechanical and electrical challenges that will require extensive engineering to resolve.

2.0 Design

2.1 Block Diagram

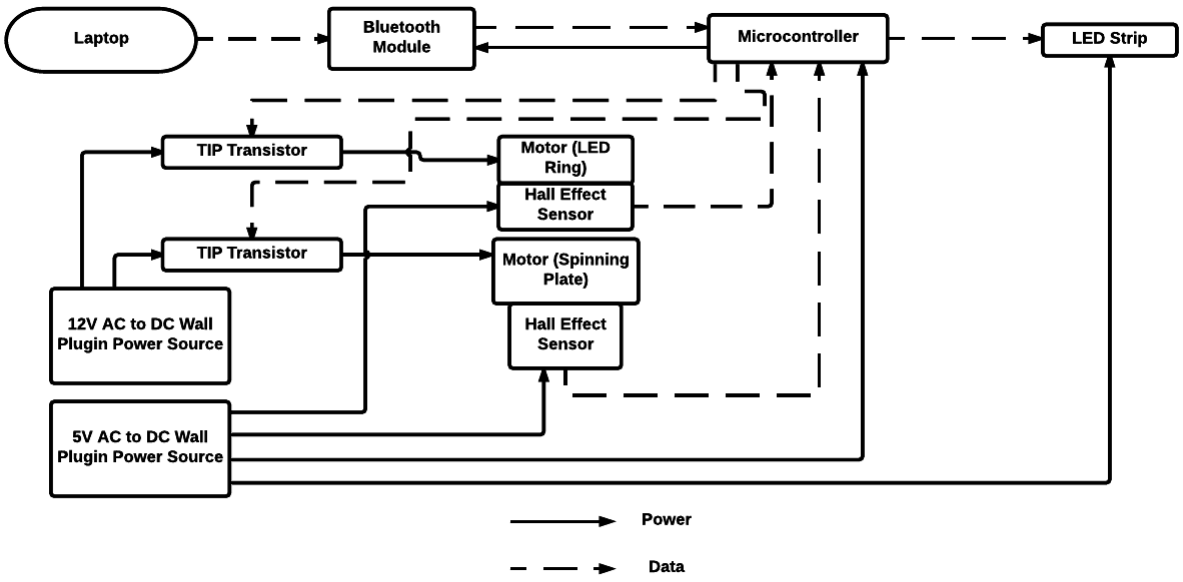

Figure 2.1 Whole System Overview

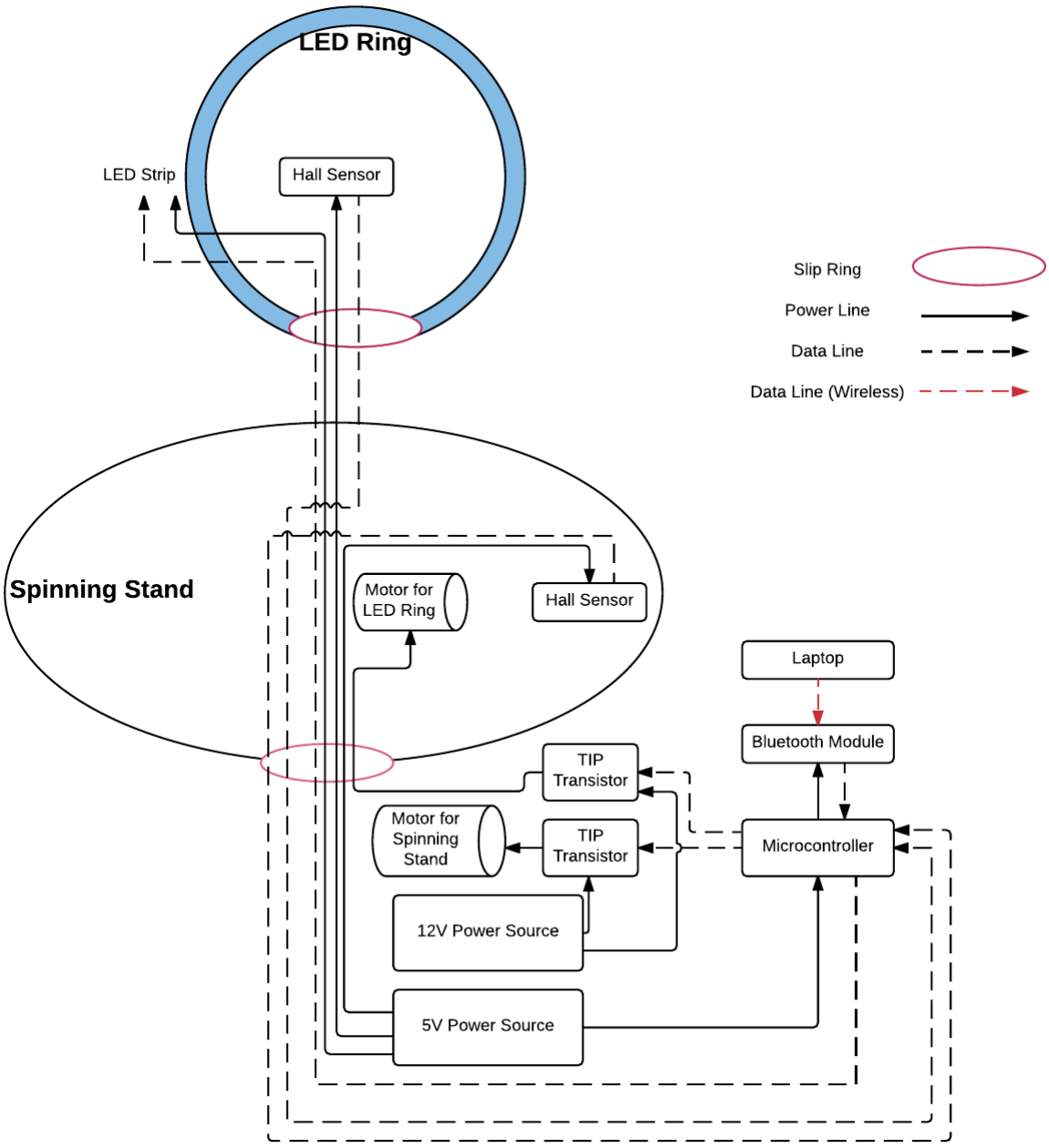

Figure 2.2: Whole System Block Diagram with Mechanical Portion

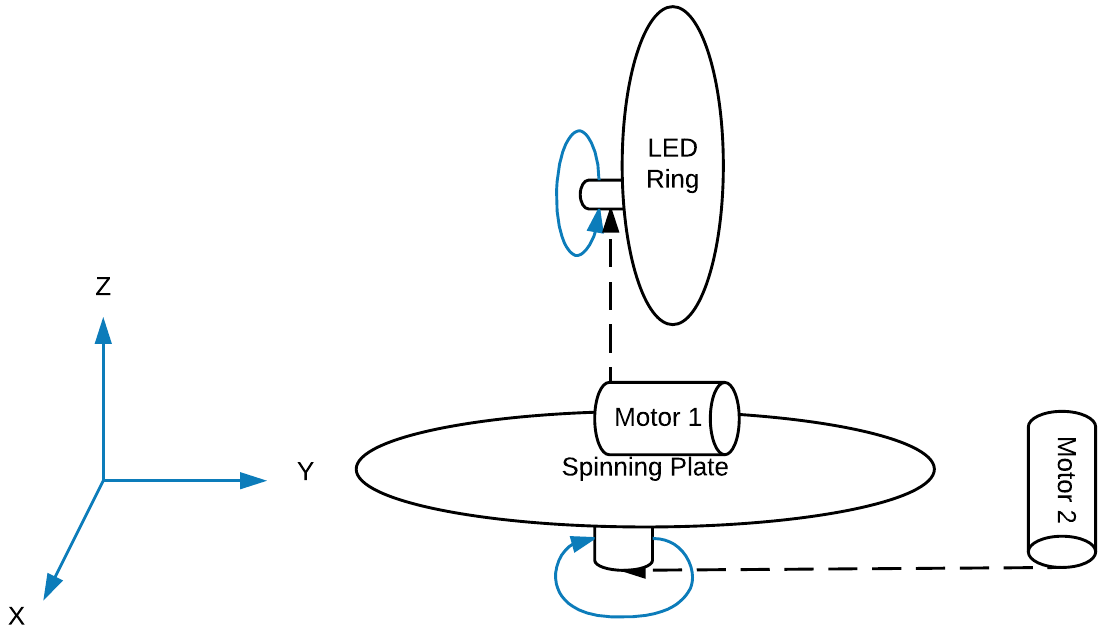

Figure 2.3: Mechanical Component

2.2 Design Details

2.2.1 Mechanical Component

Figure 2.3 is a simple, yet informative diagram that demonstrates *how* the two motors in our design will create two degrees of freedom for our project. Motor 2 will be connected to the Spinning Plate via a rubber engine belt along with a 1:4 gear ratio couplet. This means the Spinning Plate will spin four times every time the output shaft of Motor 2 completes one rotation. Motor 2 allows our display to rotate about the z-axis. This is one degree of freedom.

The other degree of freedom is made possible by Motor 1, which will be mounted on the Spinning Plate. This motor will be connected to the LED Ring, which will be attached to the Spinning Plate via two small towers (not shown in the diagram). The ring and the motor will also be attached to a 1:4 gear ratio couplet, and the ring will spin in about the xy-plane. This is the second degree of freedom in our project.

Finally, we will mount an RGB LED strip on the LED Ring such that the LEDs are facing *radially outward*. Our wheel will be 7 inches in diameter with a circumference of about 21 inches. Since our LED strip has a pixel density of a little less than four LEDs/inch, so our LED Ring will have 81 LEDs total. When both motors are spinning, the LED Ring will be spinning with two degrees of freedom. This will allow us to create a spherical surface.

2.2.1.1 Spinning Stand

The spinning stand will be the mechanical portion of our project. It will hold the LED ring (which will be spinning in the XY-plane) and microcontroller unit while spinning about the

Z-axis. This creates the two degrees of freedom we need to achieve the 3D effect of our POV display.

2.2.1.2 LED Ring

The LED Ring is the 7 inch diameter spinning wheel with the LEDs attached to it, and it will be attached to the spinning stand in such a way that it will spin around axis in XY plane while utilizing a slip ring for power and wires for data transmission. At first, we were refraining from using the slip ring to transmit data because we thought it was not very reliable when transmitting data at the speeds we required. However, we found that we can transmit the data via the slip ring and the whole process is pretty stable. Thus, we use slip ring to transmit data in our project.

2.2.1.3 Motors and Gears

In order for the persistence of vision phenomenon to take effect, the ring of LEDs must spin at a relatively fast rate. Our motors are rated to spin at 500 RPM at 12V, but we need the Spinning Plate to spin at 720 RPM (Equation 2.1) if we want an FPS of 24. Buying a motor that rotates at the speeds we require is too expensive. So, we will be using a gear ratio of 1:2 to increase the RPM of the Spinning Stand while keeping the motor's RPM at a manageable value. We will be using a 1:1 gear ratio for the LED Ring because the RPM necessary to evoke a refresh rate of 24 FPS is *less* than 30 RPM. Given the motors are rated at a free-run RPM of 500, a 1:1 gear ratio will suffice.

2.2.2 LED Strip

We choose to use DotStar APA102 LED strip in our project because it supports hardware SPI, which has high enough refresh rates for our POV display to run smoothly. It also has 24-bit RGB capabilities. The LED strip relies on serial data transmission, which allows for a low profile when it comes to wiring. In fact, only four wires are required to drive all 81 LEDs we plan to use in our project. We plan to create software that will precisely time when these LEDs turn on and off to create smooth images for our POV display.

2.2.3 Control Unit

The control unit will be consist of a PCB board with the Atmega328P microcontroller. We use this chip mainly because it is the microcontroller that is used in Arduino UNO, it will be easy to build and debug the program with the UNO [2]. The Atmega328P chip will transmit designated voltage to the base of TIP transistors which will then accordingly adjust the voltage across the motor thus control the spinning speed of the motor. We use EAGLE Layout Editor to design the schematic and printed circuit board for the microcontroller. The microcontroller will monitor the RPM of both motors using Hall Effect sensors. It will then use this data and manipulate the input of the PWM transistors wiring to each motor in order to control the motor voltage. The microcontroller will also transmit data via wires to the LEDs for timing and display purposes. There are many other microcontroller chips with higher operating frequency and larger flash memory than Atmega328P, and it is necessary to upgrade the microcontroller to better chips thus the limitation of transferring data may be solved.

2.2.4 Bluetooth Module

We will be using a Bluefruit EZ-link Bluetooth module to upload program remotely to minimize wires exposed for visual simplicity instead of using USB to upload program. The application of the module is straight-forward, we just pair the computer with the module and set the port to the Bluetooth port in the Arduino program, then we can upload the program through Bluetooth.

2.2.5 TIP120 - Pulse Width Modulation

Pulse-width modulation (PWM) is a modulation technique used to control the voltage a load receives. The microcontroller can change the output pulse width by software thus the average voltage output to the base of the transistor can be controlled without changing the voltage of power supply. The PD5 and PD6 pin on the ATmega328P chip support PWM. We use TIP120 NPN epitaxial Darlington transistor to control the motor rotation speed through the microcontroller. Our microcontroller will use the pulse signals from Hall Effect sensors to measure the speed of the motors, and control the rotation speed of the LED ring using the TIP120 accordingly. The TIP transistors we will use can handle voltages up to 60 V and currents up to 5 A.

2.2.6 Hall Effect Sensor

We use Hall Effect sensors to monitor the rotation speed of each motor. The Hall Effect sensor we use is US5881 which has south pole detection and is active low. When the sensor detects south pole of the magnet, it generates a LOW voltage signal, otherwise it stays at HIGH. By attaching the magnet to the spinning axle and counting the times that the sensor sends a falling edge signal, and being divided by the time interval, we can calculate the RPM of the motor. In addition, setting magnet to certain position on the axle, we also obtain the position of each LED when the magnetic field is detected. Although our motors have encoders for RPM monitoring, it lacks the function of indicating the position of the LED, so we finally choose to use Hall Effect sensor.

2.2.7 Power Supply

We will use two power sources for different parts: a 5V power supply with maximum current of 10A and a 12V power with maximum current of 5A. Each LED can draw a maximum of 60 mA when all three color channels (RGB) are at full brightness, which results in white light. AdaFruit suggests using the One-Third Model [49]. Assuming 1/3 brightness, it will be 20 mA for each unit, as they are all connected to the power supply parallel, the total current will be 81*0.02 A=1.62 A. The control unit will also be powered by the 5V source. For the motor, since the motor needs 12V voltage at its full speed, we will supply 12V voltage to assure the rotation speed, and because the stall current or maximum current is 5A, using a 12V with 5A maximum current is adequate. While considering using a voltage regulator to provide both 12V and 5V power using a single 12V AC to DC converter, it is risky to burden all current to one power source, with LED drawing 2A, motor drawing 4 to 5 A, it passes the maximum current of 12V source. Furthermore, it will trigger the circuit breaker in the power outlet if the current reaches 10A. Therefore, we decided to keep our two voltage source instead of just one.

2.4 Design Details

2.4.1 Software for the 2DOF Display

2.4.1.1 Algorithm for 2DOF

There are several changes in the 2DOF Algorithm in the real implementation due to the hardware limitations. First of all, in order to locate the starting position of the LED Strip and calculate RPM for both horizontal and vertical rotations, we decide to use attachInterrupt() from Arduino library to detect the magnet and record the time gap between consecutive rotations. Secondly, we decide to limit the number of pixels on each rotation due to the data transmission speed of our microcontroller.

We decide to have 24 frames per second (FPS) for both horizontal and vertical LED refresh rate because human eye can see optimal POV effect at this rate. We will consider each LED unit as a single frame, and we need to pass 24 LED units on a static point on the spherical surface every second. The RPM equations is:

$$RPM = \frac{number\ of\ rotations}{time\ for\ rotations}\ X\ 60\ second \tag{2.1}$$

For the horizontal spinning direction, it has 40 levels of LEDs and 2 LEDs at each level. Since we want to achieve 720 RPM, each rotation takes 83 milliseconds. According to our experiments, each time it takes around 1ms to send data to the LED strip. Then, we can only refresh around 80 times for the horizontal direction. The refresh rate would be 1 millisecond. The refresh rate equation is:

$$Refresh\ Rate = \frac{60\ seconds}{RPM\ \times\ number\ of\ refresh\ times\ in\ one\ rotation} \tag{2.2}$$

For the vertical spinning direction, it has 81 LEDs in total. To achieve 24fps, each LED should pass approximately one-third of the ring, which is around 23 RPM. Then, the refresh rate for vertical direction is 2.6 seconds.

In our algorithm, we will check vertical update first because vertical direction rotates slowly, and a little delay would make the whole display shakes up and down. After vertical update, we will do the horizontal update.

In the first flow chart, micros() returns the time since the program starts running in microseconds. h_time and v_time store the amount of time that horizontal and vertical directions taking separately. curr_x and curr_y store the switch information of the LED in specific division. h_revolution, and v_revolution are the number of rotations for horizontal and vertical direction separately. They will be reset every 5 rotations. h_rpm and v_rpm are the rpm value for both horizontal and vertical directions. h_increment and v_increment are the refresh rate for horizontal and vertical direction. Basically, we calculate the horizontal/vertical rpm every five rotations. We then use the rpm to calculate the refresh rate and update the LED strip by the refresh rate.

The second flow chart shows how attachInterrupts() works. Since the hall sensor sends low signal when it detects a south pole magnet, we keep checking whether there is a falling edge and increment the revolution number in the interrupt function.

2.4.1.2 Hardware limitations in 2DOF

There are several hardware limitations in our mechanical and hardware parts that limits the software development.

First of all, we are using TIP to offer motor voltage. However, due to the instability of the output of microcontroller, the motor for vertical direction cannot rotate at a fixed speed. Thus, the refresh rate for vertical rotation is changing, which causes the display pattern sometimes shifts up and down.

Secondly, we are using interrupts from Hall Effect sensors to calculate RPM and adjust LED positions. Nonetheless, data transmission between the LED Strip and the microcontroller cannot be interrupted, thus, certain interrupts may be blocked during data transmission. This causes two bad effects on the display. First of all, the LED position cannot be adjusted periodically. Sometimes, it takes extra one or two rotations to adjust the LED position. Secondly, the RPM cannot be calculated precisely. Since we are using a counter to count the number rotations and a timer to measure the amount of time passing. Blocking interrupts makes the same number of rotations taking longer time to achieve, thus, a smaller RPM, which also makes the display pattern shift up and down.

Last but not least, we are not able to display patterns with higher dimension due to the limit of data transmission. Since the hardware SPI on our microcontroller can only operate at 8 MHZ, we can only refresh around 80 times for horizontal direction. Any display pattern has more than 80 pixels on the horizontal dimension cannot be displayed on the spherical surface.

All in all, these limitations are caused by the hardware. We are unable to change them at the end of the semester, but it is still possible for us to solve them in the future. Certain solutions will be discussed in Section 5.4.

2.4.1.3 Flowchart for 2DOF

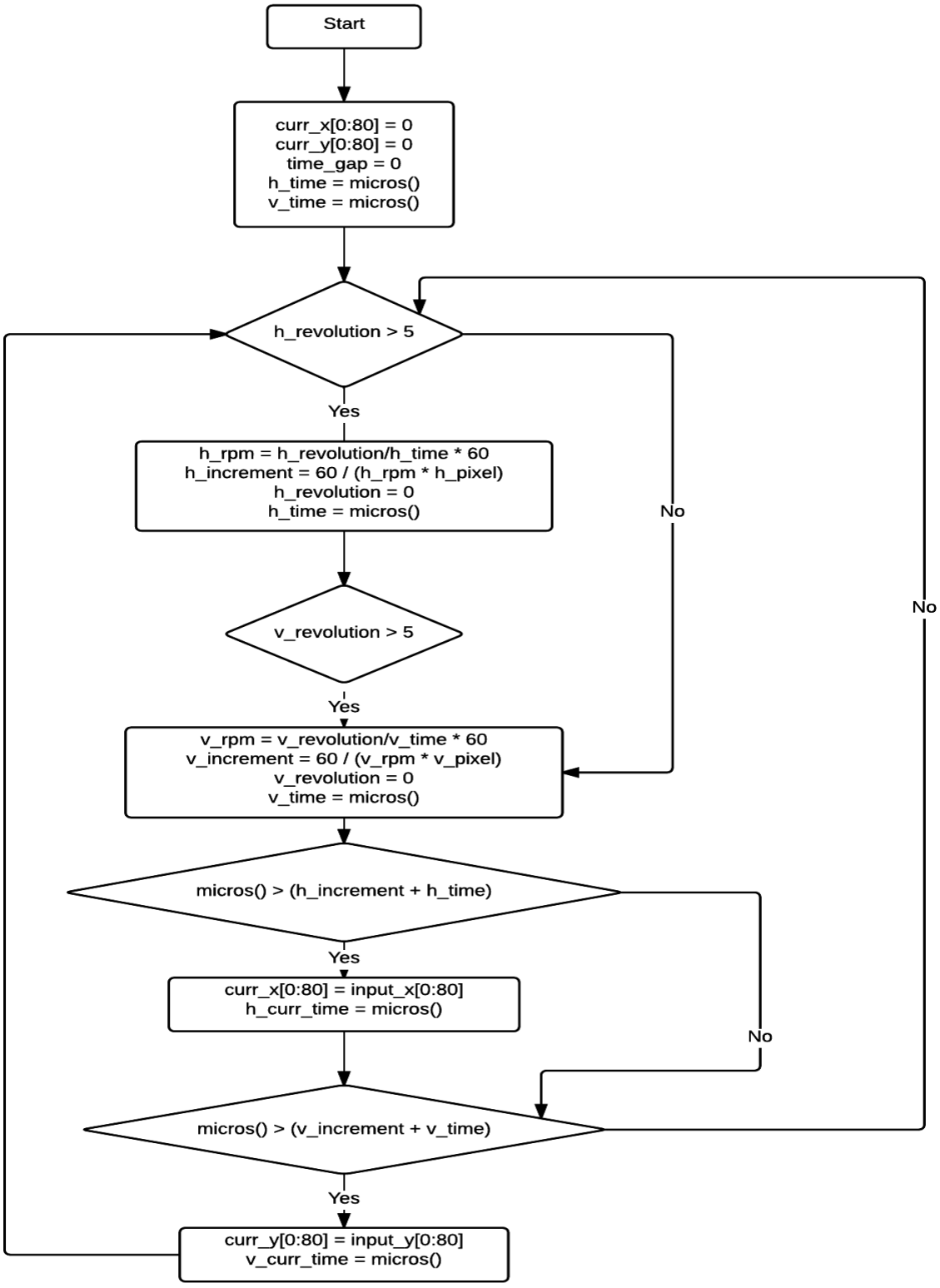

Figure 2.4: Flowchart for 2DOF Algorithm

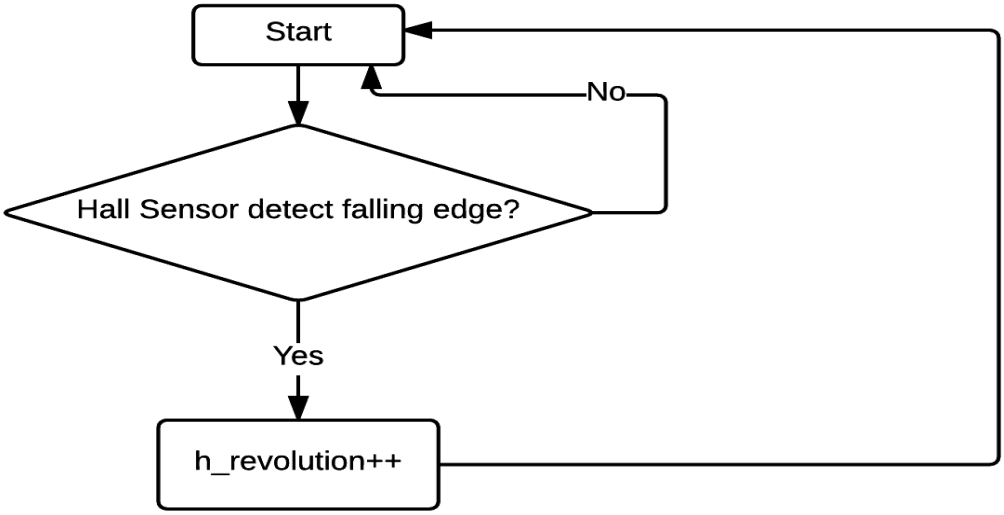

Figure 2.5: Flowchart for Interrupt

2.4.1.3 User Interface

We planned on creating Java applet that would convert a JPEG or PNG image into an unsigned long array whose size was equal to the resolution we planned on displaying. However, due to the limit memory on the Atmega328p, which is around 40k bytes, we cannot store an image in our microcontroller. Thus, we created a GUI with 80 * 40 grids instead. Figure 2.5 shows the GUI. Users have two ways to set the color pattern. One way is to fill in the row range and the column range, select the color from the dropdown menu and then click the Set button. The other way is to select the color first and then click on the grid that you want to set to that color. Once users finish setting the color pattern, they just need to click the Upload button and copy the generated string into the Arduino sketch. Then, we will load it into the Atmega328P chip that will be used in the MCU that's mounted on the LED Ring. This entails using an Arduino Uno board to program an Atmega328P chip, removing the chip from the Arduino, and physically mounting it onto the LED Ring. For now, that is the main method we plan on using to load images/animations onto our display

We will upload the Arduino Sketch to our microcontroller wirelessly as discussed in the previous Bluetooth section. Wireless communication is both safer and more convenient for users.

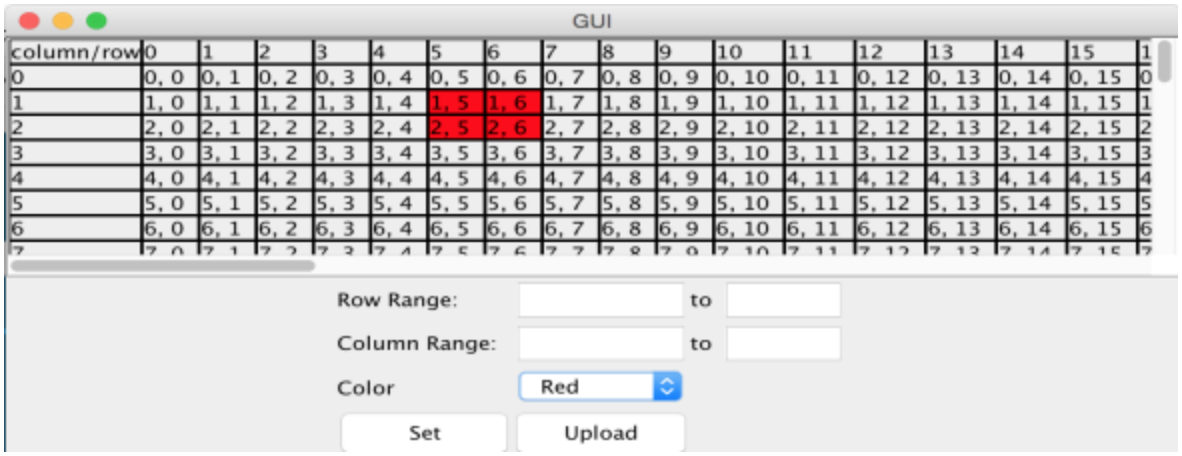

Figure 2.6: GUI with 80 * 40 pixels

2.4.2 Necessary Torque Analysis

We have split up the mechanical portion into two parts: the LED Ring and the Spinning Stand. We also have two motors in our project, each responsible for driving one of the mechanical components.

2.4.2.1 LED Ring Torque Analysis

The following calculations were performed to calculate the theoretical torque our motor will have to supply in order to drive the LED ring:

If we want a refresh rate of 24 FPS, we have to spin the LED ring about once every three seconds:

$$\frac{81\ LEDS}{1\ Rotation} \times \frac{1\ Second}{24\ LEDs} = \frac{27\ Seconds}{8\ Rotations} \tag{2.3}$$

Inverting the result of equation 2.3 gives 8 rotations per 27 seconds. We can now calculate the desired angular velocity:

$$\frac{8\ Rotations}{27\ Seconds} \times 2\pi\ Radians = \omega = \frac{16\pi Radians}{27\ Seconds} \tag{2.4}$$

Now, we can calculate the theoretical desired angular acceleration (assuming we want to get the LED ring up to speed in five seconds):

$$\frac{\omega}{t} = \alpha = \frac{16\pi\ Radians}{135\ Seconds^2} \tag{2.5}$$

We also assumed the LED ring can use the same moment of inertia equation as a hollow cylinder with:

$$I = \frac{1}{2}M(a^2 + b^2) \tag{2.6}$$

Our ring will be 7 inches in diameter (17.78 cm), and the LEDs are 4 mm tall. So, our *a* and *b* values are 0.1778 m and 0.1782 m, respectively. The LED Ring has a mass of 0.2 kg. Plugging the correct values into equation 2.5, we calculate that the LED Ring has a moment of inertia of 0.00633 kg-m^2. We now have all of the components necessary to calculate the torque required to drive our LED ring:

$$\tau = I\alpha \tag{2.7}$$

Plugging in the appropriate numbers, we arrive at the necessary torque of 0.00235 N-m. Which converts to 0.30768 oz-in. We now take our gear ratio into consideration because it will affect the torque our motors need to provide to the LED ring itself:

$$\tau_0 = \tau_i \times \frac{R_0}{R_i} \tag{2.8}$$

Since our gear ratio is 1:1, the torque needed to drive the LED Ring remains at 0.30768 oz-in. This is well below our motors' 84 oz-in torque rating.

2.4.2.2 Spinning Stand Torque Analysis

The process through which we performed our theoretical torque analysis of the spinning stand is nearly identical to the LED Ring process, except the Spinning Plate weighs approximately 0.5 kg. We also wanted to take 15 seconds to reach the appropriate speed, and we used the same moment of inertia equation as a disk:

$$I = \frac{1}{2}MR^2 \tag{2.9}$$

Performing the same calculations as the LED Ring (except with a gear ratio of 1:2, 2 LEDs instead of 81, and a radius of 3.5 in), we arrive at a desired torque of 17.795480 oz-in. This is also below our motors' 84 oz-in torque rating.

2.4.2.3 Torque Analysis Conclusion and Results

After calculating the necessary torques our motors would have to supply to drive our display at the proper RPMs, we were confident both of our motors would be able to do so in an actual test. After slowly increasing the voltages put across each individual motor, the motors were able to spin the Spinning Plate and LED Ring at 720 RPM at 9.2 V and 30 RPM at 1.4 V, respectively.

3.0 Design Verification

3.1 Mechanical Verifications

For the most part, testing the Requirements and Verifications for the mechanical of the project was pretty straight forward.

3.1.1 DC Motor

Ensuring the motors we chose for the project, two DC-motors rated at a stall-torque of 84 oz-in, could drive our system at manageable power needs, we slowly increased the voltage put across them until the desired RPM of the Spinning Plate and LED Ring were met. When all of the hardware was mounted on the display, the motor driving the Spinning Plate at 720 RPM drew about 1.4 A at 9.2 V. The motor driving the LED Ring at 30 RPM drew a mere 0.13 A at 1.4 V.

3.1.2 Mechanical Conclusion

In conclusion, the mechanical portion of our project performed pretty well. However, there is always room for improvements. We mention some of the improvements we would make to our project in section 5.4 “Future Work”.

3.2 Hardware Verifications

3.2.1 Hall Effect Sensor

For Hall Effect sensor, we monitor the output signal while the south pole of magnet approach the sensor, and it will generate a LOW signal. See Appendix B figure B.1.

3.2.2 TIP120 Transistor

For TIP120 transistor, we connect the base to the microcontroller PWM port, and collector to the GND port of motor, emitter to the power GND. We will use the program to change the width of the pulse and verify if the motor rotation speed changes accordingly. From the test we have run, the TIP120 transistor can modulate the voltage between collector and emitter from 0.5V to 12V, creating a voltage range from 0V to 11.5V to supply motors.

3.2.3 Microcontroller

For microcontroller, after connecting peripherals to the assigned port, we test the correctness of receiving from Hall Effect sensors and Bluetooth module, sending data to the LED strip and TIP120 transistor, we monitor the calculation of RPM in serial monitor through Serial.Print function comparing with the output of the sensor on the oscilloscope.

3.2.4 Power Supply

For power supply, we use a digital load device to test the power supply. This device will draw the maximum current while at designated voltage. For the test result, the 12 V power source can output 4.96 A while at 10.99V, and 5 V power source can output 9.98 A while at 3.37 V.

3.3 Software Verifications

For software verification, we will report the result for LED Strip, microcontroller data transmission speed and GUI.

3.1.1 The LED Strip

For the LED Strip, we tested the brightness and color settings following procedures in the R&V table. The result is pretty satisfactory. We are able to change the brightness of our LED strip. The result is shown in videos online. Also, we are able to set LED Strip to different colors for different patterns. The result is shown in Super Mario and the Globe videos. In the previous one, we set the Super Mario to red. For the later one, we set it to blue and green.

3.1.2 Data transmission speed for Microcontroller

For data transmission between LED strip and microcontroller, we used FastLED library. Each transaction takes around one millisecond. We can refresh around 1000 times per second. For each transaction, we send 2656 bits where each LED takes 32 bits and the start and end unit take another 64 bits. Thus, the actual speed for the microcontroller in this project is 2.656 Mbits, which fulfills the requirement in the R&V table.

3.1.3 GUI

For the GUI, we tested it by creating pattern on it and check whether the spherical surface displays the correct pattern. The result is pretty good. From the online video, we can display the Super Mario and the Globe.

**See full Requirements and Verifications Table in Appendix A*

4.0 Tolerance Analysis

4.1.1 Critical Component

The critical component we will do the tolerance analysis on is the motors and how we can achieve a desired FPS using our 12V DC motors and gear ratios.

4.1.2 Tolerance Analysis

The key for persistence of vision is the coordination between the spinning LED strip and the timing LEDs shining. The minimum FPS humans see continuous motion is 24 FPS [3]. We calculate how fast the motors will spin with the following equation:

$$60\ [sec/min]\ \times \frac{Desired\ FPS}{2\ [LED\ Passes/Ring\ Revolution]} = \ LED\ Ring\ RPM \qquad (4.1)$$

Since we will be using a ring of LEDs for our display, one rotation of the ring results in two passes of LEDs. The motors in our system ultimately determine the RPM of the Spinning Plate. We plan on using a gear ratio of 1:2, with the motor have the smaller of the two gears. So, to determine the necessary motor RPM we use the following equation:

$$\frac{Spinning\ Plate\ RPM}{2} = Motor\ RPM \qquad (4.2)$$

Again, the minimum 24 FPS. Using the two equations listed above, the following figure has been created:

Desired FPS	Desired LED Ring RPM	Necessary Motor RPM
18	540	270
24	720	360
30	900	450

Figure 4.1. Table containing necessary motor RPM based off desired LED Ring RPM

We will use a 12V +/- 10% power supply to drive the two motors. To ensure we can drive our motors at a sufficient RPM, we can test our motors' RPM using a tachometer.

4.1.3 Testing Procedure

1. Supply the motors with the maximum voltage our 12 V battery could supply: 13.2 V.
2. Measure the motors' RPM to ensure they are faster than the necessary motor RPM.
3. Supply the motors with the minimum voltage our 12 V battery could supply: 10.8 V.
4. Measure the motors' RPM to ensure they are faster than the necessary motor RPM.

5.0 Costs

5.1 Labor

Name	Hourly Rate	Total Work Hours	Total = Hourly Rate x 2.5 x Total Hours
Michael	$30	175	$13,125.00
Lunan	$30	175	$13,125.00
Jiale	$30	175	$13,125.00
Total		525	$39,375.00

5.2 Parts

Vendor	Part Number	Part Description	Cost Per Item	Quantity	Total Cost
AdaFruit	DotStar	LED Strip 144/m	$74.95	1	$74.95
AdaFruit	US5881LUA	Hall Effect Sensor	$2.00	2	$4.00
AdaFruit	SRC022A-12	12-Wire Slip Ring (max 240V @ 2A)	$19.95	2	$39.90
Pololu	2822	19:1 Metal Gearmotor with 64 CPR Encoder	$39.95	2	$79.90
AdaFruit	658	5V DC 10A Power Supply	$25.00	1	$25.00
AdaFruit	352	12V DC 5A Power Supply	$25.00	1	$25.00
Atmel	ATmega328P	Microcontroller	$4.00	1	$4.00
Amazon	BC417	KEDSUM Arduino Bluetooth Receiver/Transmitter	$10.00	2	$20.00
AdaFruit	TIP120	NPN Darlington Transistor (pack of 3)	$2.50	1	$2.50
AdaFruit	EZ-LINK	Bluetooth EZ-LINK Module	$22.5	1	$22.5
Arduino	Uno	Arduino Uno Module	$24.95	1	$24.95
				Total	$332.7

5.3 Grand Total

Expense	Total
Labor	$39,375.00
Parts	$332.7
Total	$39,707.70

6.0 Conclusion

6.1 Accomplishments

Overall, we were able meet all the requirements as specified in the verification table. At the end of the semester, our team was able to display images using 1DOF of our display at 24 FPS, with the spinning plate rotating while the LED ring remained stationary. During our Final Demonstration, we were able to display a still-image of Super Mario and the Globe. We also let Professor Kumar use our GUI to draw a simple and display a simple rectangle on our 1DOF display.

As far as 2DOF animations go, we were able to show simple shapes with periodic signs of instability. For example, some LEDs would turn on when they shouldn't be on etc. The hardware limitations (which will be mentioned in section 5.4 "Future Works") greatly increased the difficulty of displaying stable images and animations using our 2DOF display.

We were extremely proud of the work we put into our project, and we certainly hope another group will take our project and improve upon it in the Fall.

6.2 Uncertainties

Perhaps the biggest uncertainty we still have about our projects remains in the software realm. We spent two entire weeks trying to develop a method to confidently show 2DOF images and animations and were *not* too successful; however, we truly believe that, given more time, we could have developed a better way to utilize our 2DOF display.

When we look back on our semester, one of the biggest uncertainties we have today is how our project would have panned out if we only decided to create a 1DOF display. We will admit that we *completely* underestimated how difficult creating a 2DOF display would be, so we still think about how much we could have accomplished if we stuck to the development of a 1DOF display. Creating images and animations for a display with only one degree of freedom is extremely easy compared to one with two. This being said, we truly think we could have created a much more user-friendly, robust, and entertaining display had we chosen to stick with creating a 1DOF display.

6.3 Ethics and Safety

We recognize the importance of committing ourselves to high ethical and professional conduct. There are a couple of specific points in the IEEE Code of Ethics that resonate with our project and the responsibilities it gives us:

- to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment
- to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
- to avoid injuring others, their property, reputation, or employment by false or malicious action;

Our project poses some safety precautions that me we must inform the audience of our display at *all* times in the event someone who could potentially be negatively affected by our project is present. It would be extremely unethical of us to ignore this issue because the safety, health, and welfare of the public is a huge responsibility for us.

Our project poses a lot of problems in field we are not well versed in, specifically the mechanical portion of our project. With this is mind, we have been very open to seeking and accepting any "mechanical" help we can get. We would especially like to thank the people at the UIUC ECEB Machine Shop for giving us some insight as to how to keep our project mechanically sound.

We also recognize it would be unethical of us not to protect our audience from the mechanical dangers of our project. We plan to have the Machine Shop create a barrier between the display and those using/watching it. If we were not cognizant of this issue, we would not be putting forth our best effort to avoid injuring others and ourselves. Our display will be spinning upwards of 900 RPM, so mechanical danger is not something to take lightly.

6.4 Future Work

Overall, there are several changes that we can make to display much more stable and clearer image. We will discuss possible future work in three different parts.

6.4.1 Mechanical Future Work

For mechanical work, we will use stepper motor rather than DC motor for vertical rotation. A stepper motor enables precise rotation control, which can achieve more accurate RPM without using any feedback from sensors.

Currently, we are use hall sensors to determine the speed of the DC motor and adjust positions of the LED Strip. However, the RPM calculating in this way is not accurate since interrupts from the Hall Effect sensor may be blocked as discussed in the 2DOF Algorithm section.

By using a stepper motor, we no longer need to use the Hall Effect sensor to calculate the RPM and adjust the LED Strip positions. Instead, we can directly control the RPM by programming the stepper motor and use a timer to adjust the position of the LED strip, which is more accurate than the current approach.

The slip rings we used in our project contained 28 AWG wire. These wires were extremely difficult and unreliable to use with breadboards and the Arduino Uno, so we would try to obtain slip rings with 22 AWG wire the next time around. This would make the debugging/testing process a lot easier and convenient.

Lastly, we would create a new design for the entire mechanical portion of the project based around the concept of optimal weight distribution. We made the mistake of placing the motor driving the LED Ring a bit too far from the vertical axis of rotation. This created a slight imbalance in the system while it was rotating in the horizontal direction, and in the future we would try to minimize this imbalance.

6.4.2 Hardware Future Work

The limitation of data transmission speed affects the performance of the project. Since Atmega328P microcontroller has only 16MHz operating frequency, the SPI transmission frequency has only half of the microcontroller frequency which is 8MHz, so having a microcontroller like AT32UC3A3256 which is a 32-bit AVR microcontroller with 256kByte flash memory and 84 MHz operating frequency could update the LED status faster to match the higher RPM.

Since the data transmits serially in LED strip, to further lower the speed requirement for data transmission we could separate the LED strip into several segments, and utilize different ports on the microcontroller to send data to each segments. Thus the data transmission speed will be greatly boosted up.

6.4.3 Software Future Work

For software, an important improvement is to offer more user-friendly GUI so that users can create display patterns easily.

Currently, users need to set up the 80 * 40 grids by themselves, which is annoying and boring work. One possible solution is to allow users to type numbers or words in the GUI and we translate these inputs into the display pattern for them. We will handle position of the display patterns and enlarge them appropriately so that they will be displayed nicely on the spherical surface.

7.0 Citations

[1] Arduino Inc.(2015, Aug). *Arduino Uno Rev3 schematic*[Online] Available at: https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

[2] Atmel Inc. (2015, Nov). *Atmega328P datasheet* [Online] Available at: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

[3] P. Bakaus. (2014, May 21). *The Illusion of Motion* [Online]. Available: https://paulbakaus.com/tutorials/performance/the-illusion-of-motion/

[4] P.Burgess. (2013, Aug. 30). *Powering NeoPixels* [Online]. Available: https://learn.adafruit.com/adafruit-neopixel-uberguide/power

Appendix A

Requirement	Verification	Points
12 V Power Source a. V_{out} = 12 V +/- 10% b. I_{out} = 5 A +\- 10%	12 V Power Source Voltage Verification Process: 1. Connect the 12 V power source to a 2.5 Ω resistor. 2. Place a digital multimeter in parallel with the 2.5 Ω to measure the voltage difference across it. 3. Turn on the 12 V power source. 4. Ensure output voltage is within 10.8 V and 13.2 V. 12 V Power Source Current Verification Process: 1. Connect the 12 V power source to a 2.5 Ω resistor. 2. Place a digital multimeter in series with the 2.5 Ω to measure the current through it. 3. Turn on the 12 V power source. 4. Ensure output voltage is within 4.5 A and 5.5A.	5
5 V Power Source a. V_{out} = 5 V +/- 10% b. I_{out} = 10 A +\- 10%	5 V Power Source Verification Process: 1. Connect the 5 V power source to a 0.5 Ω resistor. 2. Place a digital multimeter in parallel with the 0.5 Ω to measure the voltage difference across it. 3. Turn on the 5 V power source. 4. Ensure output voltage is within 4.5V and 5.5V. 5 V Power Source Current Verification Process: 5. Connect the 5 V power source to a 0.5 Ω resistor. 6. Place a digital multimeter in series with the 0.5 Ω to measure the current through it. 7. Turn on the 5 V power source. 8. Ensure output voltage is within 9.5 A and 10.5A.	5
Motors a. V_{out} = 12 V +/- 10% b. I_{Stall} = 5 A +\- 10% c. RPM_{Max} >= 500 RPM	Motor Voltage Verification Process: 1. Connect the motor to a 2.5 Ω resistor and a 12V power source in series. 2. Place a digital multimeter in parallel with the motor to measure the voltage difference across it. 3. Turn on the 12 V power source. 4. Ensure output voltage is within 11.4V and 12.6V. Motor Current Verification Process: 1. Connect the motor to a 2.5 Ω resistor and a 12V power source in series. 2. Place a digital multimeter in series with the motor to measure the current through it. 3. Apply a torque force to the output shaft of the motor such that the shaft cannot rotate. This will force the motor to draw its Stall Current. 4. Turn on the 12 V power source.	20

	5. Ensure output current is within 4.5A and 5.5A. Maximum Rotations Per Minute (RPM): 1. Connect the motor to a 2.5 Ω resistor and a 12V power source in series. 2. Turn the power source on. The engine will be rotating at its maximum RPM. 3. Use an infrared tachometer to measure the RPM of the motor's output shaft. 4. Read the output shaft RPM and ensure it is between greater than 500 RPM.	
Hall Effect Sensors a. Produce a LOW signal that is accurate to tachometer measured RPM +/- 5%	Hall Effect Sensor LOW Signal Frequency: 1. Put 5V across the VCC and GND pins of the Hall Effect Sensor 2. Attach the output pin of the sensor to an oscilloscope 3. Ensure there is no south-pole magnetic field near the sensor and measure the output voltage of the sensor using the oscilloscope. It should be 5V. 4. Attach a magnet to a motor with the south pole facing radially outwards. 5. Run 12V across the motor and measure the output shaft with a tachometer (following the steps in the Motors section of the verification table). 6. Put the rotating magnet, which is attached to the motor shaft, within the sensor's detection range and measure the frequency of the LOW signal the sensor outputs using the oscilloscope. 7. The frequency of the LOW signal of the Hall Sensor should be within 1% of the tachometer-measured RPM.	10

Supplying Power via Slip Rings a. Transfer a voltage to the rotating wires of 5 V +/- 5% of the voltage supplied to the stationary wires of the slip ring. b. Transfer a current to the rotating wires of 2 A +/- 5% of the current supplied to the stationary wires of the slip ring.	Voltage Transfer: 1. Put 5V across two wires connected to the stationary portion of the slip ring. 2. Measure the voltage across the two corresponding wires attached to the rotating portion of the slip ring using a multimeter. 3. Ensure the measured voltage is within 4.75 V and 5.25 V Current Transfer: 1. Run 2 A across two wires connected to the stationary portion of the slip ring. 2. Measure the voltage across the two corresponding wires attached to the rotating portion of the slip ring using a multimeter. 3. Ensure the measured voltage is within 1.9 A and 2.1 A.	10
LED Strip a. Data package must be 32bits each. 31th - 29th bits must all be 1. 28th - 24th bits are brightness settings. The rest 24 bits are divided into three consecutive 8-bit chunk, representing BLUE, GREEN, and RED from right to left.	LED Strip Data Verification Process 1. Connect the microcontroller to your personal computer via USB 2. Run program to enable microcontroller and start to send data packages(32 bits) to computer 3. Run setBrightness() to change the brightness of the LED. 4. Ensure the LED strip looks brighter when setting a larger 5-bit binary. 5. Run setColor() to change the color of the LED. 6. Ensure the LED strip looks red when 7th-0th bits are set to 1 and 23th - 8th bits are set 0. 7. Ensure the LED strip looks green when 15th-8th bits are set to 1 and 23th - 16th bits and 7th -0th bits are set 0. 8. Ensure the LED strip looks red when 23th-16th bits are set to 1 and 15th - 0th bits are set 0.	10
Microcontroller a. Speed of Data transmission must be no less than 1.6 Mbit/s	Microcontroller Speed Verification Process: 1. Connect the microcontroller to your personal computer via USB 2. Run program to enable microcontroller and start to send data packages(32 bits) to computer using SPI (Serial Peripheral Interface) 3. Run countPackage() to counter the number of data packages that your computer received in one sec 4. Ensure your computer receive no fewer than 52800 packages	10

Microcontroller I/O a. Upload program to microcontroller through Bluetooth Module a. Microcontroller command LED strip to display certain pattern. b. Microcontroller control Vce of TIP120 transistor	Bluetooth I/O Verification: 1. Connect bluetooth RXD port to the PD1 (TXD port of microcontroller) and bluetooth TXD port to the PD0 (RXD port of microcontroller), Vin to 5V power line, GND and DSR to power GND. TIP transistor base connect to PD5 , 2. Use Oscilloscope to monitor the signals between the microcontroller 3. Use bluetooth function on computer to pair with the bluetooth module 4. Send texts through bluetooth communication 5. Ensure there are signals on the pins while computer sending texts to the bluetooth module, and the bluetooth module response the text accordingly.[2]	5
	LED strip I/O Verification: 1. LED strip data port connect to microcontroller PB3 pin, clk port to microcontroller PB5 pin. 2. Connect Arduino to computer and upload test program “Blink” (a program sending commands to LED to let it blink) to Arduino. 3. Run the test program and see if the LED strip blinks accordingly.[2]	5
	TIP transistor I/O Verification: 1. Connect base pin of TIP transistor to PD5 port (PWM port of microcontroller) on microcontroller. 2. Connect Arduino to computer and send PWM signal to the microcontroller to control the rotation speed of the motor through the voltage change. 3. Along with the hall effect sensor connected to the microcontroller PB0 pin, monitor the rotation speed change on the serial monitor.[2]	5
GUI a. Display the spherical surface with left one-third red, middle one-third Green and right one-third Blue.	GUI Verification Process 1. Opening GUI.exe on a Windows PC. 2. Entering 1 to 133 in row, and 1 to 88 in column, and setting the color to red. 3. Entering 134 to 236 in row, and 1 to 88 in column, and setting the color to green. 4. Entering 237 to 400 in row, and 1 to 88 in column, and setting the color to blue. 5. Clicking RUN button and ensuring the spherical surface is RED-GREEN-BLUE from left to right and each color takes almost one-third of the spherical surface.	10

Appendix B

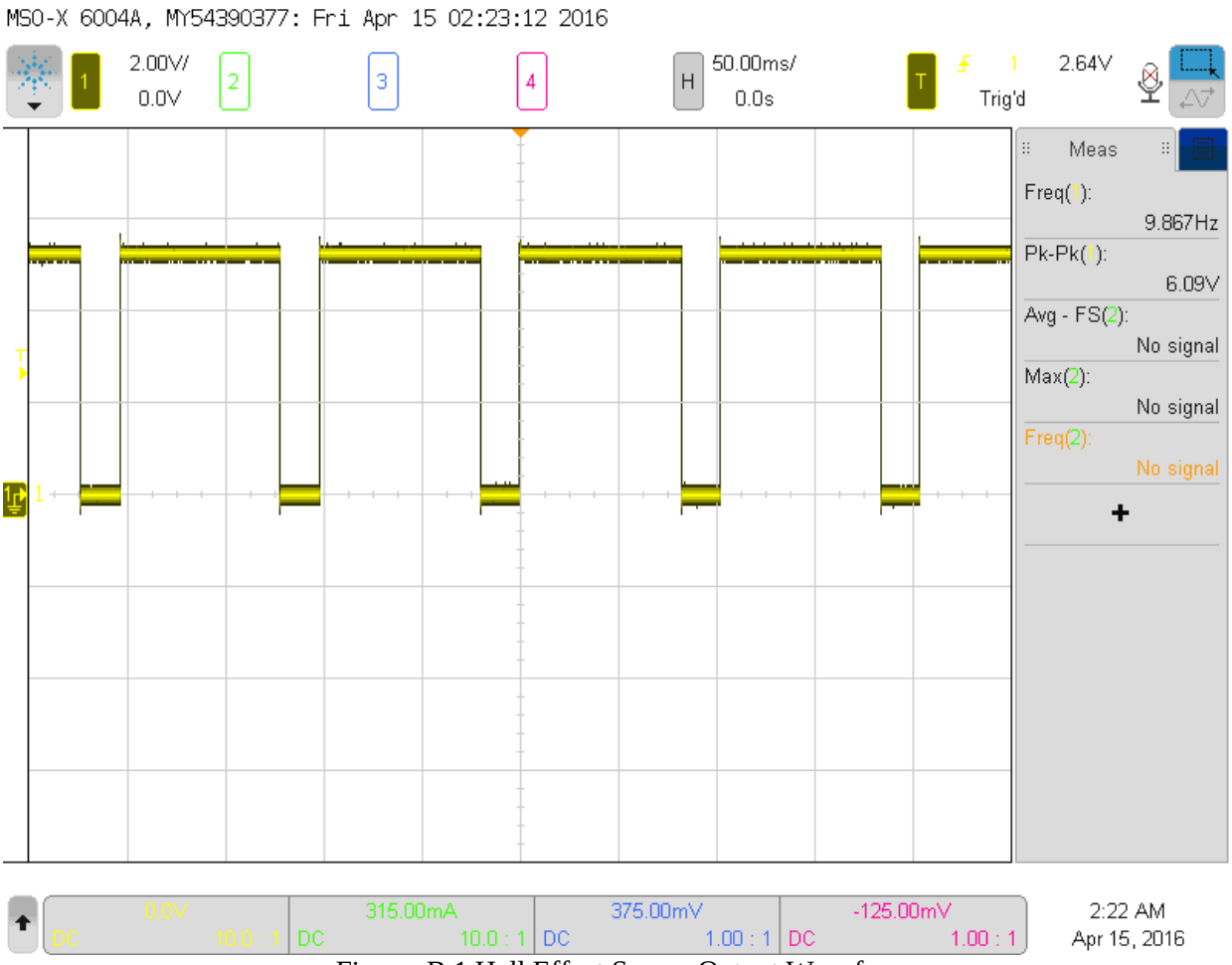

Figure. B.1 Hall Effect Sensor Output Waveform

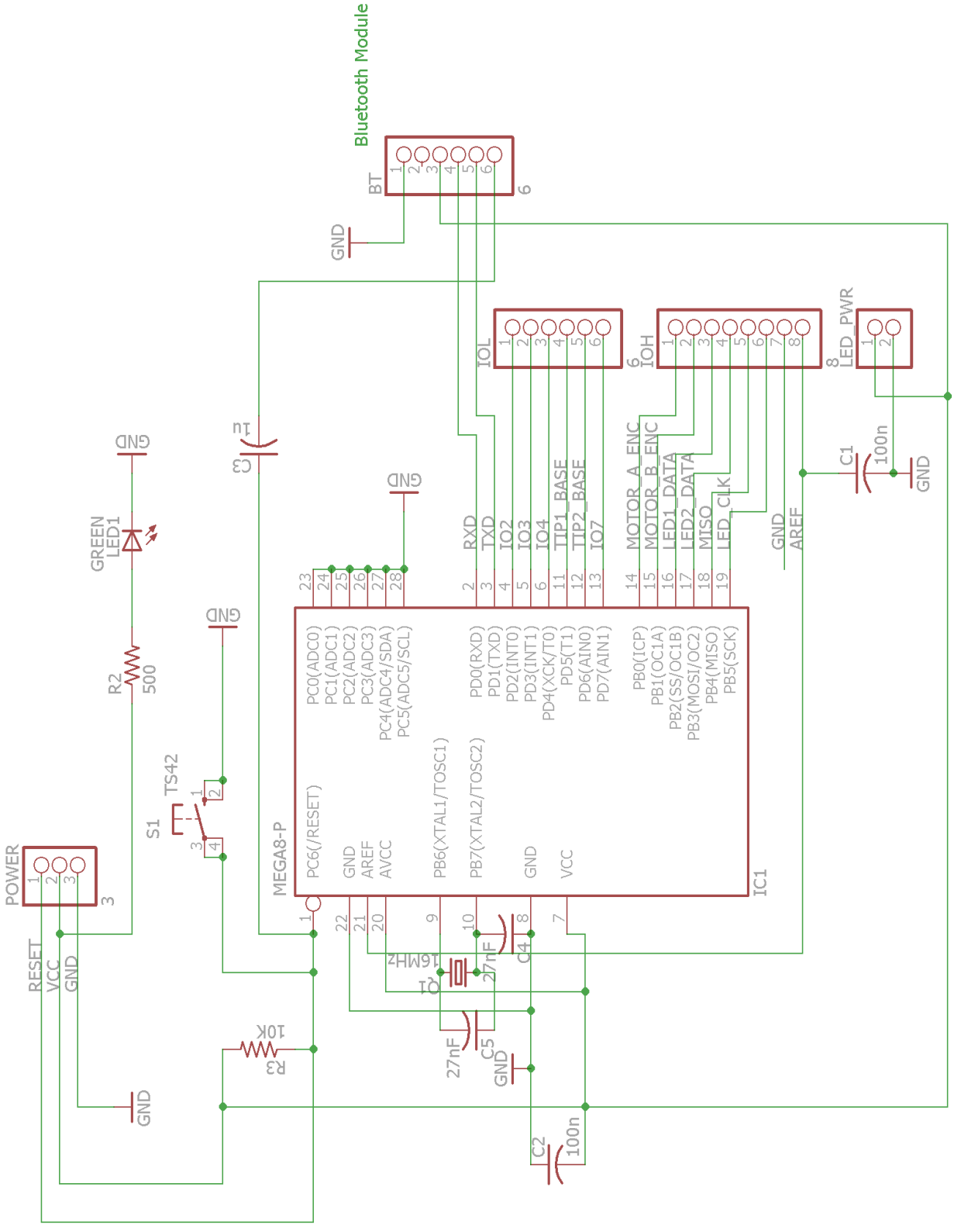

Figure. B.2: Schematic of Microcontroller PCB

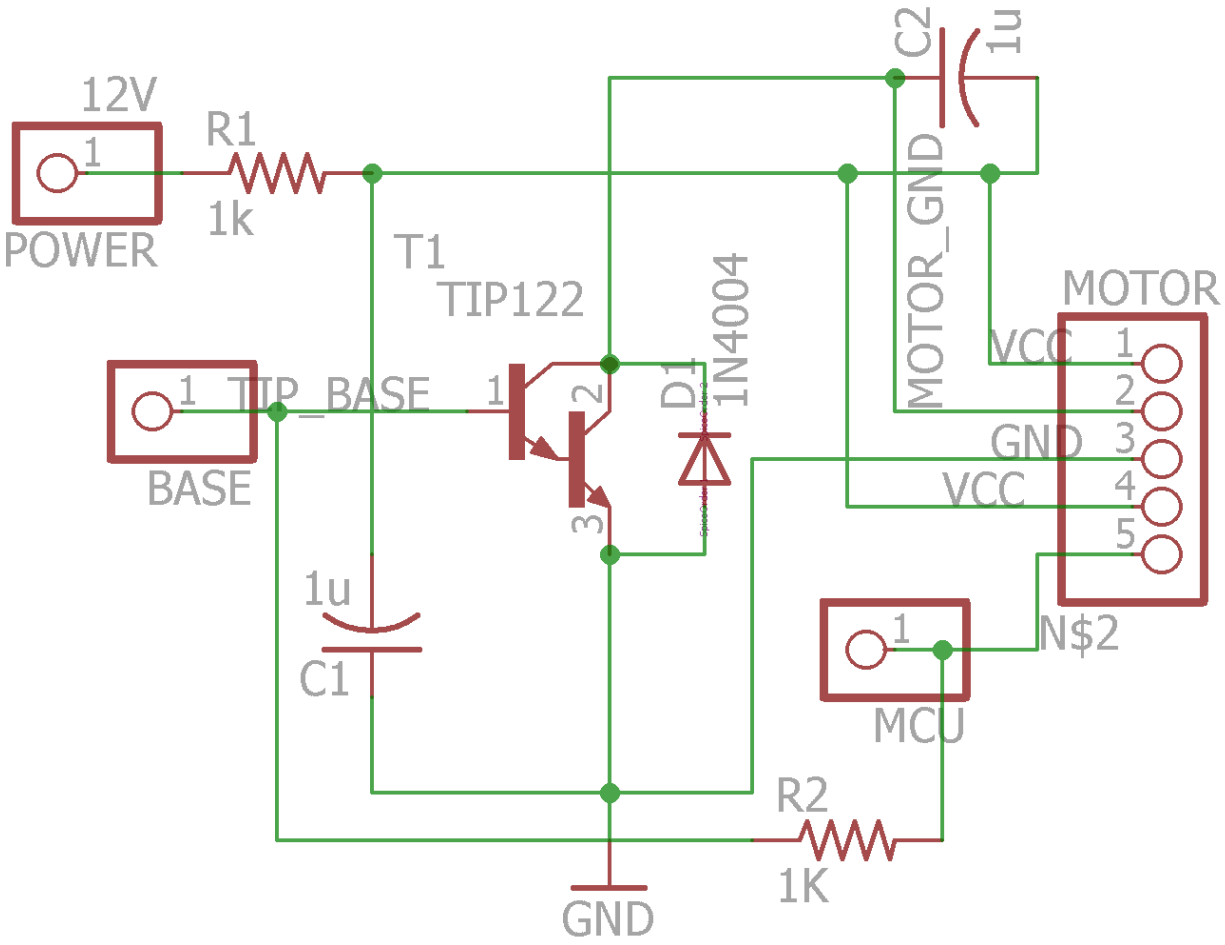

Figure. B.3: Schematic of TIP120 PCB