Benjamin Vick (bnv8) and Kiyan Rajabi (kr499)

CS 5424 / INFO 5345

Final Project: Rhino LED

RhinoLED Documentation

Overview **2**

Planning **3**

Verplank Diagram 3

Paper Prototype 4

State Diagram 5

Materials 5

Procedures **6**

Powering Up Our LED 6

Rhino Head Encasing 6

Testing Our Microphone Sensor 9

Sound Reactive Implementation 10

Discussion **14**

Design 14

Taking It Further 18

Sources **18**

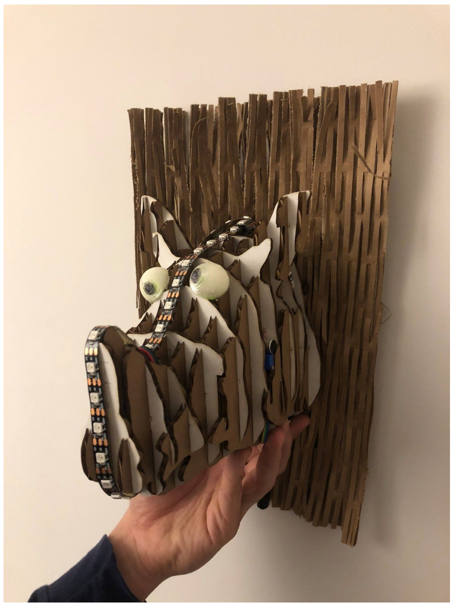
Overview

For our final project, we initially inspired to make something like this:

There were two skills we wanted to hone through the project:

1) Designing and individually programmable LEDs and have them react to sound
2) Designing and building a physical apparatus that allowed us to present this through an interesting and engaging interactive device

Final design and implementation:

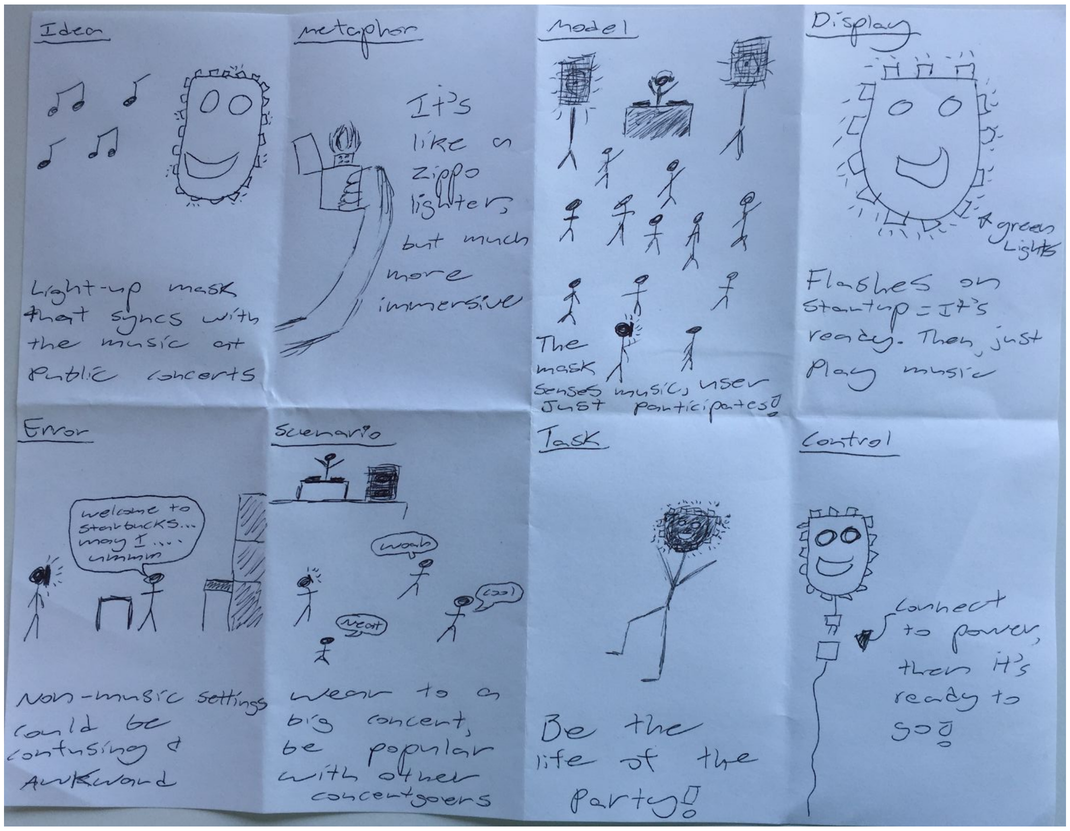


To see our final implementation, you can see our demo video at:

http://bit.ly/CrazyRhino

Planning

Verplank Diagram



Light-up mask that syncs with the music at public concerts

It's like a zippo lighter, but much more immersive

The mask senses music, user just participates!

Flashes on startup = it's ready. Then, just play music

Non-music settings could be confusing & Awkward

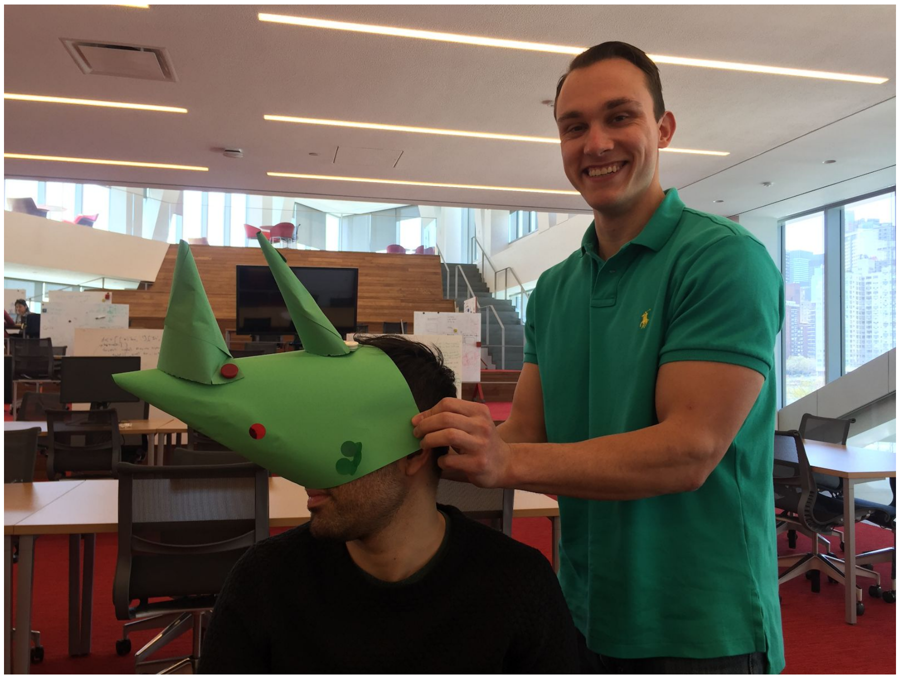
wear to a big concert, be popular with other concertgoers

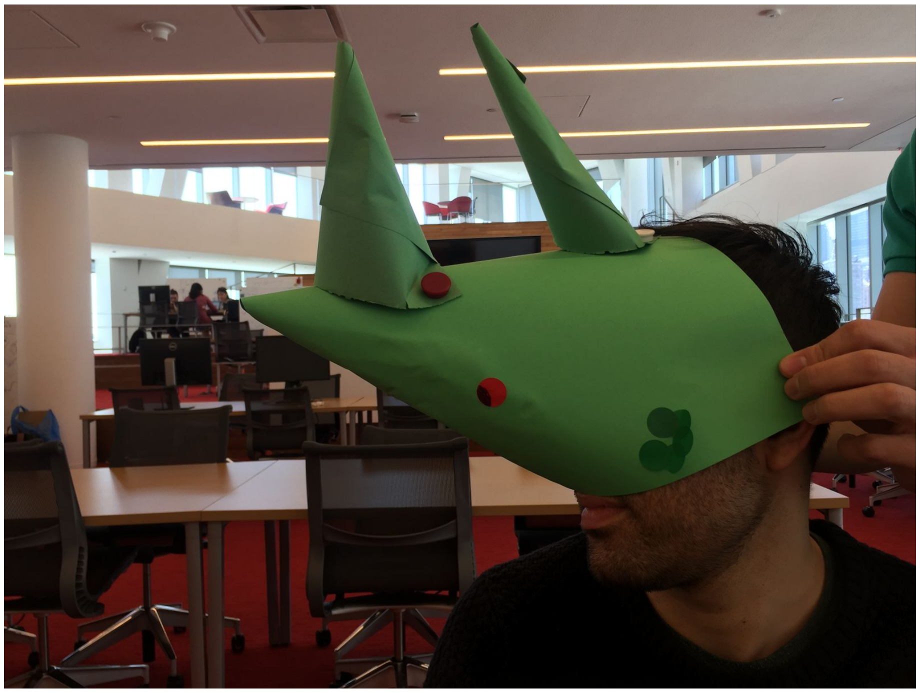
Be the life of the party!

Connect to power, then it's ready to go!

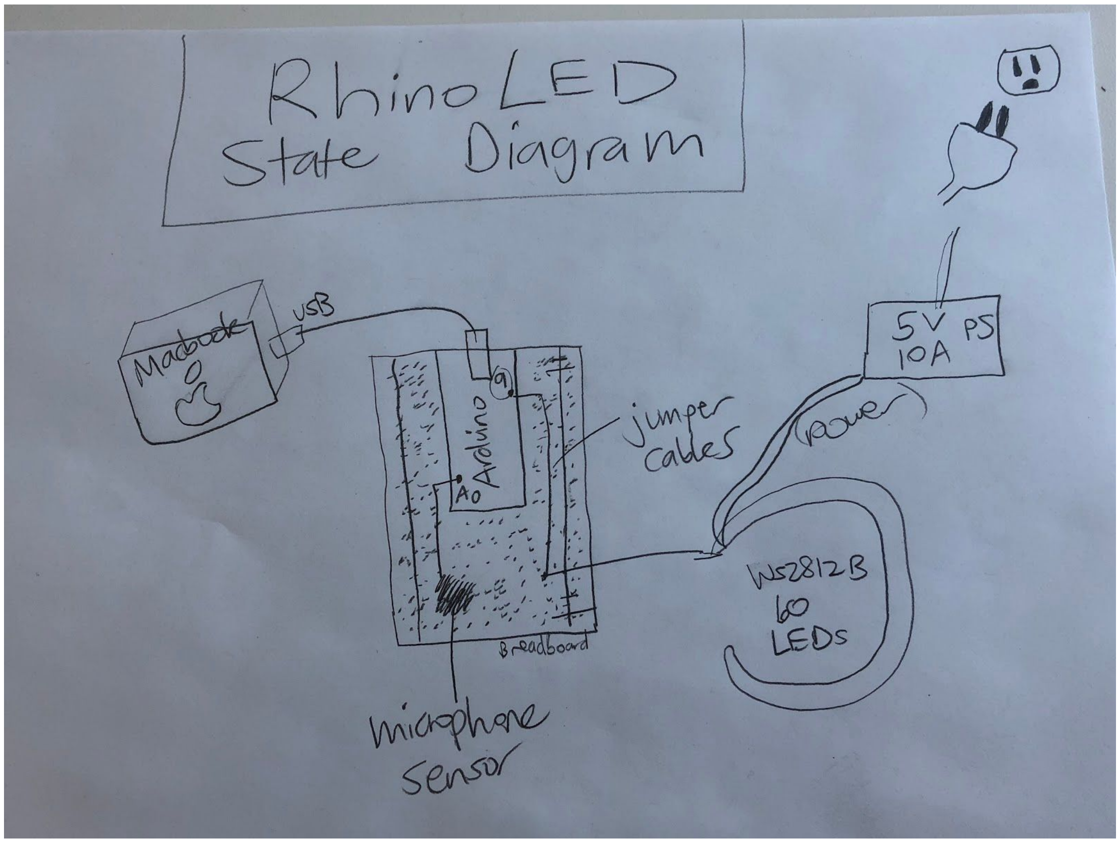
Paper Prototype

To test our prototype, we physically generated some 3D models that helped us visually experiment with what we wanted to tangibly build:



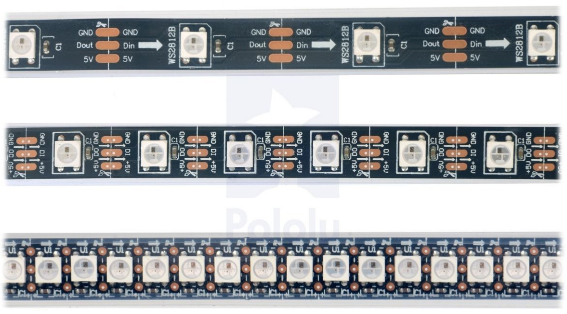


State Diagram



Materials

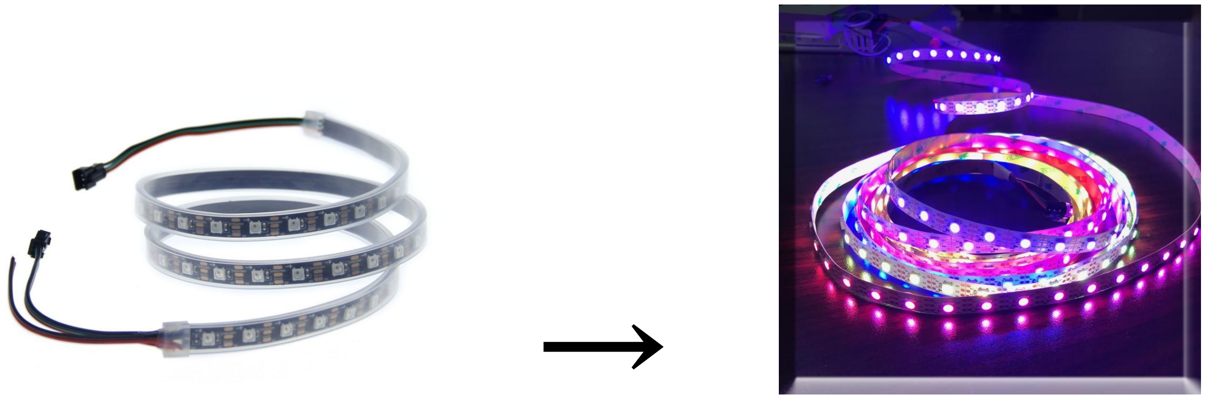
- Arduino
- Breadboard
- Jumper cables
- High sensitivity sound microphone sensor detection module
- WS2812B Individually Addressable 60-LED Kit
- 5V 10A Power Supply



Procedures

Powering Up Our LED

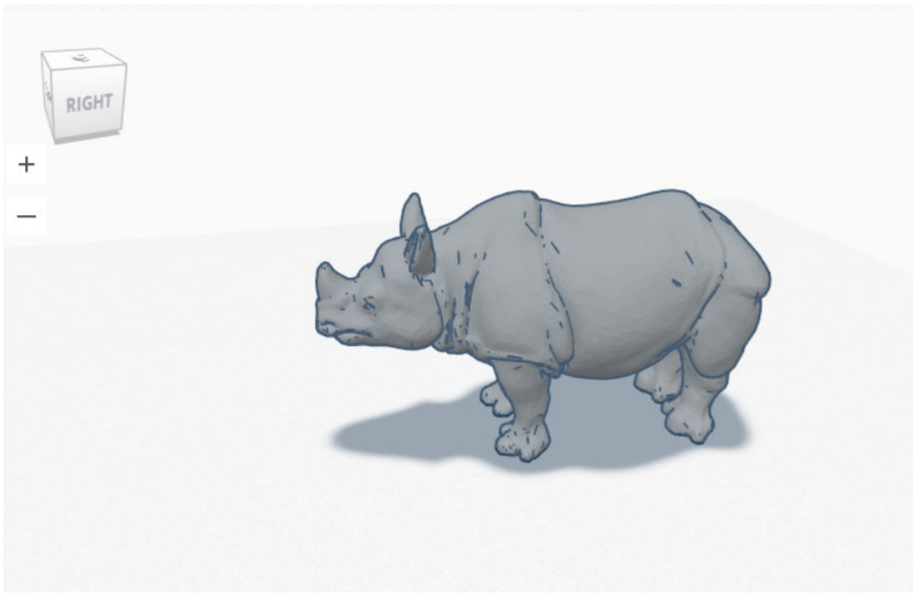
When we conducted initial testing, we used the Arduino’s power supply to test powering up about 5 of our LEDs.



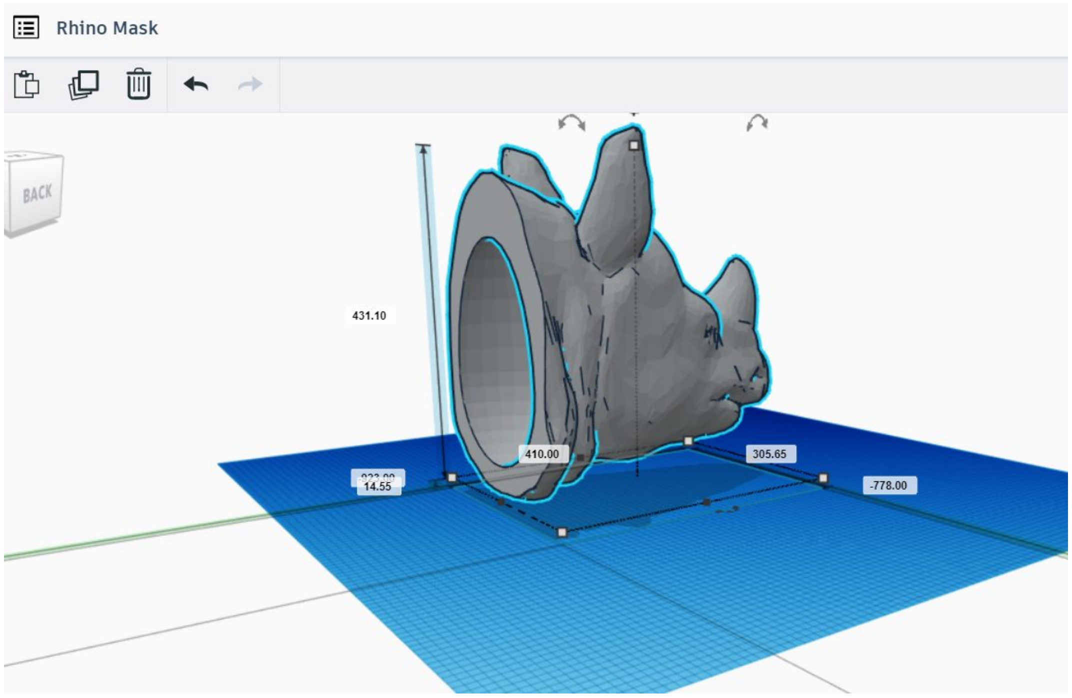
We purchased a 5V 10A Power Supply to power our LEDs. The data sheet indicated we needed at least 2.9A since the Arduino maxed at 1A.

Rhino Head Encasing and Illuminated Eyes

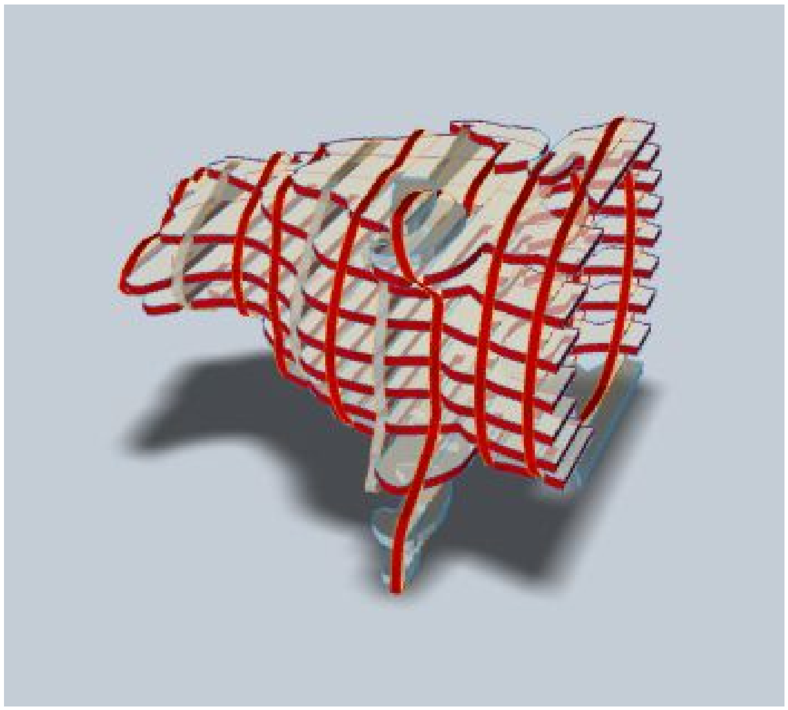
We started by looking through available projects on Thingiverse to find a suitable rhino to use for our model. Using that as our foundation, we then sliced the model so that we had just the head for our design:



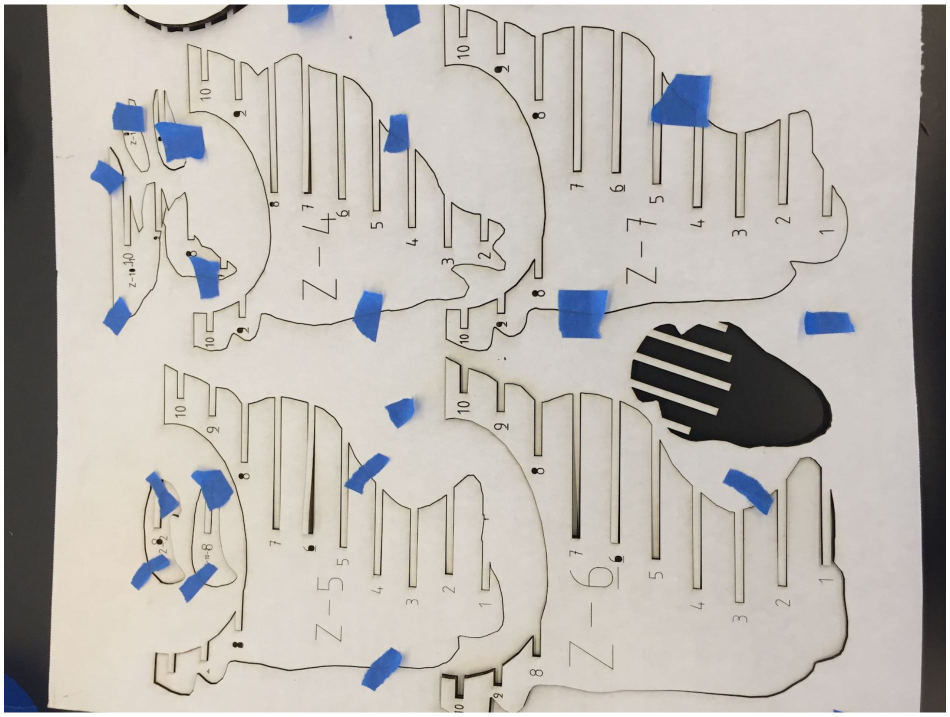
We imported the head into a new file and re-sized it to fit the approximate size we needed and created some negative space in the back of the mask to make room for the user's face and the Arduino/power assembly.



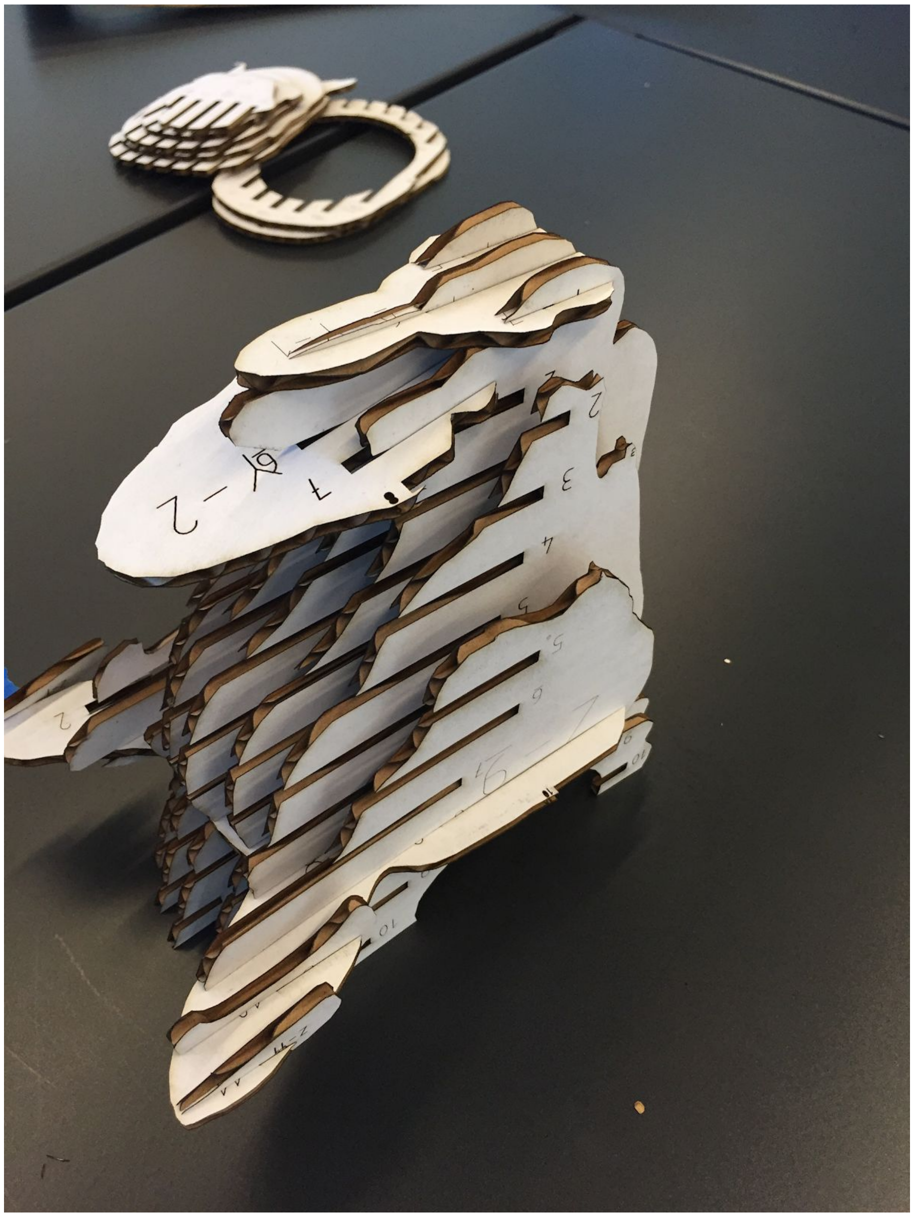
Once we were satisfied with our model, we exported the 3d model from thingiverse and imported it into slicer for fusion 360. We experimented with several different slicing methods and number of layers before settling on a design to print. We then exported the print file and transferred it to Adobe Illustrator.



Once we had the file ready to print on the laser cutter, we loaded the cardboard necessary to print each page and adjusted the settings as we went to get the appropriate scoring and cutting settings.



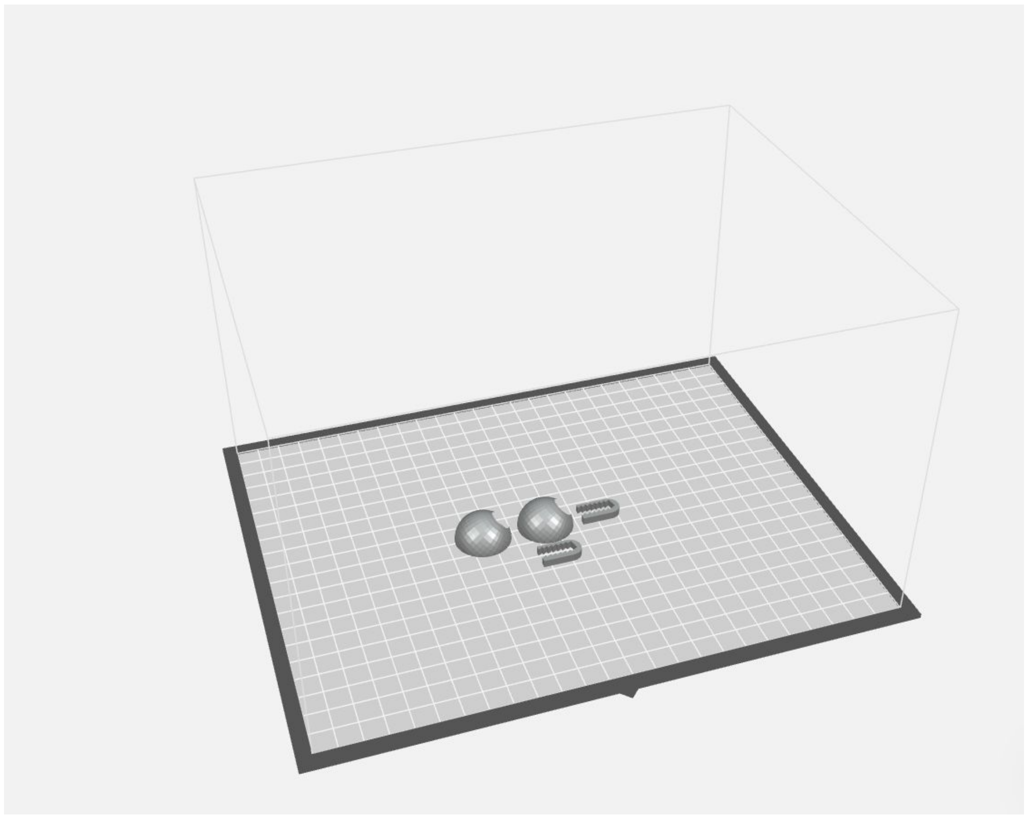
Once we had our finished cutouts, we began assembly.



We followed the instructional in fusion 360 and secured some of the loose pieces of the mask with elmers glue. Finally, we affixed the LEDs throughout the structure of the mask.

We also made 3D printed eyes for the Rhino from adapting the scale of some 3D assets we found online and slicing them to accommodate placement on the flat surface of our rhino head encasing.

}



Although the printing didn't come out exceptionally high quality on our three print trials, we decided they were high enough quality for our prototype and allowed light through the opaque transparent PLA filament we used.

Testing Our Microphone Sensor

We initially began our implementation understanding the connection between the microphone sensor and the Arduino. That was a pretty simple implementation, we did something like the following:

```
------------------------------------------------------------------------------------------------------------------------
int soundSensor = A0;
boolean LEDStatus=true;

void setup()
{
  pinMode (soundSensor, INPUT);
  pinMode (LED_BUILTIN, OUTPUT);
}

void loop() {
 int SensorData=digitalRead(soundSensor);
 if(SensorData==1){

   if(LEDStatus==false){
      LEDStatus=true;
      digitalWrite(LED_BUILTIN,HIGH);
   }
   else{
      LEDStatus=false;
      digitalWrite(LED_BUILTIN,LOW);
   }
 }
}
```

We did this to ensure we understood the HIGH and LOW values we would employ in determining the microphone's variability that would ultimately impact the sound reactive LED flicker.

Sound Reactive Implementation

To determine how to build a sound reactive LED array, we explored uses with Fast Fourier transform. We did this while experimenting with various libraries. We got stuck for a bit on this bit, so referred to some open source code from GitHub user NaturalNerd to help us out. That helped us with our first implementation (light strands up to LED 60 based on noise level).

Our next final design implementation cycles through different colors based on the frequency of the sound detected using the Open Source Trinket library. We repurposed some of the code to experiment with different modalities:

-

```
 #include <Adafruit_NeoPixel.h>

#define MIC_PIN    A0  // Microphone is attached to  pin 2 A1 on the Trinket
#define LED_PIN    9 // NeoPixel LED strand attached to pin 0 on Trinket
#define N_PIXELS   60  // number of pixels in LED strand
#define N          100  // Number of samples to take each time readSamples is called
#define fadeDelay  5  // delay time for each fade amount
#define noiseLevel 10 // slope level of average mic noise without sound

Adafruit_NeoPixel  strip = Adafruit_NeoPixel(N_PIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);

int samples[N];       // storage for a sample collection set
int periodFactor = 0;  // keep track of number of ms for period calculation
int t1 = -1;          // times of slope > 100 detected.
int T;               // period between times scaled to milliseconds
int slope;           // the slope of two collected data sample points
byte periodChanged = 0;

// Arduino setup Method
void setup()
{
   strip.begin();
   ledsOff();
   delay(500);

   displayColor(Wheel(100));
   strip.show();
   delay(500);
}

// Arduino loop Method
void loop()
{
   readSamples();
}


// Read  and Process Sample Data from Mic
void readSamples()
{
  for(int i=0; i<N; i++)
  {
     samples[i] = analogRead(MIC_PIN);

     if(i>0)
     {
       slope = samples[i] - samples[i-1];
     }
     else
     {
       slope = samples[i] - samples[N-1];
     }

     // Check if Slope greater than noiseLevel  - sound that is not at noise level detected
     if(abs(slope) > noiseLevel)
     {
       if(slope < 0)
       {
           calculatePeriod(i);

           if(periodChanged == 1)
           {
             displayColor(getColor(T));
```

```
            }
        }
      }
      else
      {
            ledsOff();
      }

    periodFactor += 1;
    delay(1);
  }
}

void calculatePeriod(int i)
{
        if(t1 == -1)
        {
            // t1 has not been set
            t1 = i;
        }
        else
        {
             // t1 was set so calc period
             int period = periodFactor*(i - t1);
             periodChanged = T==period ? 0 : 1;
             T = period;
             //Serial.println(T);

             // reset t1 to new i value
             t1 = i;
             periodFactor = 0;
        }
}

uint32_t getColor(int period)
{
  if(period == -1)
     return Wheel(0);
  else if(period > 400)
     return Wheel(5);
  else
    return Wheel(map(-1*period, -400, -1, 50, 255));
}


void fadeOut()
{
  for(int i=0; i<5; i++)
  {
    strip.setBrightness(110 - i*20);
    strip.show(); // Update strip
    delay(fadeDelay);
    periodFactor +=fadeDelay;
  }
}

void fadeIn()
{

  strip.setBrightness(100);
  strip.show(); // Update strip

    // fade color in
  for(int i=0; i<5; i++)
  {
    //strip.setBrightness(20*i + 30);
    //strip.show(); // Update strip
    delay(fadeDelay);
    periodFactor+=fadeDelay;
  }
}

void ledsOff()
{
  fadeOut();

  for(int i=0; i<N_PIXELS; i++)
  {
       strip.setPixelColor(i, 0, 0, 0);
  }
}

void displayColor(uint32_t color)
{

  for(int i=0; i<N_PIXELS; i++)
  {
       strip.setPixelColor(i, color);
  }

  fadeIn();
}

// Input a value 0 to 255 to get a color value.
// The colors are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos)
{
  if(WheelPos < 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

-

Discussion

Design

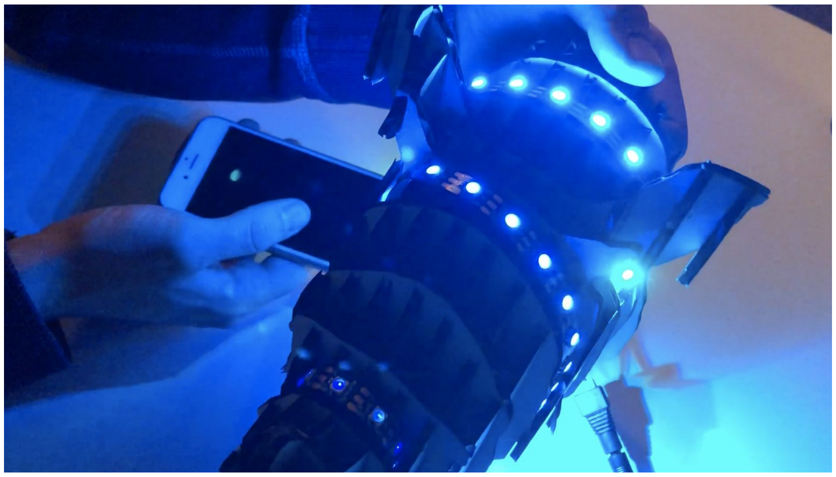
Overall, we were very content with the design choices we made with the Rhino Head; we felt like it was a very unique implementation.

The Rhino Head actually had a pretty sturdy design, and we were able to wrap the LEDs around the head to take it around without them falling off. For that, we felt we made great design decisions and think we could easily scale our project 2x or 3x and use a 144 set of LEDs if we were to make it more grand.

To help us understand how to loop through different effects and colors, we explored different NeoPixel / FastLED libraries. Using this, we experimented with different modalities to learn how to cycle through different loops and patterns.

The process we went through to land on our final design involved over 10 configurations. Some of those designs we considered include:

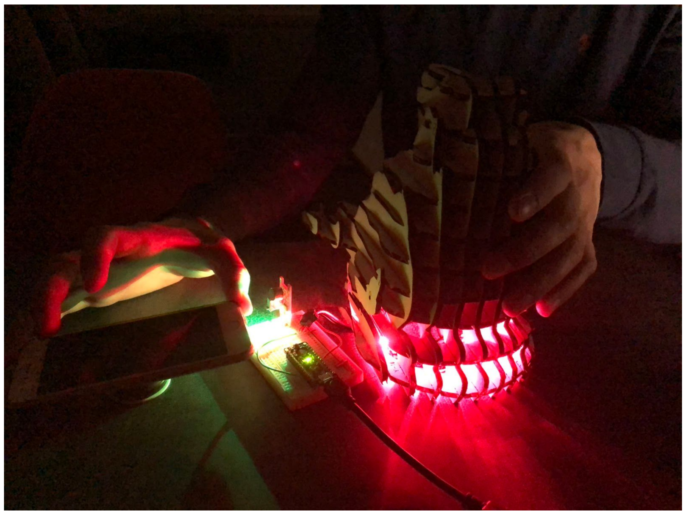
Wrapping the lights around the Rhino (this was our original sound-reactive implementation):



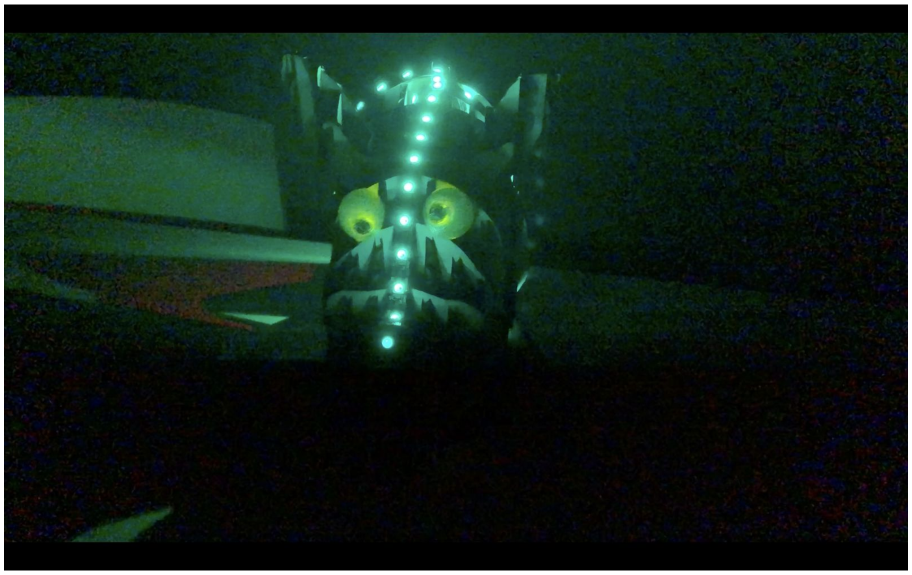
Wrapping the lights in a figure-8 to determine whether the light forms a wall projection:

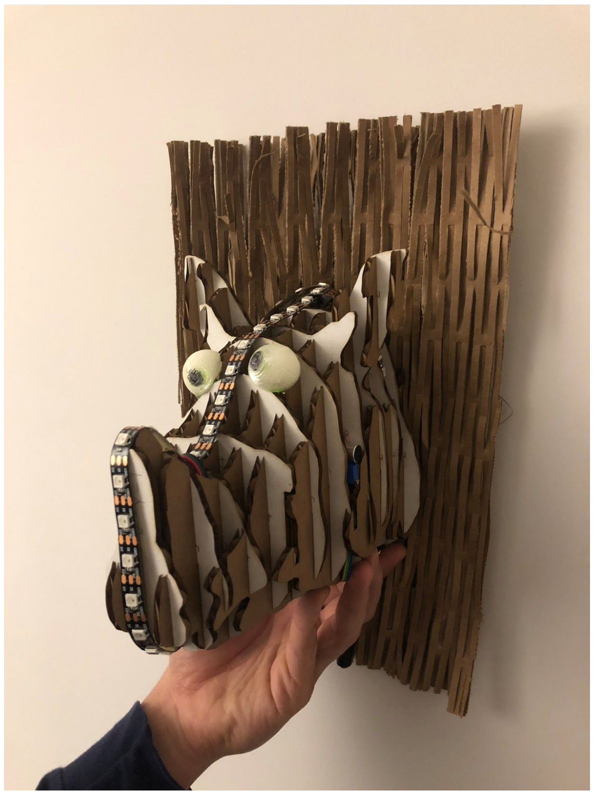
Wrapping the lights around the mouth of the Rhino:

Wrapping the lights around the base of the Rhino:



That’s how we landed on our **final design choice**:





We chose this design because we wanted to maintain an earthy, animal-like environment but the light still looks great at night. Additionally, you are able to insert a mini glowstick in the slots underneath the eyeballs for an added evening glow.

If you'd like to see our final implementation, you can view our demo video at: **http://bit.ly/CrazyRhino**

Taking It Further

To expand on our project, we had some ideas of what we would do in a future iteration to take the project further:

- Experiment with different light modalites (patterns, colors, etc.)
- Build the larger Rhino encasing (2x or 3x scaling)
- Experiment with powering the device through battery (so we can actually take it with us in public)

Sources

1: https://github.com/hansjny/Natural-Nerd/blob/master/arduino/soundsread2/sound_reactive.ino
2: https://github.com/codergirljp/Trinket-Color-By-Sound-RGB-NeoPixels

Thank you for a wonderful semester. We had so much fun in your class! :-)