Amar Jyothiprakash
A12098251
Samuel Oberlies
A11304326

Physics 124 Project Report

Motivation and Overall Concept:

We often witness certain "programmed" responses to light in our environment. We are all trained to recognize the colors and meanings of these items, such as behind traffic lights, and respond accordingly. But, what if we could step past the human need to take action and get the cars to respond to the lights themselves?

When running through project ideas, we thought it would be interesting to create an obstacle course for a vehicle, that it could potentially solve on its own. To step even further than that, we were most intrigued by the idea of designing a vehicle that could solve more than one preset obstacle course. Essentially, the course would be fully flexible and we could navigate around it any way we want.

After considering these desires, we decided to try to build a hovercraft that would respond to various colored lights and react accordingly. The main objective is designing a course that would emit various colored lights (red, blue, green, etc.), having our hovercraft sense a color and then having it perform a command based on that (i.e. sensing blue light means the hovercraft turns right).

Functional Definition:

As far as the hovercraft goes, we didn't aim for anything crazy. We wanted it to levitate off the ground and move and turn as we command. We powered the movement with DC motors and propellers. We built a model that has one propellers facing downward creating the lift, one in the rear providing forward movement, and one servo powering a fin-like rudder to help with direction. The motors were powered by the bench-top power supply as they require large amounts of voltage and current to maintain motion. We controlled the movement and turning motions through a micro-servo motor and a directional blade on the back of the hovercraft. Basically, we told the servo to move the blade to a specific angle, which directed the airflow and cause the craft to turn.

We used the Arduino and environment to give the hovercraft the commands. We have designed a simple course to show the crafts functionality, a simple start light, two turns, and a stop light. The hovercraft detected the colors with a sensor and then react to that input through the Arduino and our software. We have programmed stop and go lights which control the rear propeller turning off/on, as well as two turning lights that control the servo.

Sensors:

Our functionality solely depended on being able to sense colors and react accordingly. To accomplish this, we used a SparkFun RGB and Gesture Sensor - APDS-9960. This is mounted on the front of the hovercraft and will detect the light. We powered the sensor with the 3.3V supply off the Arduino, and had it wired into 2 specified pins, the SDA and SCL on the Arduino, to read values for the software.

Mechanical Considerations:

Initial considerations all revolve around the construction of the hovercraft itself.

The body of the craft was made of styrofoam board, a form of insulation. The design was a simple rectangular shape with a main platform top layer, and a bottom frame supporting layer. The bottom layer provided support for the platform layer and helped direct air downward.

The dimensions of our board were approximately 16 inches long and 8 inches wide, with the foam thickness being 1 inch.

The plastic skirt was the key to hovering. It was made of 3.5 mil plastic sheeting, cut to match the shape of the craft. The escaping air created a cushion that the craft will sit on. The hole was lined with tape to reinforce it, and the skirt itself was secured to the craft using tape.

Two DC motors were fitted with propellers and directed appropriately based on function. These motors were glued to wooden supports. For the back propeller, we used an A-frame type wooden support. The wood supports were attached to the craft with screws, washers, and nuts. The propellers were approximately 8in and will be aligned at the center line of the craft, facing downwards and backwards.

The servo motor was taped on in the rear, and the blade turning system was attached to the motor by mounting it directly. The blade was made of paint sticks taped together. For the blade we only needed 0-180 degrees of rotation for our turns.

The RGB sensor was mounted onto the foam board through a breadboard. It was positioned at the front of the craft to ensure the optimal sensing range.

The light emitters were Neopixel Stick-8X5050 RGB LEDs from Adafruit. We controlled their color with a separate microcontroller, not on the craft. We had them all hooked up to another Arduino Uno that maintained signal pins for the different colors.

Overall, a big concern was the weight we put on the craft and the ability to still sustain levitation. Also, balance issues were key as the craft can tend to drift, especially due to the pull of the long wires used to power it.

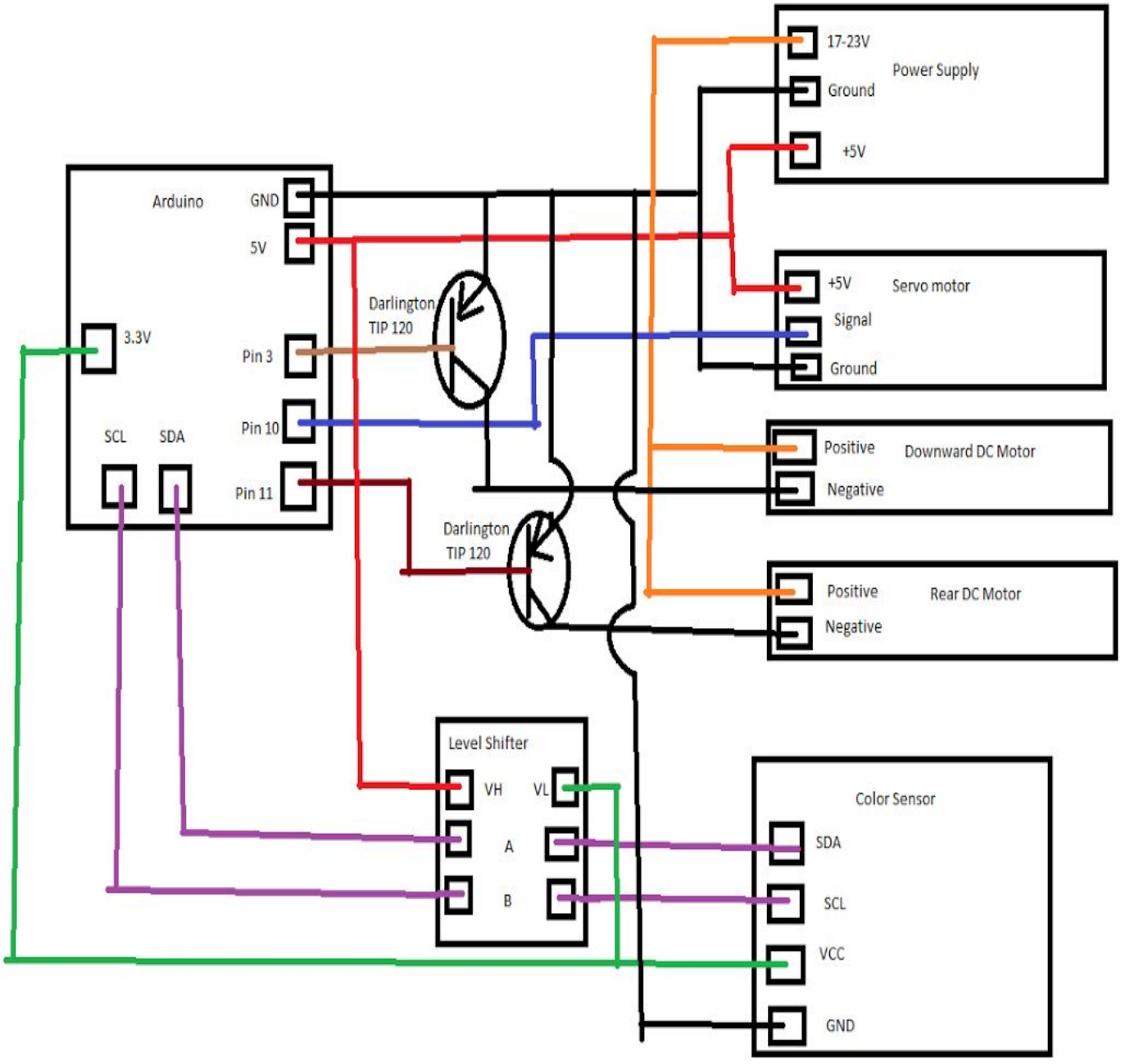
Electrical Considerations:

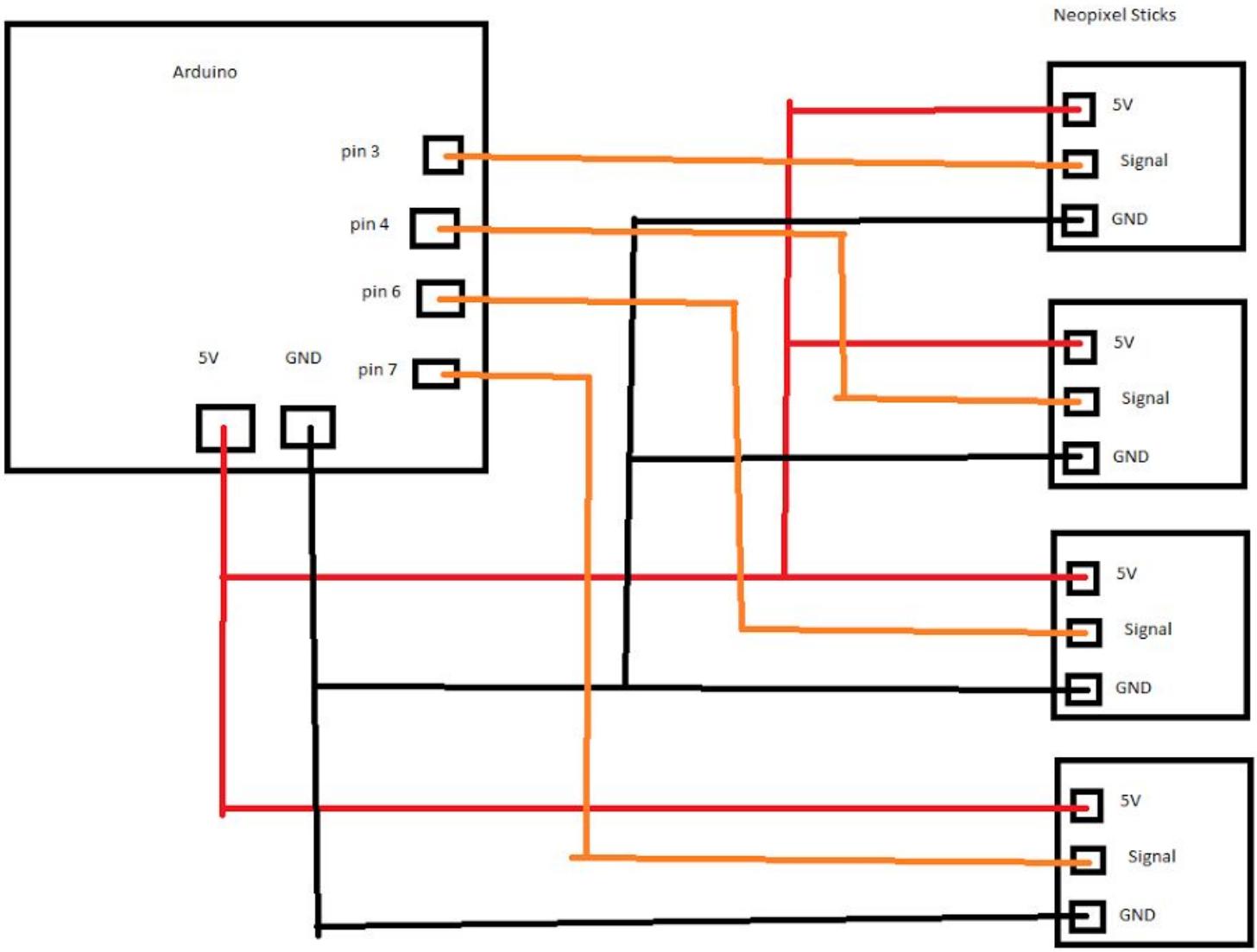
The motors were controlled by transistor switching logic that is fed into PWM pins on the Arduino Uno, (2 total transistors). The RGB sensor needed two Arduino pins, the aforementioned SDA and SCL, as well as a 3.3V supply and ground. We also included a level shifter to make sure the logic going into the sensor is the appropriate 3.3V signal level. We mounted a small breadboard to help with the wiring and establish clear power and ground lines, as well as to help keep our circuitry clean.

Interface:

We used two Arduino Unos; the same ones used in this class's previous lab assignments. An Arduino Nano would have been helpful for weight considerations, but had less pins. We had intended to use this because of our weight concerns, but the appropriate drivers were not available to us. Therefore, we used the Arduino Uno. We used the 3.3 V and 5 V power and ground pins on the Arduino, the SDA and SCL pins, two PWM pins, two transistors, and the corresponding digital pin to control the servo on the craft. For the neopixel Arduino we used 4 signal pins to control our colors. The serial monitor was helpful to allow us to see what values the RGB sensor was reading and gave us the information we needed to manipulate those values to program our commands.

Here is a schematic of the circuit used:


17-23V
Power Supply
Ground
+5V
Arduino
GND
5V
3.3V
Pin 3
Pin 10
Pin 11
SCL
SDA
Darlington
TIP 120
+5V
Servo motor
Signal
Ground
Positive
Downward DC Motor
Negative
Darlington
TIP 120
Positive
Rear DC Motor
Negative
Level Shifter
VH
VL
A
B
Color Sensor
SDA
SCL
VCC
GND


Neopixel Sticks
Arduino
pin 3
pin 4
pin 6
pin 7
5V
GND
5V
Signal
GND
5V
Signal
GND
5V
Signal
GND
5V
Signal
GND

Software:

We programmed in C for the Arduino. We needed the servo, DC motor, and neopixel libraries for movement. Also, we had to use the RGB color sensors library.

The rest of the commands were normal Arduino commands and shouldn't be too complicated. We adjusted our values and scale appropriately via trial and error. They were enforced by a series of *if* and *else-if* statements.

The main problems were making sure our commands were clearly designed to only implement with the correct colored light and at the appropriate time or distance that we wanted.

Testing:

During testing we ran into many issues. One of the most significant was when we found interference and noise ruining our RGB sensor. The DC motors leaked radiation that greatly affected the performance of our sensor. This is due to the high voltages we required to initiate movement. To solve this issue, we used shielded wire. The complications of lots of wiring and subsequent inconsistencies defined our testing procedure.

Safety:

There were not many safety concerns for this project, the only things added to the ambient environment are a hovering piece of foam and some lights which did not cause any harm.

A potential concern, which did not occur, could have been the propellers coming into contact with something they should not. This was avoided with general safety rules and being aware of when things are switched on or off.

Parts and Reusability:

We ordered the color sensor and micro servo. Foam and plastic sheeting were from Home Depot. The DC motors we used from the lab. The propellers we ordered. The directional fin on the servo was made from paint sticks we purchased. We ordered the neopixel strips we used as the light emitters.

Expansion Options:

If we had extra time, we would have liked to add further commands and colors to react to, beyond our basic plans for turning that we have now. We would have liked to be able to implement features such as reversing, 180 turning, and more functionality. We would have liked to create a set course for the hovercraft instead of just demonstrating the features one at a time.

De-scope Options:

We had to descope from our initial plans and maintain just start, stop, and turn commands rather than some of the expansion options we had planned before. Due to some last-minute issues, we downsized from the original plan and did not have side propellers to help

with the turns. We did not set up a course as intended, and as previously mentioned, we demonstrated the commands individually instead.

Drawbacks:

The power needed to run the craft was a major issue. We had to overcome the problem of the DC motors needing a lot of juice to keep the craft up and running. Additionally, all of our movement functionality was much harder with a hovercraft than it would have been with a car design. Without wheels to easily turn, the movement can be inconsistent and hard to control. Additionally, since we have no points of friction, all directional control and speed are dependent solely on airflow, which is a lot less controllable and reliable.

Highlights:

We actually got something to hover and move which was a big part of our goals. The light sensing mostly worked and our basic ideas came to life in our project. Although it was very challenging, overcoming each new obstacle and every step forward was very rewarding.

Final Result:

During the final demonstration, we showed the ability of our craft to react and move as described. Also, we turned in our code (see Appendix A) and this updated report that has all information pertaining to any changes we made and what issues we ran into. Additionally, the major changes to the write up were what aspects of the project worked best, what were the drawbacks, and how we could see this expanding if we were to devote much more time to it, which can all be reviewed in the previous sections of this report. Lastly, for your viewing pleasure we have included a picture of us with our final craft, see Appendix B.

APPENDIX A: The Code for Our Project

Craft:

```
int pos = 0;
int backprop=0;

#include <SparkFun_APDS9960.h>

#include <Servo.h>
Servo servo;
const int servopin = 10;
const int dcdown=3;
const int dcrear=11;
//const int dcleft=5;
const int off=0,on=1;
//dcright=6
//, off=0,on=1;

const int centerdeg=87, leftdeg=60, rightdeg=120;
#include <Wire.h>
#include <SparkFun_APDS9960.h>
SparkFun_APDS9960 apds = SparkFun_APDS9960();

uint16_t ambient_light = 0;
uint16_t red_light = 0;
uint16_t green_light = 0;
uint16_t blue_light = 0;
float ratioRed,ratioGreen,ratioBlue;


void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);
motorsetup();
servo.attach(servopin);
RGBsetup();
servo.write(centerdeg);


}

void loop() {
//  sweep();
  apds.readAmbientLight(ambient_light);
       apds.readRedLight(red_light) ;
       apds.readGreenLight(green_light);
       apds.readBlueLight(blue_light);
```

```
 ratioRed = ((float)red_light)/((float)ambient_light);
 ratioBlue= ((float)blue_light)/((float)ambient_light);
ratioGreen = ((float)green_light)/((float)ambient_light);
  colorstoSerial();
if(
//  (ratioRed <.43)&&
// (ratioBlue<.5)&&
//(ratioGreen >.35) ){

ambient_light>600 && (backprop==off)){
  turnon();
  backprop=on;
}
 else if ((ratioRed >.60)&&
 (ratioBlue<.3)&&
(ratioGreen <.3) )
 {
  turnoff();
  backprop=off;
 }
else if ((ratioRed <.5)&&
 (ratioBlue>.3)&&
(ratioGreen <.5) )
 {
   turnright();
   red_light=0;
  }
else if ((ratioRed <.35)&&
 (ratioBlue<.3)&&
(ratioGreen >.35) )
 {
   turnleft();
   blue_light=0;
  }

}

void RGBsetup() {
//   Serial.begin(9600);
  Serial.println();
  Serial.println(F("--------------------------------"));
  Serial.println(F("SparkFun APDS-9960 - ColorSensor"));
  Serial.println(F("--------------------------------"));
  // Initialize APDS-9960 (configure I2C and initial values)
  if ( apds.init() ) {
    Serial.println(F("APDS-9960 initialization complete"));
  } else {
    Serial.println(F("Something went wrong during APDS-9960 init!"));
```

```
  }
  // Start running the APDS-9960 light sensor (no interrupts)
  if ( apds.enableLightSensor(false) ) {
    Serial.println(F("Light sensor is now running"));
  } else {
    Serial.println(F("Something went wrong during light sensor init!"));
  }
  // Wait for initialization and calibration to finish
  delay(500);
}


void colorstoSerial() {
    if (  !apds.readAmbientLight(ambient_light) ||
        !apds.readRedLight(red_light) ||
        !apds.readGreenLight(green_light) ||
        !apds.readBlueLight(blue_light) ) {
    Serial.println("Error reading light values");
  } else {
    Serial.print("Ambient: ");
    Serial.print(ambient_light);
    Serial.print(" Red: ");
    Serial.print(red_light);
    Serial.print(" RatioRed: ");
    Serial.print(ratioRed);
    Serial.print(" Green: ");
    Serial.print(green_light);
    Serial.print(" RatioGreen: ");
    Serial.print(ratioGreen);
    Serial.print(" Blue: ");
    Serial.print(blue_light);
    Serial.print(" RatioBlue: ");
    Serial.println(ratioBlue);

delay(100);   }
  // Wait 1 second before next reading
  //delay(1000);
}

void turnright () {
  servo.write(rightdeg);
//  delay(100);
//  analogWrite(dcleft,150);
//  delay(100);
  delay(6000);
  servo.write(centerdeg);
//  delay(100);
//  analogWrite(dcleft,0);
```

```
   delay(500);
}

void turnleft (){
  servo.write(leftdeg);
//  delay(100);
//  analogWrite(dcright,200);
  delay(6000);
  servo.write(centerdeg);
//  delay(100);
//  analogWrite(dcright,0);
   delay(500);
}

void turnon () {
  analogWrite(dcrear,255);

}
void turnoff () {
  digitalWrite(dcrear,0);

}

void motorsetup(){
    TCCR2B = TCCR2B & 0b11111000 | 0x07;
pinMode(dcdown, OUTPUT);
pinMode(dcrear, OUTPUT);
//pinMode(dcleft,OUTPUT);
//digitalWrite(dcleft, LOW);
digitalWrite(dcdown, HIGH);
analogWrite(dcrear,0);
}

void sweep(){
    for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    servo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                     // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    servo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                     // waits 15ms for the servo to reach the position
  }
}
```

Neopixel:

```
// Global Variables


#ifdef __AVR__
  #include <avr/power.h>
#endif


// Which pin on the Arduino is connected to the NeoPixels?
// On a Trinket or Gemma we suggest changing this to 1
#define REDPIN            6
#define BLUEPIN           3
#define GREENPIN          4
#define PURPLEPIN         7

// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS      30
#define CONTROL 4
#define MINIPULSE 1000
#define MAXPULSE 2000

Adafruit_NeoPixel redpixels = Adafruit_NeoPixel(NUMPIXELS, REDPIN, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel bluepixels = Adafruit_NeoPixel(NUMPIXELS, BLUEPIN, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel greenpixels = Adafruit_NeoPixel(NUMPIXELS, GREENPIN, NEO_GRB +
NEO_KHZ800);
Adafruit_NeoPixel purplepixels = Adafruit_NeoPixel(NUMPIXELS, PURPLEPIN, NEO_GRB +
NEO_KHZ800);

void setup() {
  redpixels.begin();
  bluepixels.begin();
  greenpixels.begin();
  purplepixels.begin();// This initializes the NeoPixel library.
mysteriousEndifs ();
initializeColorstripred();
initializeColorstripblue();
initializeColorstripgreen();
initializeColorstrippurple();
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
}

void initializeColorstripred(){
    for(int i=0;i<9;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    redpixels.setPixelColor(i,255,0,0); // bright red color.

    redpixels.show(); // This sends the updated pixel color to the hardware.

 // Delay for a period of time (in milliseconds).

  }
}


void initializeColorstripblue(){
    for(int i=0;i<9;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    bluepixels.setPixelColor(i,0,0,255); // bright blue color.

    bluepixels.show(); // This sends the updated pixel color to the hardware.

 // Delay for a period of time (in milliseconds).
  }
}

void initializeColorstripgreen(){
    for(int i=0;i<9;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    greenpixels.setPixelColor(i,0,255,0); // bright blue color.

    greenpixels.show(); // This sends the updated pixel color to the hardware.

 // Delay for a period of time (in milliseconds).

  }
}
void initializeColorstrippurple(){
```

```
    for(int i=0;i<9;i++){

    // pixels.Color takes RGB values, from 0,0,0 up to 255,255,255
    purplepixels.setPixelColor(i,255,255,255); // bright red color.

    purplepixels.show(); // This sends the updated pixel color to the hardware.

 // Delay for a period of time (in milliseconds).

  }
}

void mysteriousEndifs (){
   // put your setup code here, to run once:
#if defined (__AVR_ATtiny85__)
  if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
#endif
  // End of trinket special code
}
```

APPENDIX B: PICTURE