Andrew Eldred

SENG 5831 – Final Project Report

# Summary

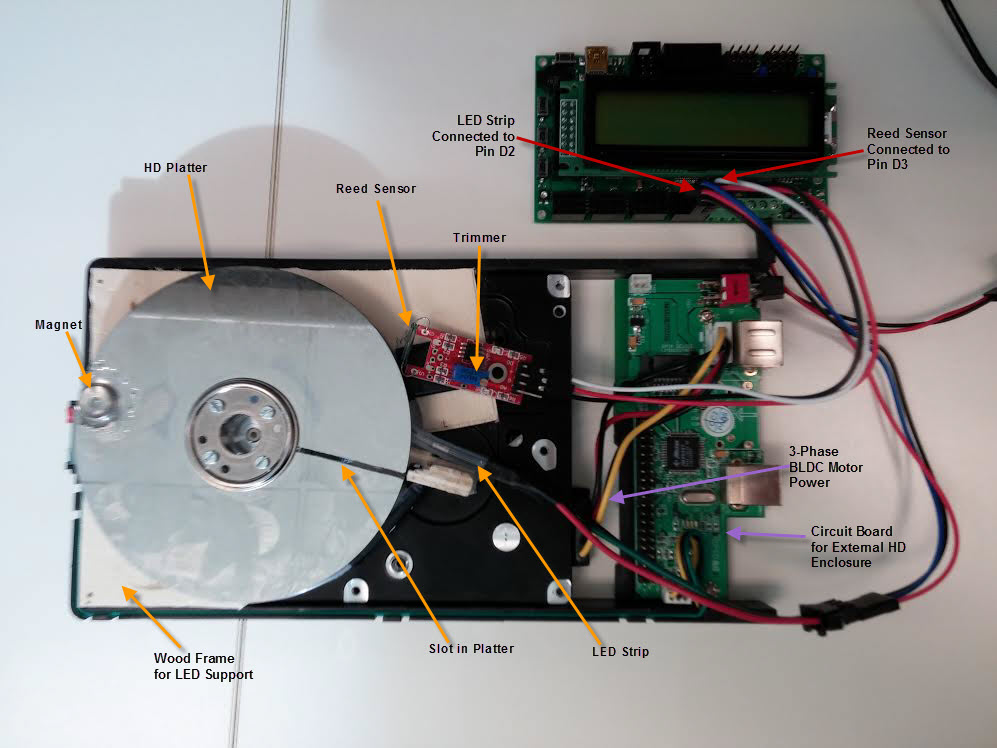
My initial project idea was a digital communications center that would allow family members to communicate with each other using a sort of digital “whiteboard”. The project as I first envisioned it would consist of two main components – a screen where messages could be displayed and a screen showing where family members are currently located (e.g. school, work, driving, etc.). Messages, user-defined locations, and GPS would be managed via a mobile application installed on each family member’s mobile device. The mobile application would then communicate with the digital whiteboard to update the displayed information.

After some research, I decided to use the idea of Persistence of Vision (POV) to create the location display because I wanted to learn more about it and the display could also then be switched to display a clock or other patterns. Given the complexity of the initial project idea, time limitations, and my general interest in the ideas behind POV, I decided to focus my efforts on the location display portion of the project.

The end product was further reduced in scope to simply getting the platter to spin, the sensor working, and the interrupts firing so that a pattern could be displayed. While I was able to achieve this result (sort of), I found that the hard drive I am using actually spins too fast so that instead of seeing the colors of the LEDs, you just “see through” the platter to the interior of the hardware setup. I played around with increasing the number of revolutions before changing the colors which helped some, although the display was still not great. I was better able to see the pattern if I slowed the disk by placing my finger on it, so I think if I could reduce the speed of the motor I would have better results.

# Implementation Overview

Hardware



* Orangutan SVP 1284p microcontroller

The “brains” of the system that controls the timing of the LEDs.

* Hard drive motor, chassis, platters, and spacing collars

These pieces form the main hardware component where the image is displayed using POV concepts. The individual parts were collected from multiple hard drives to find the right combination that would spin consistently at an adequate speed and provide a working base for the LEDs. Power for the hard drive motor was provided using an external hard drive enclosure as the motor is a 3-phase Brushless DC motor that is otherwise difficult to spin without spending a significant amount of time on it. The chassis was modified by attaching would blocks to support the LEDs and painting the inside surface white to better reflect the light from the LEDs. The platter was altered by cutting a slot perpendicular to arc of the disc, which extended from the outside edge to within about 3/8 inch of the inside edge. A magnet was also attached to the platter to provide the magnetic field for the reed sensor.

* Flexible RGB LED strip

Attached beneath the spinning platter, the light from the LEDs shines through the slot in the platter to create patterns.

* IR emitter/detector - Hall Effect sensor - Reed sensor

The sensor is used to determine when the platter has turned a full revolution. This information is then used by the interrupts to determine when to change the color of the LED strip. I first tried using an IR emitter/detector setup, but found that the communication between the two was not consistent (bad components maybe?). I then tried a few variations of hall-effect sensors, but they didn’t seem to respond fast enough. I finally settled on a reed sensor circuit with a trimmer. I played around with the trimmer to try and get more accurate readings from the reed sensor, though I’m not sure if it much effect on the results.

Software

* main.c

This is the primary driver for the pov display and controls the timing of the LEDs. Timing of the LEDs is accomplished using an external interrupt from the reed sensor.

* led.h

This is a library for the LED strip that is provided on Pololu.com. Although I had originally planned to try and write my own controller, once I downloaded the code and read through it I found that it contained many optimizations. These optimizations were mostly in the form of assembly code that allowed quicker response time from the LEDs. Given the importance of timings for my project and a lack of knowledge about or time to learn assembly, I decided to move forward with the provided libraries.

* led\_pattern.h

This code provides the necessary values to control which colors are output to the LED strip. Each pattern has its own function which returns the colors in the order they are to be displayed.

What I Learned

A lot…it was an interesting journey tackling this project!

1. Building the hardware takes much more time, patience, and knowledge than I expected.

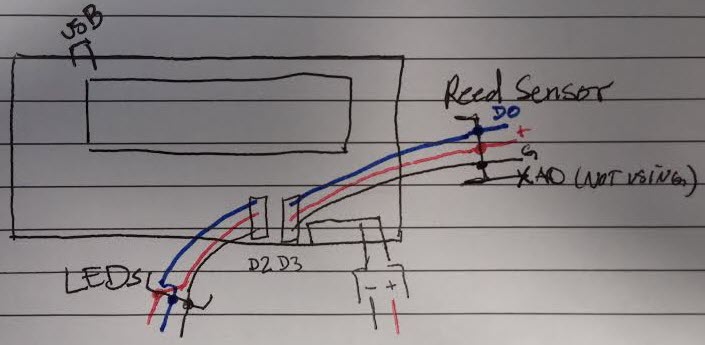
Although my online research made it seem relatively simple, that was not my experience. The first two drives that I took apart would not spin continuously once the disk readers and magnets were removed. Some research online revealed that it’s because the processor driving the motor has been coded to take action if no data is being received. So the drive assumes an error has occurred, stops itself, resets, and then starts back up. The third drive I tried finally worked, but the chassis was completely wrong for supporting the LEDs. So I then learned some remedial woodworking skills to create a support structure I could mount the LEDs to. Finally, the part of the LED strip that came with connections burnt out so I had to cut out the bad LED, learn to solder, and re-solder the wires.

1. There are many types of motors and they aren’t interchangeable.

My original design for the project was to be able to control both the motor and the LEDs using our microprocessor. I soon realized that most hard drives today use a 3-phase brushless DC motor that is much more difficult to drive than the motors we worked with. Along the way I learned that BLDC motors are controlled by a current that is supplied to each of 3 coils. The current must be supplied in the correct sequence and with the correct timing, which can vary by each individual motor, so it involves a lot of trial and error. But, by connecting each coil’s lead to a digital output pin and then sending high or low signals at the right time, it would technically be possible to drive the motor using our microcontroller. This would also allow me to change the speed of the platter which would be useful.

1. Troubleshooting a system composed of homemade hardware and software is very difficult.

It was really tough to know when something wasn’t working because of a hardware malfunction, a bad/wrong connection, or a coding issue. I think my most frustrating moment was when I spent 2 hours troubleshooting my sensor only to realize that I had reversed the LED and sensor pins. This taught me to pay careful attention when connecting wires and to make a diagram of where everything plugged in so I didn’t make the same mistake again.



1. External interrupts are cool.

I tried out the sensor using both a software interrupt and an external interrupt, just to see the difference. I ran into issues with the software interrupt where it wasn’t triggered every time the magnet passed by the reed sensor. I was unable to verify the exact reason suspect either the circuit for the reed sensor was performing some other manipulation that was interfering with the overall timing and/or the software interrupt was being trumped by some other process. The external interrupt had a better and more consistent response time, so my final implementation uses that.

Not Functioning

* Pattern does not display clearly. The fix for this is to slow down the speed of the motor by creating a controller that can drive the BLDC motor directly. Additionally, if the LEDs could be angled so that the light shines more directly through the slot in the platter the image might be clearer even at a higher speed.
* There’s only one pattern currently programmed and there’s no mechanism built in to change the pattern other than changing the code.

Future Work

* Write a controller for the BLDC motor so that I can control the speed and determine if the optimal speed of rotation depends on which pattern is being displayed
* Include external controls (i.e. pushbuttons on our board) that will allow changes to the program without modifying the software
* Code additional patterns such as a clock and family locator
* Angle the LEDs so that more of the light is reflected outwards
* Create the LCD portion of the original project, build the mobile application that would interface with the hardware components, and design GPS/wireless functionality