Light Brush

Design Review

Team 34
Joseph Xiong jcxiong2
Willie Zeng wzeng4

ECE445
TA: Henry Duwe
October 14, 2016
Table of Contents

1.0 Introduction
   1.1 Statement of Purpose
   1.2 Objectives
      1.2.1 Goals and Benefits
      1.2.2 Functions and Features

2.0 Design
   2.1 Block Diagrams
   2.2 Block Description
      2.2.1 User Interface
         2.2.1.1 Outputs
         2.2.1.2 Inputs
         2.2.1.3 Controls
      2.2.2 Data Processing Unit
         2.2.2.1 Preprocess & IR LED Detection
         2.2.2.2 Store Coordinates & Create Circle
         2.2.2.3 IR Detection & Update Brush Size
         2.2.2.4 RGB Detection & Default Color & Coordinate Passing
         2.2.2.5 Display Current Frame and Check End Case
      2.2.3 Processing Hardware
         2.2.3.1 Microcontroller
         2.2.3.2 Camera
         2.2.3.3 Monitor
   2.3 Circuit Schematic
   2.4 Simulation
   2.5 Calculations
      2.5.1 IR LED Resistances
      2.5.2 RGB LED Resistances
      2.5.3 PWM Frequency

3.0 Requirements and Verification
   3.1 Requirements and Verifications Table
      3.1.1 Stylus
      3.1.2 Microcontroller
   3.2 Tolerance Analysis
   3.3 Ethics and Safety
3.3.1 Ethics
3.3.2 Safety

4.0 Cost and Schedule

4.1 Cost Analysis
   4.1.1 Labor
   4.1.2 Parts
   4.1.3 Grand Total

4.2 Schedule

5.0 Citations
1.0 Introduction

1.1 Statement of Purpose

Currently, drawing digitally is expensive because it requires platforms that are designed for gaming or personal computing. The Light Brush provides a cheaper and more direct way to draw digitally when compared to tablets, the Wii, and Xbox Kinect. The current design allows users to utilize a stylus and wirelessly draw onto a monitor. Colors and brush sizes are configured on the user-friendly stylus.

The Light Brush was originally designed as a glove but has been updated into a stylus for ease of use. The glove raised unnecessary concerns on button/circuit placement, which pushed for a much more reasonable design (a stylus) that will be more comfortable for the user. In addition, the user can adjust the color they want to draw with by controlling an RGB LED with potentiometers. There will also be a camera and microcontroller to see the location of the stylus and continuously update in real-time to locate the stylus, color of the RGB LED, and brush size IR LED. This information will be processed and then sent to a monitor which will display the output the user’s drawing. This provides a new product that anyone can use to draw with a canvas of almost any size.

1.2 Objectives

1.2.1 Goals and Benefits

- To be as accurate and as configurable as it’s expensive counterparts, while remaining inexpensive
- Drawing without using paper and a pencil/pen/marker
- Accurately and quickly updates the color, location, and the brush size of the stylus
- Inexpensive

1.2.2 Functions and Features

- To create an accurate and configurable drawing onto a monitor based on the user’s settings on the stylus.
- To display and constantly update/obtain the movement of the stylus

---

1 https://us-store.wacom.com/Catalog/Pen-Displays/Cintiq/Cintiq-27QHD-S01
2 http://www.nintendo.com/wiiu/buynow
● To provide a range of colors for the artist to select
● To display and constantly update the color
● To provide a range of brush sizes for the artist to select
● To display and update the brush size
2.0 Design

Figure 1: Design of the stylus

2.1 Block Diagrams

Figure 2: Block diagram of the device, the gray section represents the stylus
2.2 Block Description

2.2.1 User Interface

The User Interface module composes of all the aspects of the stylus. It involves the user inputs that will control the different features of the stylus and will signal the camera necessary data through the configuration of the LED output. The following are descriptions of blocks in the user interface:

2.2.1.1 Outputs

The outputs of the stylus will only be 2 IR LEDs and 1 RGB LED.

The first IR LED will be used for tracking purposes. The camera will see the location of this IR LED and send the corresponding coordinates to the microcontroller which will process the data and output correspondingly to the monitor.

The second IR LED will be used to control the brush size that the user is drawing with. To change brush sizes, the second IR LED will turn on then off and this blink of the light will be seen by the Data Processing Unit and understand that the brush size will change. There will be seven different brush sizes and each blink of the second IR LED
would cycle to the next largest brush size. If the current brush size is the largest one and the IR LED blinks, then the brush size would return back to the smallest size. To avoid confusion in tracking, the second IR LED can only be powered if the first IR LED is also powered. This way, instead of searching for the second IR LED to determine brush size, which can be difficult, the Data Processing Unit can determine a change in brush size when there are two IR LEDs on compared to one.

The RGB LED can be configured by the user to control what color they want to draw with. The color that the RGB LED shines will be seen by the camera and the Data Processing Unit can identify this color and use these values when outputting to the monitor.

2.2.1.2 Inputs

There will be six inputs to control the features of the stylus: 2 push buttons, 3 potentiometers, and 1 switch.

The first push button controls whether the first IR LED and the RGB LED light up. When this push button is pressed, both the first IR LED and RGB LED will be powered. This push button can essentially be used as the “drawing” button where the user can draw when it is pressed and stop when the button is let go.

The second button is to control the second IR LED. Pushing of this button will blink the second IR LED which will be seen by the camera and the brush size will be changed.

Both push buttons will be debounced using a resistor and capacitor as shown in Figure 7. The buttons need to be debounced to prevent the effects of noise.

The 3 potentiometers are used to control the color of the RGB LED. One potentiometer will be used to control the Red LED, another for Green, and the third for Blue. Turning the potentiometer allows for the corresponding LED to become brighter or dimmer and the three colors can merge which allows for any color to be chosen.

The switch is used to save a bit of power in the circuit. The RGB LED will be defaulted to off, but will turn on while the user is drawing by pressing the first push button (drawing button). The switch was added to allow the user to change colors and see the RGB LED while not drawing. When the switch is turned on, the RGB LED will be powered and can be seen even if the drawing button is not pressed. It will be on the user however to turn off the switch after changing colors to save power. The switch will also be debounced in a similar manner as the push buttons in Figure 7.

2.2.1.3 Controls
The stylus’ circuit will take the user inputs to control the output LEDs. The circuit schematic can be seen on Figures 6 and 8. The power of the stylus will be powered by a 9V battery with a 5V voltage regulator. This is to keep a steady source of power to the circuit minimizing any noise from the power supply as possible. A 5V voltage regulator was chosen, compared to other voltage values, because of different specifications and calculations needed for components later on in the circuit.

The next portion of the circuit involves the two push buttons and two IR LEDs. The first part is simple where if the first push button is pressed, the first IR LED is powered and shines. The second push button is wired so that power is only supplied to the second IR LED if both buttons are pressed which is what was described earlier to make tracking the stylus simpler on the algorithm. The resistors are needed for the IR LEDs to limit and control the optimal voltage values that the IR LEDs will operate.

To connect to the last portion of the circuit, 2 wires must be extended to reach the PWM (pulse width modulation) circuits. The first is the power for the circuits, from the output of the 5V voltage regulator, which will be called Vin. The second is for the logic needed to control the RGB LED. As described above, the RGB LED will be defaulted to off and will shine if the first push button (drawing button) is pressed or the user controlled switch is flipped on. So this second wire is the switch and first push button OR gated together and the output of the OR gate will be called input logic. The input logic will be equal to Vin if either the switch is flipped or push button is pressed.

The final and most complex portion of the circuit is the PWM which controls the RGB LED. A PWM controls a certain load by the use of the input power. The supply that is powering the load will change between high and low and the percentage of the time that the power is high compared to the total time is the duty cycle. So if the power is high for 90% of a cycle, it will have a 90% duty cycle. If the power is high for 40% of a cycle, it will have a 40% duty cycle. This PWM will control the RGB LED because the duty cycle of the power supply that powers an LED is proportional to the intensity of the LED. A 0% duty cycle supply that powers an LED will not power the LED and a 100% duty cycle supply that powers an LED will make the LED shine intensely.

The PWM will be implemented with a 555 timer which will control the oscillation of the power supply. A few of the important pins on the 555 timer are the trigger, threshold, discharge, and reset. The trigger is an active low input that starts the oscillation. So when the trigger has a low signal (below ⅓ of Vcc) the output will be high. The opposite pin is the threshold pin where it will cause the output to be low when this value is high. So when the threshold pin rises above ⅔ of Vcc, the output pin will become low. The discharge pin is used to help discharge capacitors between intervals. It is also in phase with the output so when the output is high, the discharge is also high and vice versa. One of the differences between the output and discharge pins is that the output pin can carry current both in and out but the discharge pin can only carry current.
out. The last important pin is the reset pin which will override any signal given by the threshold, trigger or any other input and set the output to low. The reset pin is also active low and will start a reset if it’s value is low. After the reset pin is powered high again, the PWM can only restart through the trigger pin becoming low once more.

The PWM works by tying the trigger and threshold pins together and adding a capacitor where the capacitor is wired between the trigger and threshold node and ground. The charging and discharging of this capacitor will give the 555 timer its oscillation. The capacitor will discharge across a resistor along with some diodes to keep the current flowing in the specified direction. The oscillation occurs since the threshold pin will start out low and this causes the output to be high. Then the capacitor starts to charge because of this high voltage until it reaches ⅔ of Vcc and the threshold pin turns the output low. The capacitor would then discharge across the resistor and the trigger node will become low again and restart the process. The user variability comes from the resistor as the resistor used to discharge the capacitor, is actually a potentiometer that can vary its resistance. If the resistance is very low, the duty cycle should decrease and the LED should become very dim or off. If the resistance is very high, the duty cycle should increase the the LED should shine brightly. The discharge pin is actually the one used to control the LED since the current would only leave one way which adds some protection to the circuit. The output pin is then sent to the variable resistor which, through the diodes, can charge or discharge the capacitor. The discharge pin is also connected to an NMOSFET which will be used as a voltage controlled current source. The gate of the NMOS will be connected to the discharge pin, the source of the NMOS is grounded, and the drain is connected to the RGB LED. When the transistor is on, current will flow through it and the RGB LED, powering it. The reason a 5V regulator was chosen is because the gate voltage of the NMOS is optimal around 5V.³

The added capacitor to the control voltage pin of the 555 timer is to help reduce the power input noise and the resistor and capacitor that are connected between Vcc and the discharge pin is to bring more stability to the discharge pin which is needed to keep the NMOS operating in the desired region.

All together there will be 3 PWM circuits, one for each color, where each circuit will receive the input voltage from the 5V voltage regulator for Vin and the input logic for the reset pin on the 555 timer.

2.2.2 Data Processing Unit

Figure 4: More In-depth flowchart of the Data Processing Unit

Intro to Data Processing Unit

Before we hit the FOR loop, we need to initialize the Camera and see if it is available to turn on. If the camera is not detected, the code should continuously look for the camera. Once the camera is detected, we use the camera to take a video, but although it is a video, we are processing every frame of the video. The 0.1 seconds of wait time is for the camera to turn on. We also need to initialize the default color of the brush, which will be black. The initial size of the brush will be size 1 (which we will determine how big size 1 will be by testing).
2.2.2.1 Preprocess & IR LED Detection

Within the Data Processing Unit we implement a FOR Loop to continuously process the frames of the video until the video is terminated by the user. Each frame is then preprocessed by downsizing the image. By downsizing the image, we are able to process the frame faster, leading to an increase in FPS. Also in this preprocess stage, we will blur the image to reduce high frequency noise. Then the image is converted from RGB to HSV (Hue, Saturation, Value) color space so we can extract the IR LED. The big reason is that it separates color information (chroma) from intensity or lighting (luma). Because the value parameter is separated, we can construct a histogram or thresholding rules using only saturation and hue. This in theory will work regardless of lighting changes in the value channel. Even by singling out only the hue you still have a very meaningful representation of the base color that will likely work much better than RGB. The end result is a more robust color thresholding over simpler parameters. Hue is a continuous representation of color so that 0 and 360 are the same hue which gives you more flexibility with the buckets you use in a histogram.

![Figure 5: HSV representation. Photo obtained from https://en.wikipedia.org/wiki/HSL_and_HSV](https://en.wikipedia.org/wiki/HSL_and_HSV)

2.2.2.2 Store Coordinates & Create Circle

Once the IR LED location is obtained after preprocessing the frame. The X,Y coordinates are stored as a list so that we can create a radius around the stored point. This circle/radius will be larger if the IR LED is closer to the camera and smaller if the LED is farther from the camera. The purpose of the circle is so that the software only searches for the RGB color inside this circle area. Our stylus will have an RGB LED

---

4 https://en.wikipedia.org/wiki/Gaussian_blur
5 https://en.wikipedia.org/wiki/HSL_and_HSV
adjacent to the IR LEDs. The circle will allow the microcontroller to ignore any background colors that may affect the RGB detection.

2.2.2.3 IR Detection & Update Brush Size

Once the circle is made, we see if the IR LED can be detected using the HSV method described earlier. If TWO IR LEDs are detected, then we increment and update the default brush size by 1. If there is only ONE IR LED, then we use the default brush size. When we increment the brush size, if the brush size is at its maximum value, we set it back down to the initial size 1. This allows the user to cycle through the brush sizes. You may be asking, what is size 1 or size 2, etc.? We will deduce the size/thickness of the brush through testing and seeing, which is the preferable start size and incremented size. Furthermore, the thickness of the line drawn onto the frame depends on how the drawing function in openCV works (we will specifically use cv2.line)\(^6\) It takes in a value and draws a line, the thickness of this line depends on that value. This value will be related to the brush size value. At this stage, there is no drawing taking place, it happens at the next stage.

2.2.2.4 RGB Detection & Default Color & Coordinate Passing

Once the circle is made, we filter the RGB within the circle of the image and see if we get a strong enough color by using HSV (this will take many tests). If there is no RGB detection then we use our default color, which will be initialized in the beginning of the code as black. If there is a RGB detection, then our default black color, or the previous color, will be updated to the corresponding color. Finally, we use the X,Y coordinates that we stored and we draw at this coordinate. We also use the brush size to draw at this coordinate. Whatever the thickness may be, the drawing should be implemented by using the drawing functions of openCV, such as cv2.line.

2.2.2.5 Display Current Frame and Check if the End Case is Satisfied

The displaying of the frame of the video depends on the resolution that the video is captured in. For this project we are capturing the video in 640x480 and therefore, the frame displayed will be 640x480. The X,Y coordinates of the IR light will be captured from this 640x480 coordinate system.

The current end case is based on the IR LED’s movement. If the IR light has not moved in 1 minute then power off the Raspberry Pi. This will be done by analyzing each

\(^6\) [http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html](http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html)
frame of the video for 1 minute and if the X,Y coordinates of the IR light has not moved or is OFF for 1 minute, power off.

2.2.3 Processing Hardware

2.2.3.1 Microcontroller

The microcontroller is a Raspberry Pi 3 Model B. It has an output HDMI port, and input ribbon cable for the camera. All of the software explained in Section 2.2.2 will be programmed on the Raspberry Pi and can hold a microSD card for memory. It also uses a microUSB for it’s power supply.

2.2.3.2 Camera

The camera being used is the Raspberry Pi camera module v2 which supports 1080p30 and 720p60 video. It connects to the Raspberry Pi through a ribbon cable which is also its power supply. The camera will constantly take pictures of the LEDs on the stylus (assuming the stylus is inside the view of the camera) to be processed by the microcontroller which will be how the processing unit “tracks” the stylus.

2.2.3.3 Monitor

The monitor does not have to be specific to this project, so any monitor will be able to be used assuming it supports HDMI input.
2.3 Circuit Schematic

Figure 6: Circuit Schematic of Stylus

Figure 7: Debounced Push button circuit

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
<td>Ground Reference</td>
</tr>
<tr>
<td>2</td>
<td>Trigger</td>
<td>Triggers the start of the 555 timer running. When this pin is triggered, the output becomes high. The trigger pin is active low so when the trigger is low (below ⅓ of Vcc) the output is high</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>Output</td>
<td>Output voltage pin. The high voltage of the output is usually 1.7V lower than Vcc</td>
</tr>
<tr>
<td>4</td>
<td>Reset</td>
<td>Overrides all other commands and changes the output to low. Reset is active low.</td>
</tr>
<tr>
<td>5</td>
<td>Control Voltage</td>
<td>Used to level out fluctuations in supply voltage</td>
</tr>
<tr>
<td>6</td>
<td>Threshold</td>
<td>Changes the output to low when threshold is above $\frac{2}{3}$ of Vcc</td>
</tr>
<tr>
<td>7</td>
<td>Discharge</td>
<td>Used to control the timing interval</td>
</tr>
<tr>
<td>8</td>
<td>Vcc</td>
<td>Power Supply. Min and max voltage of 4.5V to 16V</td>
</tr>
</tbody>
</table>

*Table 1: Pin layout for 555 Timer (LM555)*

Figure 8: Circuit Schematic of PWM
2.4 Simulation

Figure 9: Schematic of PWM simulation

Figure 10: Snapshot of Discharge signal of the PWM High Duty Cycle ~ 100%
The signal is HIGH for ~3.2ms
The signal is LOW for ~0.0005ms
Duty cycle of 99.98%
2.5 Calculations

2.5.1 IR LED Resistances\textsuperscript{8}

100mA Forward Current
1.5V Forward Voltage

\[ V = IR \]

5\text{V} - 1.5\text{V} = 3.5\text{V} \]

\[ \frac{3.5\text{V}}{.1\text{A}} = 35\Omega \]

\[ Power = VI = .15\text{W} \]

2.5.2 RGB LED Resistances\textsuperscript{9,10}

NMOS Saturation Current = .03A

NMOS \( V_{DSon} = 1\text{V} \)

Ideal Red LED Voltage = 2V

Ideal Blue and Green LED Voltage = 3.2V

Red: \( V_R = V_{in} - V_{LED} - V_{DS} = 5 - 2 - 1 = 2\text{V} \)

Red Resistor = \( \frac{V_R}{I_D} = 66.67\Omega \)

\textsuperscript{8} \url{https://cdn-shop.adafruit.com/datasheets/IR333_A_datasheet.pdf}

\textsuperscript{9} \url{http://www.nteinc.com/specs/400to499/pdf/nte464.pdf}

\textsuperscript{10} \url{https://cdn-shop.adafruit.com/product-files/2739/p2739.pdf}
Red LED Power $= V_{LED}I_D = .06W$

Green, Blue: $V_R = V_{in} - V_{LED} - V_{DS} = 5 - 3.2 - 1 = .8V$

Green/Blue Resistor $= \frac{V_R}{I_D} = 26.67\Omega$

Green/Blue LED Power $= V_{LED}I_D = .096W$

2.5.3 PWM Frequency\(^{11}\)

LEDs have a noticeable flicker around 60Hz

100Hz should be high enough to have a less noticeable flicker

$$f = \frac{1}{RC}$$

Choose 100kΩ potentiometer

$$100 = \frac{1}{100k \cdot C} \Rightarrow C = .1\mu F$$

\(^{11}\) https://en.wikipedia.org/wiki/Flicker_fusion_threshold
### 3.0 Requirements and Verifications

#### 3.1 Requirements and Verifications Table

##### 3.1.1 Stylus (40 pts.)

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Output of PWM has duty cycle &lt;10% when potentiometer is at minimum resistance (5 points)</td>
<td>1) Isolate PWM circuit and confirm that the potentiometer is at it’s minimum resistance with a multimeter. Then use an oscilloscope to measure the voltage of the discharge node of the 555 timer vs time. Then calculate the duty cycle by finding the period of the wave and the amount of time that the signal is high.</td>
</tr>
<tr>
<td>2) At all levels of resistance for the potentiometer, the frequency of the PWM output is never lower than 70Hz (10 points)</td>
<td>2) Measure the period of the discharge node of the 555 timer of the PWM at the min and max resistances of the potentiometer with an oscilloscope. Then calculate the frequency with the measured period.</td>
</tr>
<tr>
<td>3) All seven mixtures of light (Red, Green, Blue, Yellow, Cyan, Magenta, White) are able to be outputted by the RGB LED through only the inputs of the three potentiometers and can stay at that color for 5 seconds without visible flickering (No instances of the LED being off) (15 points)</td>
<td>3) Each color will be outputted by turning the corresponding potentiometer (i.e. to test Red, the red potentiometer will be turned so the red LED will be a maximum intensity and the other LEDs will be at no intensity) The camera will then take a picture of the RGB LED and filter the picture through a colored mask and if the LED still shows in the output picture then the color is satisfactory. (If the color red is being tested, the picture of the RGB LED will be filtered through a red mask (color value greater than 150) and if the output does not filter out the LED, then the test is passed.)</td>
</tr>
<tr>
<td>4) The first IR LED (tracking IR LED) can stay on for 1 minute without</td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td>Verification</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1) Capture and process video of at least 35 frames per second. (5 points)</td>
<td>1) Implement a small script in the code in openCV (CV_CAP_PROP_FPS) to capture the frames per second when recording at a resolution of 640x480. The FPS should never drop below 35 fps, the 5 extra fps is to take into account time/lag from calculations.</td>
</tr>
<tr>
<td>2) Track IR LED accurately (within 10 pixels from the actual coordinate of the IR by comparing it to a before processed image) up to 6ft (20 points)</td>
<td>2) Move IR LEDs horizontally and vertically from 1 foot, 5 feet, and 6 feet and display the X,Y coordinates for debugging purposes. The X coordinate should never be above 320 or below -320 and the Y coordinate should never be above 240 or below -240. (The origin is in the middle of the screen) We will compare the X,Y values to the values obtained by saving an image before it gets processed and finding the location of the IR LED.</td>
</tr>
<tr>
<td>3) Able to detect the color of the RGB values from the LED (The Hue and Value category should be ±1 units)</td>
<td>3) Use the microcontroller to pick up the RGB values of a high definition picture. Translate the RGB values</td>
</tr>
</tbody>
</table>
and the saturation should be ±5 units.) when the current is between 25mA and 30mA (20 points)

4) Find the most accurate RGB detection shape to differentiate the LED values from the background colors. Since different shapes affect the HSV values, we want HSV values to be ±2 degrees in hue and /units to the HSV values obtained from a higher quality image. (15 points)

to HSV values and see if a similar color from the stylus can be mimicked or picked up. Compare the Hue, Saturation, and Value values to the Hue, Saturation, and Value values from a high definition picture of the same RGB LED captured by another, superior quality camera. The Hue and Value categories should be ±1 units and the saturation should be ±5 units.

4) Set the RGB LED to the same color as the background (alternate to different colors every trial) and change the area we want to obtain the HSV values. Compare the HSV values of the RGB found in this area to the HSV values found in a higher quality picture of the same scenario. We want HSV values to be ±2 to the HSV values obtained from a higher quality image. Run through 5 trials with every color. (Further explained in Tolerance Analysis)

3.2 Tolerance Analysis

A critical component of this project is the software used in the microcontroller. The software is the most important component because it encompasses all the aspects of reading and understanding input image data, processing it, and outputting it correctly to the monitor. The image processing portion needs to be accurate in identifying the correct location of the stylus and color of the IR LED in order for the project to properly function.

Since accuracy of the image processing software is crucial to the project, there are a few requirements for the accuracy of the software. The first is for capturing and processing videos of at least 35 frames per second. This is to take into account the calculations, tracking, and drawing methods used in the code. The lower bound of 30 frames per second was chosen to make sure the output images to the monitor is
smooth to the user. A typical movie plays around 24 fps\textsuperscript{12} so 35fps should give enough
tolerance to account for processing. This will be tested by running a simple script to find
the fps. Confirming that the frame rate is high enough to have a smooth output is
important in having a pleasant user experience.

After running trials and measuring the average FPS and time allocated for certain
processes we can accurately predict the FPS drops on certain functions. For example,
when the camera is just recording without an FPS cap, we get more than 300 FPS.
However, once the image is converted to HSV, the FPS drops to an average of 34.8
FPS. Then, once the mask (for LED color detection) is applied we get 21.9 FPS. As you
can see, the number of functions (which use “expensive” algorithms) we can call is
going to be limited if we are going to keep the FPS above 30 FPS. A solution currently
in the works is only masking the specific area (a range in the array) for the colored LED
after obtaining the IR LED. Unfortunately, an FPS drop will occur for the initial IR LED,
but further FPS drop can be limited once we only mask that certain space in the array.
The space in the array will have a radius of around 30 pixels and the x,y coordinates will
constantly be tracked. Therefore, the stage that is in the works is only masking the
space that is within this 30 pixel radius from the center x,y coordinates. An additional
solution is to check the mask every other frame or every third frame instead of every
frame (essentially dividing the amount of processes by half or by a third).

The above FPS was calculated by obtaining the number of frames and dividing it
by the seconds that it took to obtain the frames.

Another issue that has been discovered is that the code may incorrectly track the LEDs.
It may confuse the luminosity of the LED with other LED’s. Fortunately, a trend that has
been found is that the radius obtained by tracking the LED’s increases in value based
on the luminosity and the distance of the LED. If the LED is within 1 foot from the
camera, the radius obtained is ~300 (it constantly adjusts). If the LED is 5ft away, the
radius obtained is around 30 pixels. This can be easily solved by setting a range for the
radius to be detected. However, we must notify users that the “brush” cannot be within 3
feet or the detection may be skewed.

The second requirement is to be able to accurately track the IR LED. This will be tested
by having an input image with the IR LED at a certain location, then after processing,
comparing the output image and the located IR LED to the location of the IR LED of the
original input image. The comparison will be done by either using a software tool like
MS paint to find the coordinates of the before and after images and comparing these
numbers or the other method for comparison is using the terminal of the microcontroller
to overlay one image over the other to find differences. By overlaying one image over

\textsuperscript{12} \text{http://www.100fps.com/how_many_frames_can_humans_see.htm}
the other, we can test how far away the processed location is from the original location. Being able to accurately locate the IR LED is key for this project since this accuracy is directly responsible for the accuracy of the location of the output image on the monitor to simulate the user’s motions accurately.

The next requirement is to accurately find the RGB values of the RGB LED. To test the accuracy, a high definition picture will be used as an input image of the software and the HSV values will be found. Then RGB LED will be adjusted to match the color of the high definition picture as best as possible and the camera will take a picture of the RGB LED and the same values will be found from the pictures. The HSV values of the two pictures will be compared and will need to fit within the tolerance. Being able to accurately detect the color of the RGB LED is important to the project because the user will want the color they choose to draw with to be the actual color that is outputted to the monitor.

The last is identifying the correct HSV values of the RGB LED. The software will use a circle, centered at the located IR LED, to search for the RGB LED values. The problem is which shape (not necessarily a circle) would find the most accurate HSV value within the tolerance chosen in the R&V Table. Because the movement of the user can be very quick, it will be advantageous to try other shapes to use in addition to trying to use a circle. The background may be color values may be picked up. For example, a circle may be advantageous to use because it has no edges, so it has a less chance in picking up background colors. However, if the captured video is lagging slightly, a hexagon may prove advantageous with its edges to detect the RGB as it moves slightly faster than our IR detection. Testing which shape that gives the most accurate values will be important in accurately displaying the color to the output.

![Figure 12: Illustrations of how the shape will help differentiate the LED’s from the background](image-url)
3.3 Ethics and Safety
3.3.1 Ethics

In regards to safety, we have created the following statement to abide by:

“We will communicate honestly and with interest to improve our project, while following and properly citing guidelines, standard operating procedures, and datasheets.”

Issues that may arise during the development of our project will always center around misunderstanding. Fortunately, due to our lab notebooks and our clear schedule on which sections each member is responsible for, misunderstanding, plagiarism, and dishonesty can be avoided. Additionally, by citing datasheets, plagiarism can be avoided. By using guidelines, damage to property can be avoided. This includes disconnecting the power when changing a circuit and handling/moving the microcontroller. Furthermore, by using simulations and calculations power issues can be avoided by making sure the power source is able to support all the necessary modules. Following these rules and guidelines, while also keeping our integrity in avoiding plagiarism will be the standard of how to keep our project ethical. Whenever we need help we will seek help through our TA and maintain a strong communication with our TA. With that being said, our project follows the following IEEE Code of Ethics:

13

[1] to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

[3] to be honest and realistic in stating claims or estimates based on available data;

[5] to improve the understanding of technology; its appropriate application, and potential consequences

[6] to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

[7] to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

[10] to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

---

13 http://www.ieee.org/about/corporate/governance/p7-8.html
3.3.2 Safety

Our Drawing Stylus has a low power consumption, therefore, there are only some cases where certain parts can be harmful. However, because the Stylus utilizes LEDs, it is crucial to state certain health risks. These cases should be taken considered:

1. Do not look directly into LED to avoid eye discomfort and stress to the retina\(^\text{14}\)
2. Do not look directly into the IR LED to avoid eye damage\(^\text{15}\) (Make sure the stylus is off when inspecting)
3. Make sure battery is connected properly (positive terminal and negative terminal correctly positioned)
4. Do not leave the battery at a high temperature environment. High temperature batteries may lead to explosion and fire
5. Do not use battery in a high magnetic field
6. Avoid using the camera and microcontroller in high magnetic fields
7. Batteries should not be charged
8. Batteries should be inspected before usage
9. Avoid any liquid substance around the Stylus
10. Avoid liquid and metal contact to the microcontroller and camera
11. Caution, the microcontroller may be hot to touch
12. Before touching the microcontroller or camera, de-static yourself.

# 4.0 Cost and Schedule

## 4.1 Cost Analysis

### 4.1.1 Labor

<table>
<thead>
<tr>
<th>Name</th>
<th>Hourly Rate</th>
<th>Total Hours Invested</th>
<th>Total Cost (Rate<em>2.5</em>Hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Willie Zeng</td>
<td>$35.00</td>
<td>200</td>
<td>$17,500.00</td>
</tr>
<tr>
<td>Joseph Xiong</td>
<td>$35.00</td>
<td>200</td>
<td>$17,500.00</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td></td>
<td><strong>$35,000.00</strong></td>
</tr>
</tbody>
</table>

### 4.1.2 Parts

<table>
<thead>
<tr>
<th>Item</th>
<th>Manufacturer</th>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 3 Model B</td>
<td>Raspberry Pi</td>
<td>1</td>
<td>$36</td>
</tr>
<tr>
<td>microSD</td>
<td>SanDisk</td>
<td>1</td>
<td>$8</td>
</tr>
<tr>
<td>Duracell 9V battery</td>
<td>Duracell</td>
<td>2</td>
<td>$8</td>
</tr>
<tr>
<td>Raspberry Pi Camera Module</td>
<td>Raspberry Pi</td>
<td>1</td>
<td>$25</td>
</tr>
<tr>
<td>IR LED (IR333-A)</td>
<td>Everlight</td>
<td>2</td>
<td>$12</td>
</tr>
<tr>
<td>RGB LED (259RGBM5C-013)</td>
<td>LuckyLight</td>
<td>1</td>
<td>$2</td>
</tr>
<tr>
<td>Potentiometer (100kΩ)</td>
<td></td>
<td>3</td>
<td>$2</td>
</tr>
<tr>
<td>Resistors</td>
<td></td>
<td>9</td>
<td>&lt;$1</td>
</tr>
<tr>
<td>Button (SPST Push Button)</td>
<td></td>
<td>2</td>
<td>$3</td>
</tr>
<tr>
<td>Switch</td>
<td></td>
<td>1</td>
<td>$1</td>
</tr>
<tr>
<td>Description</td>
<td>Source</td>
<td>Quantity</td>
<td>Price</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------------</td>
<td>----------</td>
<td>--------</td>
</tr>
<tr>
<td>Diode (1N5818)</td>
<td>Diodes Incorporated</td>
<td>6</td>
<td>$0.48</td>
</tr>
<tr>
<td>7432 (OR Gate)</td>
<td>Fairchild Semiconductor</td>
<td>1</td>
<td>$0.41</td>
</tr>
<tr>
<td>SK9158 NMOS</td>
<td>NTE Electronics, Inc.</td>
<td>3</td>
<td>$3</td>
</tr>
<tr>
<td>555 Timer (LM555)</td>
<td>Texas Instruments</td>
<td>3</td>
<td>$0.75</td>
</tr>
<tr>
<td>Capacitors</td>
<td></td>
<td>9</td>
<td>$9</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>$111.64</td>
</tr>
</tbody>
</table>

4.1.3 Grand Total

<table>
<thead>
<tr>
<th>Section</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labor</td>
<td>$35,000.00</td>
</tr>
<tr>
<td>Parts</td>
<td>$111.64</td>
</tr>
<tr>
<td>Grand Total</td>
<td>$35,111.64</td>
</tr>
</tbody>
</table>

4.2 Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Tasks</th>
<th>Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/12-9/18</td>
<td>Finalize Proposal Section 1</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Proposal Section 2</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Proposal Section 3</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Proposal Section 4</td>
<td>Willie</td>
</tr>
<tr>
<td>9/19-9/25</td>
<td>Select 2 Cameras to choose from and write down Specs</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Select RGB LED Lights and IR LED lights and write down specs</td>
<td>Willie</td>
</tr>
<tr>
<td>Date Range</td>
<td>Task Description</td>
<td>Responsible Person</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>9/26-10/2</td>
<td>Select microcontroller</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Select power sources</td>
<td>Willie</td>
</tr>
<tr>
<td>9/26-10/2</td>
<td>Prepare Design Review first part</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Prepare Design Review second part</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Purchase microcontroller and LED’s</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Purchase camera and power supply</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Start glove circuit with RGB LED</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Check if both finished Laboratory Safety Training</td>
<td>Joseph</td>
</tr>
<tr>
<td>10/3-10/9</td>
<td>Program microcontroller with IR Camera</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Implement IR LED lights into glove circuit</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Get / utilize shop for glove design</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Work on glove circuit with LED light functionality with buttons and sliders</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Test power source compatibility with signaling module</td>
<td>Joseph</td>
</tr>
<tr>
<td>10/10-10/16</td>
<td>Finalize glove circuit with Power Source</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Test if Microcontroller is receiving data</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Test Power Source for monitor, microcontroller, and camera</td>
<td>Willie</td>
</tr>
<tr>
<td></td>
<td>Ensure functionality of sending signals by pressing user controls (buttons and potentiometer) on the glove circuit</td>
<td>Joseph</td>
</tr>
<tr>
<td>10/17-10/23</td>
<td>Code Microcontroller to output to Monitor</td>
<td>Joseph</td>
</tr>
<tr>
<td></td>
<td>Run tests to see if Microcontroller can display location of glove to monitor</td>
<td>Willie</td>
</tr>
<tr>
<td>Date</td>
<td>Task Description</td>
<td>Responsible</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>10/24-10/30</td>
<td>Finish Individual Progress reports</td>
<td>Both</td>
</tr>
<tr>
<td>10/31-11/6</td>
<td>Start working on Presentation</td>
<td>Willie</td>
</tr>
<tr>
<td>11/7-11/13</td>
<td>Fix errors and edge cases/bugs for Hardware</td>
<td>Willie</td>
</tr>
<tr>
<td>11/14-11/20</td>
<td>Finalize Presentation</td>
<td>Joseph</td>
</tr>
<tr>
<td>11/21-11/27</td>
<td>Prepare Final paper</td>
<td>Willie</td>
</tr>
<tr>
<td>11/28-12/4</td>
<td>Demonstration</td>
<td>Both</td>
</tr>
<tr>
<td></td>
<td>Finalize Final Paper</td>
<td>Both</td>
</tr>
</tbody>
</table>
5.0 Citations


