3D Persistence of Vision Device

Aaron Burlison, Antonio Ortiz III,
Timothy Egan, Patrick Srofe

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Persistence of vision is widely used as a method of image recreation and distribution. While generally done through the use of various monitors or touch screens available on the market there are some other methods using motors and light emitting diodes that have been used to perform the same function. The objective of this paper is to present a design of such a device at a much grander scale, including higher resolutions and wireless image processing.**

***Index Terms* — AC-DC Converters, Image processing, Image coding, Image generation, Light emitting diodes, Permanent magnet motors, Visual effects.**

I. Introduction

Persistence of vision is a phenomenon that has motivated engineers for years to create a variety of inventions. This has motivated engineers for years to create a variety of inventions. This has not changed even to this day. There are still devices using this visual trick being constructed with a wealth of internet examples available to show for it.

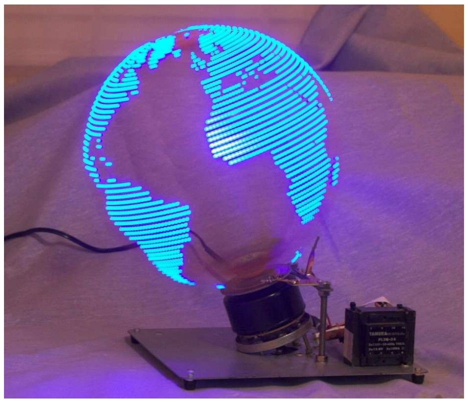

Fig. 1.1. Persistence of vision globe.

These spinning devices that utilize LEDS to create the illusion of one solid image come in a variety of shapes and sizes from spheres and discs, to cylinders.

A. Frame Rate

Since human vision is tricked to perceive motion around the rate of twenty-five frames per second we needed a device to spin at a rate capable of recreating this illusion. Since twenty-five was the bare minimum we decided to overshoot to thirty frames per second, this would hopefully either make a more seamless image or account for any variations that may occur within the device.

B. Computer Interfacing

We designed the device with the intent of being able to connect directly to any computer and receive images to process and display. This is entirely being done through an Ad Hoc wireless network.

D. Portability

Since we had decided the device needed to be portable it could be no heavier than a small television and only about as bulky. This meant the materials we chose to build this device out of need to be durable and light weight.

A. Programmability

The device also needed to be easily programmable and capable of at least simple marquee text displays that would be implemented with our own self developed program. This would allow the user to simply input a text banner or the time, and have it displayed on the device instead of just the computer interface. In addition, we also wanted the device to be complex enough that someone with experience in programming could also program the processor and create custom images and animations. This would allow for a lot of space for user development.

II. Chassis design

The chassis is the main support structure that will be used to mount the motor, the LED array support frame and the motor controller. The chassis will also allow for the motor to integrate with the LED array support frame through a bearing to allow the frame to be driven by the motor.

A. LED Support Frame

Before we were able to finalize the chassis design, some critical dimensions had to be determined. The first dimension that needed to be determined was the size of the LED array support frame. The LEDs chosen for this project had a vertical dimension of 2.8mm. Using 1mm of spacing between LEDs resulted in each LED requiring 2.85mm of space. Multiplying the number of LEDs by the space requirement per LED resulted in an overall vertical dimension of 364.8mm or 14.362 inches. Using the idea that the desired look of the POV display will be a cylinder, we then calculated the distance the primary LED array will need to be from the center of rotation, which was equal to 6.97 inches or approximately 7 inches. The secondary LED array will need to appear to stand off from the primary LED array. To achieve this appearance we offset the secondary LED array from the primary by 2 inches. This resulted in the longest piece of the LED array support frame, as measured from the center of rotation, to be approximately 9 inches. The overall dimensions of the LED array support frame are 15 inches by 18 inches. In order to reduce the weight and torque required to spin the LED support frame, we constructed the frame out of carbon fiber tubing with aluminum gussets at the corners. The gussets increased the strength of the frame while it is spinning and help provide a rigid frame for the LED arrays. We chose to use a square carbon fiber tube with 0.315 inch sides.

B. Chassis Base Dimensions and Assembly

Once we had determined the longest piece of the LED array support frame that will be rotating, we were able to determine the size of the chassis base. For safety reasons, we decided to make the chassis base large enough to allow the LED array support frame to spin inside the foot print of the chassis base or a square base with 18 inch sides. We decided to construct the chassis base from two 1/4 inch aluminum plates. The two plates were offset by 12 inch solid aluminum rods and bolted together. The aluminum plate and rods provided strength while minimizing the weight of the chassis. The 12 inch rods allowed the chassis base to be open in the middle for mounting the motor and the motor controller. The overall dimensions for the chassis base are 18 inches wide by 18 inches long by 12.5 inches high.

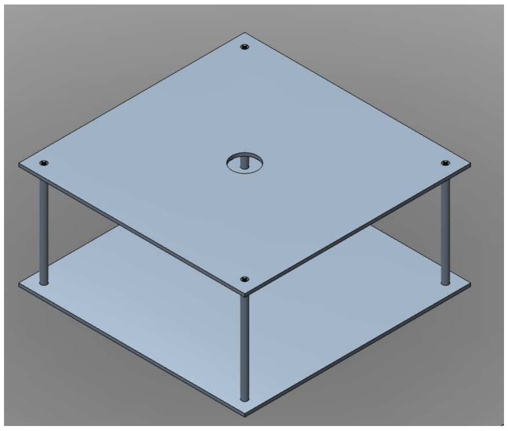

Fig. 2.1. Model of the chassis base showing the assembly of the base plates and rods.

C. Motor and LED Array Support Frame Rotating Interface

To allow for the LED array support frame to be driven by the motor, we decided mount an extended-ring bearing in the center of the top plate of the chassis base. In order to provide the most space for feeding the power supply cable through the rotating interface, we selected an extended-ring bearing with a one inch shaft diameter. We secured the bearing to the base of the chassis by welding a 2 inch, schedule 40 pipe to the top plate with four set screws to

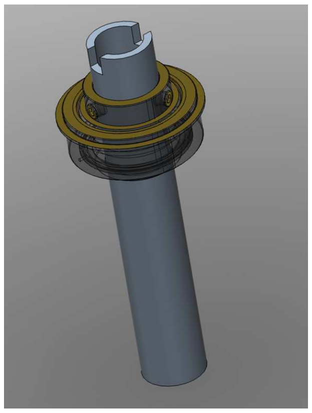

Fig. 2.2. Model of the extended-ring bearing and support tube for mounting the LED array support frame.

secure the bearing to the pipe. To secure the LED array support frame to the bearing, we inserted an aluminum tube through the inner ring of the bearing. The tube was secured to the bearing using the two set screws that come installed on the bearing. This allowed for the POV display to be easily disassembled when moving between locations. Also, this design allowed the tube to be used to mount a slip ring for electrical power transfer. The tube was then attached to the shaft of the motor using additional set screws. Lastly, the tube was notched to 0.315 inches wide on the top to allow the LED array support frame to be secured to the tube.

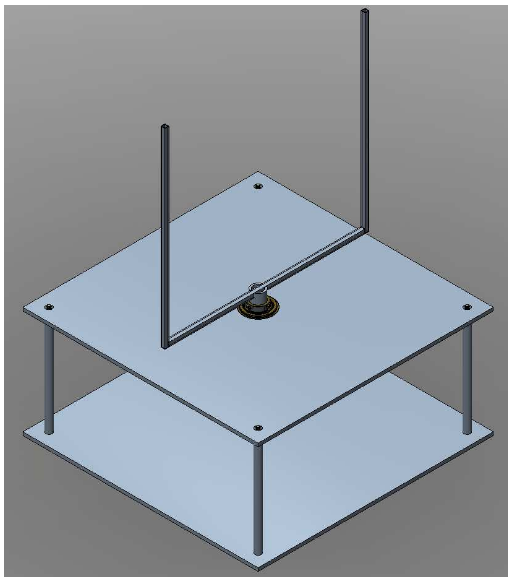

Fig. 2.3. Model of the chassis assembled with the LED array support frame.

III. PRIMARY LED ARRAY

The primary LED array is used to display the main image of the POV display. The primary LED array consists of 128 LEDs and 24 LED controllers. The LEDs chosen for the primary LED array are surface mount RGB LEDs with

A. Primary Array LEDs

The LEDs chosen for the primary LED array are surface mount, super bright, RGB LEDs. As previously discussed, the LEDs are only 2.8mm wide. This allowed us to achieve a high density of LEDs and therefore increasing the resolution of the display. The final design of the primary LED array allowed for 16 LEDs to fit in approximately 1.85 inches. Lastly, the RGB LEDs in each LED chip can be individually driven allowing use to easy integrate the LEDs with the LED controllers.

B. Primary Array LED Controllers

The LED controllers chosen for the primary LED array are the TLC5940 by Texas Instruments. The TLC5940 allowed for controlling 16 individual LEDs through 16 individual controlled pulse width modulated channels. Each channel is capable of 4096 grayscale steps of the pulse width channel allowing us to achieve a wide range of RGB colors. Additionally, the communications of the controllers can be wired in series allowing us to write to each controller using only one output line from the primary microcontroller. Essentially, the output of one controller is the input of another controller. Lastly, the controllers have an enable line allowing us to latch all the controllers at the same time. This allows us to flash the controllers at the same time to generate the image display.

C. Primary Array Circuit Boards

The primary LEDs and controllers required mounting on printed circuit boards. We chose to break the 128 RGB LEDs into 8 individual boards. Each board houses 16 RGB LEDs and 3 LED controllers. This allowed us to have the flexibility to change the size of the primary LED array or replace a string of LEDs if required. Each board has terminal blocks allowing it to be wired in parallel for power and common inputs, and in series for communications. The outputs of the LED controllers are wired in sequential order to each RGB LED. Outputs 0, 1 and 2 of the first LED controller are wired to the inputs of the first RGB LED. The pattern continues until the last RGB LED.

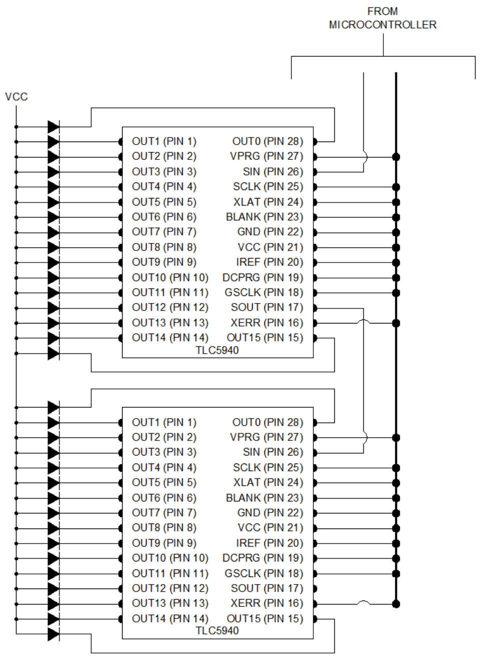

Fig. 3.1. Example wiring scheme of LEDs and LED Controller

IV. SECONDARY LED ARRAY

The secondary LED array is used to display a text image that appears to stand off from the image being displayed by the primary LED array. The design for the secondary LED array follows closely to the design for the primary LED array. The changes that we made for the secondary LED include using green LEDs instead of the RGB LEDs. Also, the size of the secondary LED array is smaller than the size of the primary LED array. The secondary LED array consists only of 16 green LEDs. This also resulted in only requiring one LED controller. The secondary LED array still maintains to module design of the primary LED array allowing us the flexibility of adding or replacement LED modules as required.

V. Computer Side Image Processing

We will be accepting image input from users via computer side GUI. Image processing functions will then extract then determine the image file type, and extract the RGB data from the image. The RGB data will then go through various forms of processing including color depth reduction, cropping, and padding. The image may also be top justified, bottom justified, or centered. The RGB data will also be rearranged and packed into a binary file which will then be sent to the POV display. The rearranged data will be stored in the same order that it will be used by the PIC32, which is an effort to reduce the required processing power.

A. Image Format

We wanted to accept any reasonable image file format that the user might input. This includes .jpg, .bmp, .png, and many more. In order to obtain the RGB data from these various image formats, we used several java functions in the standard library. The first function is ImageIO, which is supplied an image file format. ImageIO then returns a Buffered Image, from which RGB values can be extracted by calling Buffered Image.getRGB with specific x and y coordinate as arguments. This extracted RGB value is an integer in sRGB colospace, which can be used as an argument to create a new java color, from which the RGB data can be extracted individually using Color.getR, color.getG, and color.getB, each having a value between 0 and 255.

B. Color Depth

The maximum color depth that we can obtain from an image is 24 bit color, which is far higher than what our display needs. We can convert this RGB data into 8 bit color by only using the 3 most significant bits of red and green, and 2 most significant of blue, since the human eye is less attuned to changes in blue.

C. Image Editing

If the image does not exactly match the required dimensions of our display, it will require cropping and padding since our output format is fixed in size. This is accomplished by adding a padding/cropping value to each index being used to obtain data from the buffered image. If this altered index is out of range, the attempt to obtain the pixel will fail and a predefined black or white pixel will be written to the output file. There is a padding/cropping value for both the X and Y index. These two values can be determined by comparing the Height and Width of the buffered image with the predefined Height and Width of our display.

C. Image Storage

The RGB data will be stored into a binary output file in the order that the PIC32 will use it. Since the display shows a single column of the image at a time, we will iterate through the image column by column and output the RGB data as we arrive at each pixel. We will transverse the image from top down first, then left to right.

VI. Outputting data to the LED Array

The PIC32 contains two timers which can be set to specific frequencies. The first timer, Timer 2, will be set to operate at 20 Mhz. The second timer, Timer 3, will be set to operate at 9600 Hz. The PIC32 also has 5 output

Fig. 6.1. ChipKIT uc32 Micro Processor

compare modules which can be used to pulse their associated pins, and generate interrupts. An output compare module can use either Timer 2 or Timer 3 when configuring it, and the pulse width and behavior can be controlled by selecting from various operating modes.

The Grayscale clock will use Output Compare Module 1 and pulse on pin 3 at 20 MHz. Modules 2 and 4 will be configured using timer 3 and used to generate control signals from Blank and XLAT. The blank signal will fire first, driving pin 6 High. While pin 6 is high, the XLAT pulse will fire and drive pin 6 High. Blank will go low after XLAT has gone low, since it has a wider pulse width.

The LED controllers contain a 192 bit shift register to which the Grayscale data is written via serial transfer. We are attempting a few different ways of doing this. The first method, via SPI interface, has been successful, but has created conflicts with other devices sharing the SPI data bus.

A. LED Driver Communication

Another scheme for transmitting the serial data involves setting up an output compare module on pin 5 to act as the SCLK. The falling edge of each pulse creates and interrupt which sets a flag allowing a function to change the SOUT pin to high or low depending on the current bit of data.

We are using a TLC5940 library which allows us to manipulate the Grayscale Data Buffer in various ways, and sets up a scheme in which to update the data in the controller via SPI communication. This library has become a template for our own software design, but with many changes. This includes altering the Output Compare Module code to move the blank signal off of pin 10 and onto pin 6, and also using a different scheme to alter the GS data buffer, by reading the data from the SD memory and storing it directly into the buffer. Also, we will need to add a means for the library to either share the hardware SPI or to not use it at all.

The control program will initialize the TLC5940 library and begin reading data stored in SD memory. For each iteration of the control program, the Grayscale Data buffer is altered, and then an update function is called which writes the new data to the LED controllers. The control program then checks for incoming date via wireless communication. The BLANK and LAT signals will eventually go high, and the new data will be latched in and displayed.

The data is being read from SD memory one column at a time, these bytes are then combined to create an unsigned integer, and then stored in the grayscale buffer. The grayscale buffer size is exactly one column of data, that being 6 unsigned integers times the number of LED controllers in series. This corresponds to 192 bits per controller.

B. LED Text Array Communication

A second LED array used for displaying text will be connected to the primary LED array's SOUT and receive the same SCLK signal that the primary array receives, effectively extending the main array by one more LED controller. The user may want to only display text and without an image, or an image without text, or both. Because of this we will have various modes of operation which help to determine what data to write into the grayscale data buffer. If either is disabled, the data for that display must still be written as all zeros so that the shift registers that the user specified read from SD will be in the proper position.

C. Scrolling Text

Image and text files stored on the micro SD card all start with a one byte header which specifies whether or not the image should rotate left, right, or stand still. The speed of this rotation can also be set. This header is created by the GUI based on user input. In order to cause an image to scroll left or right as it is being displayed, we first save a pointer to the start of the image. A pointer used to transverse the image is then incremented as it is output, if it increments beyond horizontal resolution of our display, the pointer is reset to zero. The starting position of the transversal pointer is adjusted based on the speed value stored in the header file.

D. Internal Storage and External Communication

We will also occasionally check for communication from the computer connected to the device. These messages could be telling the device to turn on or off the main display, or to switch which image to display that has already been loaded onto the SD memory storage. Additionally, new data for the text array can also be received so that the text displayed updates to the new message. At this time, we are not sure if we will be able to receive live updates of images not already stored in SD memory because of the amount of time it would take to receive an image in entirety. It may be possible to receive the image in smaller packets and write them to a file in SD memory as they are received, all while still displaying an image on the primary array uninterrupted. If this proves infeasible we could enter into a data transmission mode, where image displaying ceases and the controller is dedicated to receiving and processing the data. The controller would then enter back into display mode, and display the newly received image.

VII. Wireless Communications

We will be using ad-hoc mode Wi-Fi communication in order to send data to be stored on the SD card. The microcontroller will use the attached Wi-Fi shield to act as a server and create an ad-hoc network with the SSID "POV Display". This is the network that will allow us to send various signals to the microcontroller while it is rotating and displaying images. We will be using 104-bit WEP encryption in order to secure the network and prevent unwanted connections from vandalizing the display.

A. Data to be Received

After the microcontroller is powered on and the network is created it will wait for an incoming connection. It will then be possible to connect a computer to the "POV Display" network using the required 104-bit WEP key. Once a computer is connected to the network it will then be possible to use the GUI application to send properly formatted image data to be saved on the SD card and later displayed. The data that is received will have a header to differentiate the types of data that the microcontroller may receive. Depending on the contents of the header the microcontroller will know where to store the image file since there will be separate files for the main display and the smaller text display. Any possible animation settings contained in the header will not be saved on the SD card but instead on the flash memory of the microcontroller. Any other information that the microcontroller receives that is not image data will be used to manipulate variables in the flash memory that will control the operation of the display.

A. Server Operation

The server software running on the microcontroller will use a state machine apporach. There will be a global variable to store the current state of the server. This is an efficient way of having the microcontroller decide what should be done (if anything) during each iteration of the main loop. Different possible states will include: initialize, listen, isLisening, availableClient, acceptClient, read, write, and close. The initialize state will ititialize the IP stack and begin the listen state. The listen state will have the server begin waiting for a connection on a predetermined port number. The server will continue lisening until a client attempts a connection which will then enter the avialableClient and acceptClient states in order. After the client is accepted the read state will wait until data is sent. All data will then be received and stored and then the connection will be terminated within the close state. If any information needs to be sent back to the computer this will happen within in the write state before the close state. The write state will most likely be primarily used for debugging purposes since information will not normally be sent back to the computer.

C. Client Operation

The computer running the GUI application will act as the client in the "POV Display" network. The connection to the network will need to be made manually using the wireless network manager provided by the operating system. Once the connection is established the GUI application will be required to connect to the proper port number and send meaningful information to the display. The GUI will handle all image formatting as detailed previously and send the data to the server. The connection will be terminated after each successful transfer and reestablished for each new transfer. This will allow the user to take as much time as they need between transfers without having to worry about timeouts and manually reconnecting.

VIII. Display Sensor

In order to help the process of creating a clear image a trip sensor is needed to determine an exact point in the LED array's trajectory to begin a new frame. The method we chose to implement is an infrared sensor that will trigger when passing over a reflective surface on the top of the Chassis. The total Sensor encompasses two circuits, a sending circuit and a receiving circuit. Both circuits will be on the same chip and mounted on the bottom of the spinning apparatus. The sensors will point towards the top of the housing chassis where it will rotate along a trajectory. Most of its trajectory will be covered in black paint but at one point along the trajectory a reflective surface will break up the path in order to create a trigger point. This point will be where the device will be triggered to start displaying a new frame.

A. Sending Circuit

The sending circuit, as seen in figure XYZ.1, is using an LM358 operational amplifier to implement. In this case a 5 volts Vin signal is required to run the circuit. The CTRL line is a signal coming from the rotating microcontroller, when this signal reads 2.5 volts the output will go high and turn the infrared LED on.

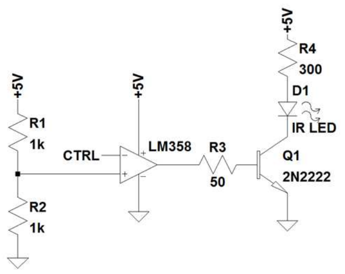

Fig. 8.1. Infrared Sending Circuit

A. Receiving Circuit

The receiving circuit will receive a 5 volts signal just like the sending signal. Both IR LEDs will be adjacent to each other and use a common property among LEDS in which when subjected to light a voltage drop can be read on the terminals of the LED. In this case when the receiving signal is hit by the reflected light from the sending signal it will create a voltage drop along its leads which will cause the LM358 on the receiving side to read high, turning on the indicator LED and sending a signal to the output terminal which will be connected to the microcontroller.

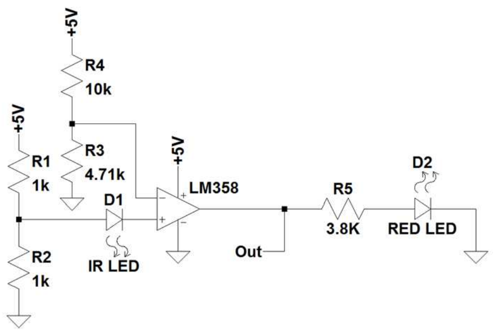

Fig. 8.2. Infrared Receiving Circuit

IX. Motor

A heavy duty motor is required to obtain the torque to rotate the LED apparatus at 1800 rpms or 30 frames per second. Using the two masses method to estimate the amount of torque that is required came out to be about 0.248 Nm. We needed our motor to be well above that to conceivably rotate the device at the speed we desired. It also needed to be large enough so that we could mount a slip ring and the apparatus onto its shaft.

A. Motor Choice

Our choice was the Dayton 9FHD7 permanent magnet motor. The motor is relatively light weight, but more

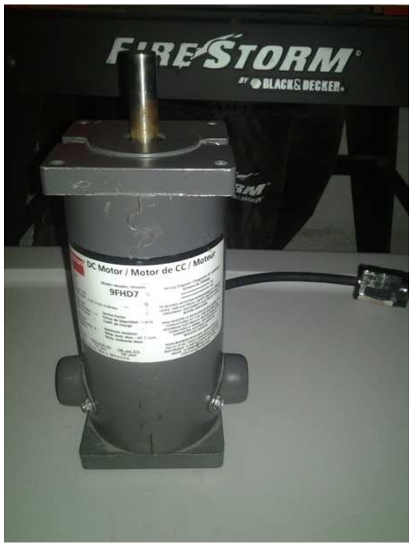

Fig. 9.1. Dayton 9FHD7 Motor

importantly is rated for 1800 rpm, and 0.49 Nm which is almost double what we need. It has a 0.5 in diameter shaft with 1.38 in shaft length, which is suitable for power transmission via slip ring from the stationary side to the rotating side. It is rated at 90 volts and 1.5 amps.

B. Motor Control

Since we need to both power this motor and have some control over its speed it would be ideal to get a driver chip

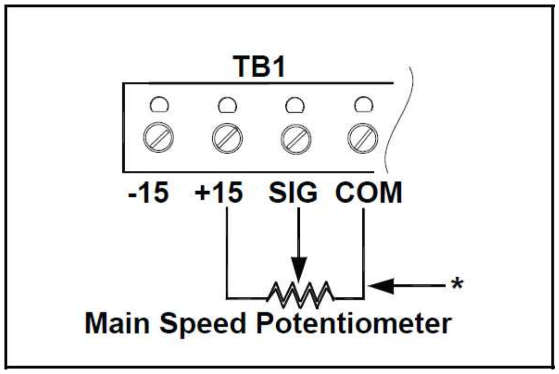

Fig. 9.2. Main Speed Potentiometer

for it. The one we found to use for this project is the KBRG-212D Regenerative Drive chip. This chip has a variety of features that are useful for our project and for possible scalability. However, for the purposes of our project the chip will be set up just for running and maintaining the rpm value of the motor.

There are two sections of the motor driver that are important to us and that is the +15 volts input, the SIG line and the COM. A 5 kilo-ohms mechanical potentiometer will be wired into these inputs in order to vary the speed of the motor. The speed of the motor is directly related to the amount of voltage that is seen at this terminal. Figure XYZ.2 shows the connection required for this. This set up only allows for single direction rotation, with the potentiometer only controlling how fast or slow the motor is rotating. During normal operations this will be turned to the maximum speed.

The second section of the motor driver of importance is the enable line. This allows for an on and off switch to the motor. There are two settings for the braking mechanism this feature uses but for our purposes we are going to use the coast to stop feature since we do not require a lot of torque to stop the device. This portion of the circuit will be connected to a switch that will be normally on and when the switch is thrown it will cause the motor to slow down to a stop, after which flipping the switch again will rev the motor back up to the speed correlating with its current setting via the potentiometer. Figure XYZ.3 shows the connections for this.

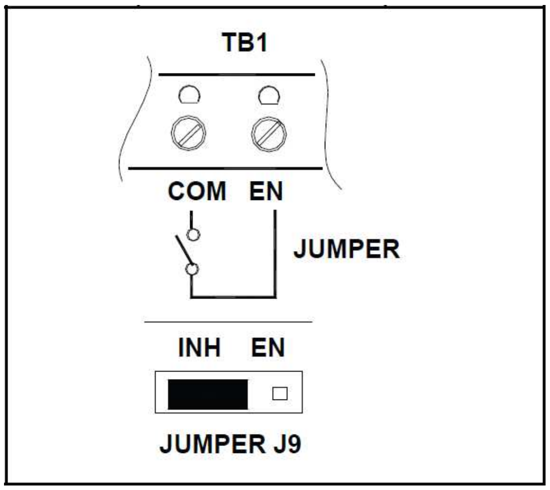

Fig. 9.3. Enable Line

X. Conclusion

The design described above was the best solution found for the specifications of the project. The product is relatively light weight and portable allowing for it to be transported from one location to another for ease of use. The device is capable of displaying a variety of images both preprogramed and custom, allowing for wide user specified applications. With further time and resources the device is openly capable of scalability ranging from a more powerful microcontroller to additional LEDs.

Acknowledgement

The authors wish to acknowledge KEMCO for sponsoring this project by donating the chassis.

Biography

Patrick Logan Srofe is currently a senior at the University of Central Florida and will receive his Bachelor's of Science in Electrical Engineering in December of 2012. His primary interests lie in embedded systems, semiconductor fabrication, and biomedical nanotechnology.

Aaron Burlison is currently a senior at the University of Central Florida and will receive his Bachelor's of Science in Electrical Engineering in December of 2012. He is currently working as a project manager at Kemco.

Timothy Egan will graduate from the University of Central Florida with a Bachelor's of Science in Computer Engineering in December of 2012.

Antonio Ortiz III will graduate from the University of Central Florida with a Bachelor's of Science in Computer Engineering in December of 2012.

References

[1] "IKA-TACH." IKALOGIC.N.P., n.d. Web. 01 Aug. 2012. http://www.ikalogic.com/ika-tach/.

[2] "99000 RPM Contact-Less Digital Tachometer." IKALOGIC. N.p., n.d. Web. 01 Aug. 2012 http://www.ikalogic.com/99-000-rpm-contact-less-digital-tachometer/.

[3] "Infra-Red Proximity Sensor Part 1." IKALOGIC. N.p., n.d. Web. 01 Aug. 2012 http://ikalogic.cluster006.ovh.net/infra-red-proximity-sensor-part-1/>.