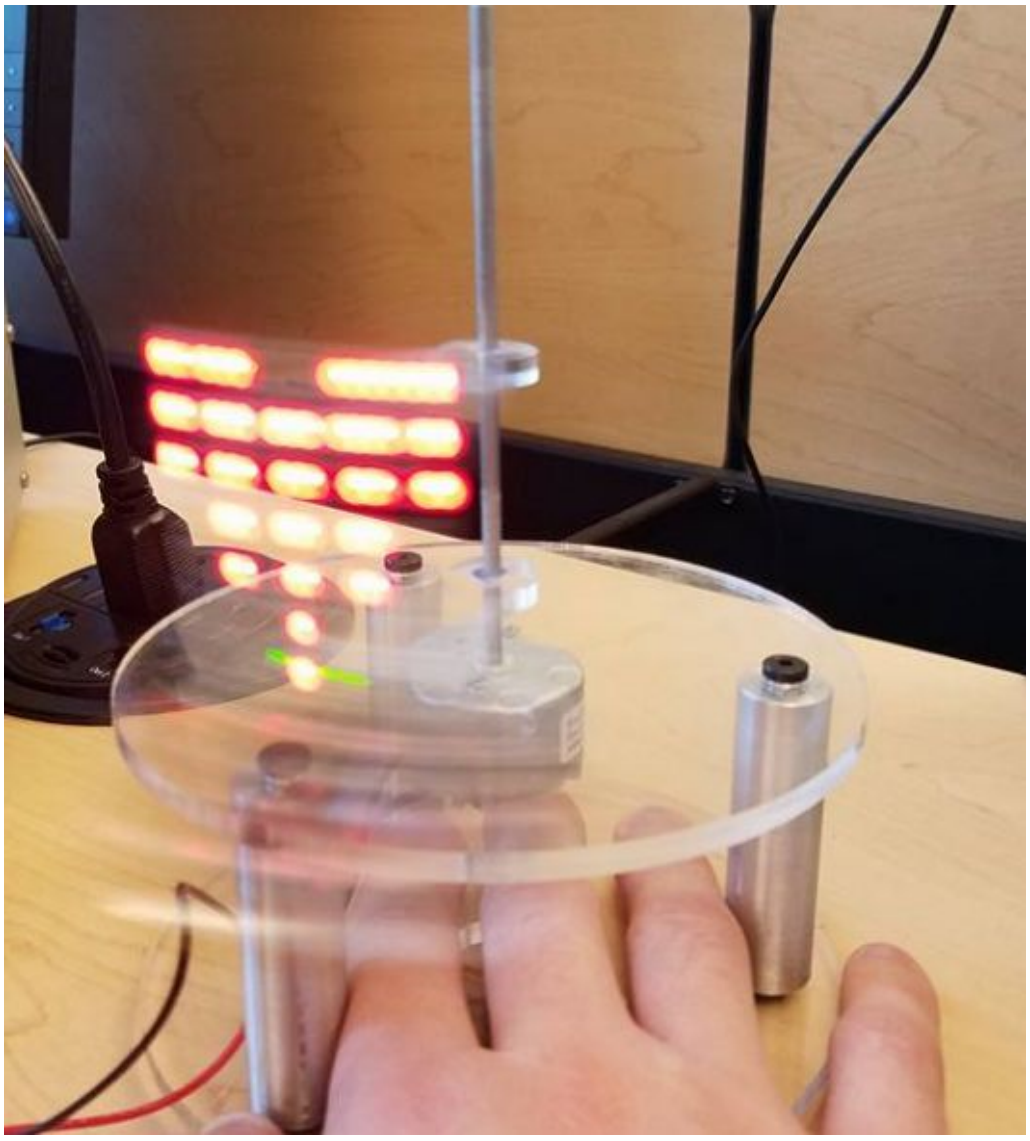
E15B Final Project: Displaying Text and Symbols with NeoPixels Using Persistence of Vision



Liam Kelly and Eric Chen

Abstract

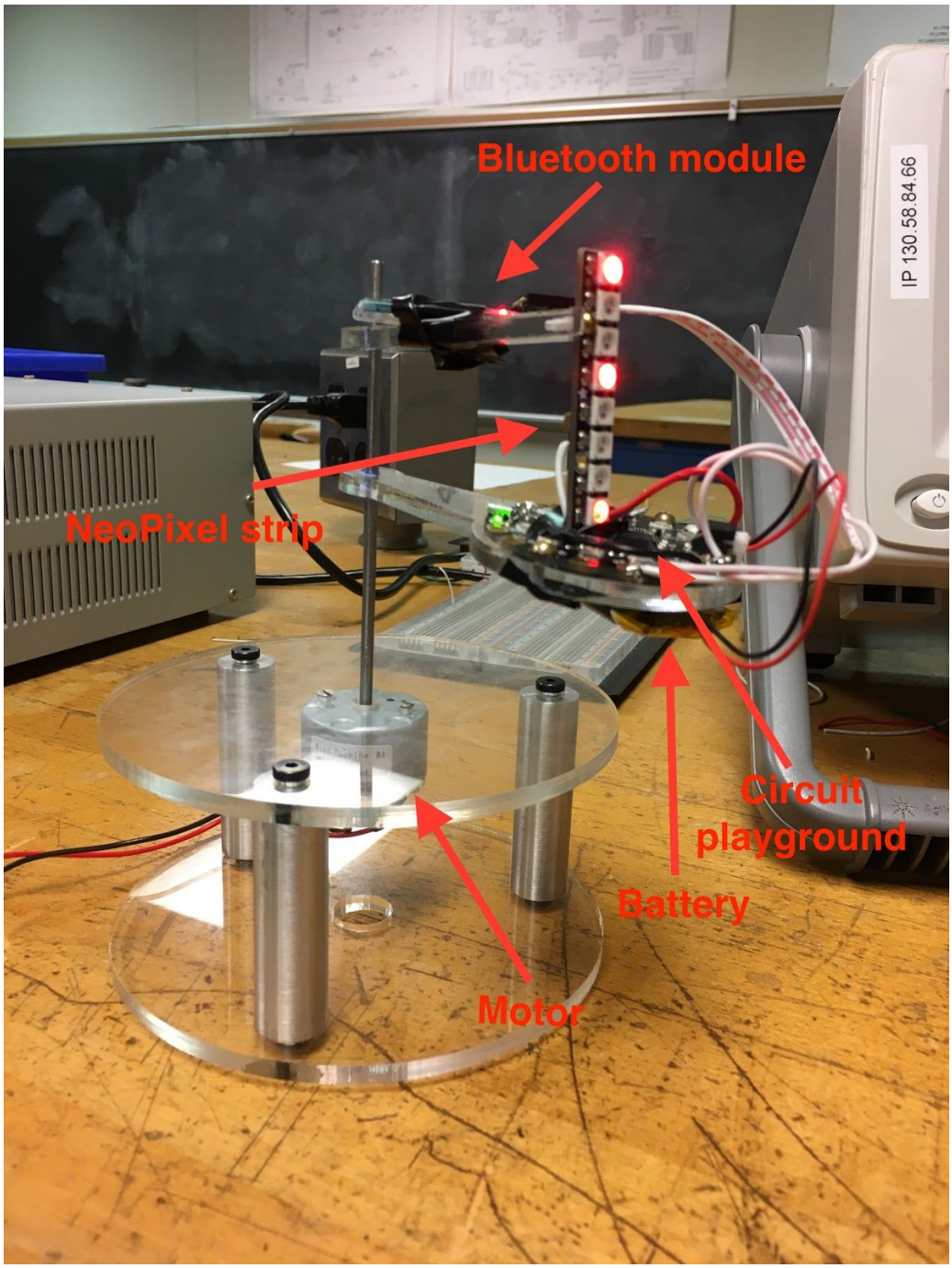
Due to chemical reactions, the eye maintains absorbed light for about a tenth of a second. Consequently, light that is continuously absorbed over the course of a tenth of a second will appear to form a single image.[1] Our project takes advantage of this fact, utilizing a phenomenon known as “persistence of vision”. By achieving a minimum linear velocity, our project appears to form letters and symbols that move around in a circle. The project works, but it does not work as well as hoped. Rather than working independent of voltage after reaching a threshold voltage, there is a “sweet-spot” of operation. Intended improvements to the project included utilizing a hall-effect sensor to detect RPM and using a bluetooth transceiver to interface with an android application in order to select messages to show. Unfortunately, both implementations were unable to be completed; this will be discussed further in results.

Motivation

The project was selected in order to present a mechanical and software challenge. Having witnessed the phenomenon before, the project owners sought to recreate the persistence of vision effect. Furthermore, the project was largely motivated by the fact that it is quite cool.

[1] "The Persistence of Vision - Foundations Magazine." The Persistence of Vision. Foundations Magazine, n.d. Web. 10 Dec. 2016.

High-level Description



The hardware setup centers around a rotating strip of NeoPixel LEDs. A motor powered by an external power supply turns an axle, on which two arms are attached. The lower arm

houses the Circuit Playground on top and the battery on the bottom, and the upper arm primarily serves to support the NeoPixel strip, which is about 2.5 inches from the axle. We found this to be a good radius for persistence of vision, as it was far enough out to display characters with accuracy but not far enough out to cause too much wobbling (though there is some). The battery is crucial because it allows us to power the LED strip and Circuit Playground without having to connect them to another external power source. The bluetooth was going to be used to change the color of the display remotely but we ran into some difficulty implementing the bluetooth peripheral.

One piece crucial to communication between the hardware and software was the accelerometer. We used accelerometer values to determine the tangential velocity of the NeoPixel strip (using the equation for centripetal acceleration in uniform circular motion), then used this to determine the period and subsequently how long each NeoPixel should be left on. Details on the theory behind this and difficulties we ran into are included below in the implementation section, but essentially we were getting inaccurate values from the accelerometer, which translated into bad LED displaying. We ordered a hall effect sensor to have a more accurate period reading (it would go high once every rotation due to detecting a magnetic field) but unfortunately broke a pin off in testing it on the breadboard, so we stuck with the accelerometer.

Two main parts of the software were the reading from the registers of the accelerometer and the actual displaying of predetermined messages. Reading from the registers involved a SPI interaction between the accelerometer peripheral and microcontroller, which required us to both read values and store them as well as set certain options in the control registers of the accelerometer that suited our demand (e.g. setting the scale of the accelerometer to 16 Gs). The displaying of the LEDs was done with the use NeoPixel library, hard-coded arrays, and delays as mentioned below in implementation.

Implementation

The project began with concept sketches (see appendices) designed in AutoCAD. The first part of the project that was implemented was the mechanical side. CAD files were designed for laser cutting the arms and base plates. These were cut out of ¼” acrylic; the top arm is 2.5” in length, the bottom arm extends 2.5” to the center of the base where the arduino rests, and the base plates are 5” diameter. The standoffs were made out of aluminum; they were machined and threaded. The shaft is ⅛” diameter aluminum that is 4” in height. Threads were made in the arms in order to hold the neopixels and the arduino in place. Holes were drilled in several locations in the base plates in order to hold the standoffs and the motor.

Upon completion of the mechanical structure, the electronics were assembled. The neopixels’ data line is soldered to digital pin 9 on the circuit playground, its ground pin is

soldered to ground, and vcc is soldered to the battery voltage pin. Though the neopixels call for 5V supply, it was found that they would operate perfectly well with the 3.7V supplied by the 3.7V Lithium-Ion battery used. The bluetooth receiver's Vcc, GND, RX, and TX pins are soldered to Vbatt, GND, TX, and RX, respectively. The battery is held by tape beneath the plate on which the arduino rests, and when it is used to power the arduino, its JST connector is plugged into the arduino's JST port.

The software side required obtaining the linear velocity in order to determine when the LEDs should be illuminated. Each letter and symbol has its own 2D array that draws the letter out of 1's and 0's, where the 1's draw the letter and 0's are background. This means that as the column spins, the 2D array will be looped through, turning on the LEDs one column of the 2D array at a time. Illuminating one column of the 2D array is referred to henceforth as an LED event. Specific equations and derivations are in the appendices; however, the concept utilized is that the LEDs should be on for the fraction of the period of rotation that corresponds to the fraction of the circumference that their diameter makes up. Essentially, this means that each LED event occurs for enough time so that they are lit for the width of the physical neopixels before the next LED event occurs. In order to obtain the period, the circumference is divided by the linear velocity. The linear velocity is determined via the radial acceleration, obtained through the X-Y acceleration from the accelerometer onboard the arduino. This radial acceleration is translated into linear velocity using basic kinematic relationships.

```
const int h[8][7] = {
      {1,0,0,0,0,0,1},
      {1,0,0,0,0,0,1},
      {1,0,0,0,0,0,1},
      {1,1,1,1,1,1,1},
      {1,0,0,0,0,0,1},
      {1,0,0,0,0,0,1},
      {1,0,0,0,0,0,1},
      {1,0,0,0,0,0,1}
      };
```

The initial results after implementation indicated that the accelerometer values were inaccurate. This may be due to the fact that the arm on which the arduino sits is slanted, meaning there is an additional acceleration in the X-Y. However, even accounting for this, the values did not lead to an accurate calculation of the period, which was essential for determining the length of an LED event. In order to get a more accurate determination of the period, a hall-effect sensor was purchased to build a rotary encoder. However, while testing the hall-effect sensor, no readings were obtained. The sensor may have been damaged by soldering; this damage could have been avoided by soldering M-F jumpers to the board and attaching the hall-effect sensor to the jumpers. Had readings been obtained, interrupts would have been used in order to get readings from the hall effect sensor.

The final aspect of the implementation involved developing an android app to interface with a bluetooth transceiver (HC-06) so that signals could be sent to the arduino indicate what letters and symbols to draw. After the android application was developed, the transceiver was tested using an oscilloscope. However, no signal was detected coming out of the RX pin of the transceiver. No heat damage would have been possible because M-F jumpers included with the transceiver were soldered to the board and the transceiver was attached to these. Furthermore, the android application had been certified to work with other transceivers in the

past, and it was confirmed that the android application was correctly pairing with the transceiver through the indicator LED on the device.

Results

After the initial implementation, the device was tested by asking subjects to identify the message written. Though tests were successful, the project owners sought to make the message even clearer by getting more accurate readings for the period through a rotary encoder. However, as noted in implementation, the hall-effect sensor was incapable of producing readings, and consequently, the rotary encoder could not be implemented. Because the accelerometer values were inaccurate, the ultimate approach taken was to modify delay values and note which were the most successful -essentially trial by error. Furthermore, the voltage supplied to the device was varied until an optimal value was found.

The android application should have been successful, however, no signal was produced by the RX pin of the transceiver. The transceiver was debugged using the serial monitor and AT command mode, but the project owners were incapable of implementing it successfully.

Ultimately, the device is capable of producing hard-coded messages that can be understood after adjusting the voltage supplied to the motor to an optimal value. The message can be understood better from afar; this is likely because the message is moving in the circle and a wider view allows it to be appreciated more accurately.

Advice for Future Work

For those who wish to recreate the project, we suggest implementing a rotary encoder using a hall-effect sensor to obtain more accurate values. In order to avoid head damage, solder using M-F jumpers and then attach them to the hall-effect sensor. We also recommend obtaining a more powerful motor; our motor was obtained from spare parts and could not achieve a very high angular speed. Counterweights to the Circuit Playground and a heavier base are also highly recommended to reduce wobbling. Finally, we recommend using the HC-05 bluetooth module, which has been used previously with success for interfacing android with arduino. The interaction between android and arduino could be expanded to displaying custom messages or symbols as entered on the phone side. Please see our github (https://github.com/zappyfish/E15B-final-project) for additional files that may be of use. We would be very excited to see our project reproduced more successfully!

Bill of Materials

(note: materials for mechanical structure are NOT included)

Description	Vendor	Part Number	Qty	Cost	Link
3.7V 400mAh Li-Ion Battery	Sparkfun	PRT - 13851	1	$4.46	https://www.sparkfun.com/products/13851
Arduino Circuit Playground	Adafruit	ID: 3000	1	$19.95	https://www.adafruit.com/products/3000
Adafruit Neopixel Stick	Adafruit	ID: 1426	1	$5.95	https://www.adafruit.com/products/1426
Bluetooth Transceiver	Maker Central	B008AVPE6Q	1	$9.99	https://www.amazon.com/Bluetooth-converter-serial-communication-master/dp/B008AVPE6Q
Hall-Effect Sensor (NO magnet)	Adafruit	ID: 158	1	$2.00	https://www.adafruit.com/products/158

Acknowledgements

We would like to thank Professor Cheever, James Johnson, and Edmond Jaoudi for their invaluable assistance and consult in developing this project. We would like to also thank James Johnson for the hours he put forward helping us machine the physical structure.

Appendix A: Selected Excerpts of Code

Displaying the LEDs (in the loop function) :

```
 if(accelXY != 0) {
        period = getPeriod(accelXY);

        drawLetter(i, period);
        delayMicroseconds(1000000*period);
        drawLetter(heart, period);
        delayMicroseconds(1000000*period);
        drawLetter(e, period);
        delayMicroseconds(700000*period);
        drawLetter(n, period);
        delayMicroseconds(700000*period);
        drawLetter(g, period);
        delayMicroseconds(1000000*period);

    }
```

drawLetter function:

```
void drawLetter(const int let[8][7], float period) {
  float LEDtime;
  LEDtime = 1000000*period*LEDwidth/circumference;
  for(int j = 0; j<7; j++) {
    for(int i = 0; i<8; i++) {
      if(let[i][j] == 1) {
        strip.setPixelColor(7-i, red, green, blue);

      }
    }
    strip.show();
```

```
    //delayMicroseconds((int)(LEDtime));
    for(int i = 0; i< 8; i++) {
    strip.setPixelColor(i, 0, 0, 0);
    strip.show();

  }
//  delayMicroseconds((int)(LEDtime));
    delayMicroseconds(700);  //arbitrary delay so that letters
are not spread out too much yet do not move too quickly
   }
}
```

Multi-bit read for accelerometer values:

```
void mBitRead (byte address, byte numToRead) {
  int i;
  digitalWrite(CS, LOW); //pull CS low to begin transfer
  SPI.transfer(READ| MBIT | address); //read mode, multi bit,
address of reg to read
  for (i = 0; i < numToRead; i++) {
    accelVals[i] = SPI.transfer(DUMMY); //master tranfers dummy
while reading into data[i]
  }
  digitalWrite(CS, HIGH);
}
```

Appendix B: Mathematical Relationships

For circular motion, we have a simple relationship between acceleration and tangential linear velocity:

$$a = \frac{v^2}{r}$$

Where a is the acceleration, v is the velocity, and r is the radius of motion.
Therefore

$$v = \sqrt{a \cdot r}$$

For the time of an LED event, we want it to correspond to the portion of the circumference taken up the by diameter of an LED. Therefore we have that:

$$\frac{t_{LED}}{T} = \frac{d_{LED}}{c}$$

Where t_{LED} is the time of an LED event, T is the period of rotation, d_{LED} is the diameter of an LED, and c is the circumference of the rotation.