

# LED Brush Pro

Customizable LED for Light Painting Photography

Kyoungyeon Cho

Seungjun Cho

Jihong Min

Final Report for ECE 445, Senior Design, Spring 2017

TA: Eric Clark

3 May 2017

Project No. 42

## Table of Contents

1. Introduction .....	4
2. Design Procedures.....	5
2.1. Light System .....	5
2.1.1. Digital LED Strip Module .....	5
2.1.2. LED Case Module.....	5
2.2. Power System .....	5
2.2.1. NiCd Battery Charger Module.....	5
2.2.2. NiCd Battery Module.....	5
2.2.3. DC/DC Buck Converter Module.....	5
2.3. User Interface .....	5
2.3.1. Android Smartphone & Application.....	5
2.3.2. Battery Status Bar .....	6
2.3.3. Rotary Encoders and Power Button.....	6
2.4. Microcontroller System .....	6
2.4.1. Atmega328p Microcontroller.....	6
2.4.2. HC-05 Bluetooth Module .....	6
3. Design Details.....	6
3.1. Light System .....	6
3.1.1. LED Case Module.....	6
3.2. Power System .....	7
3.2.1. NiCd Battery Charger Module.....	7
3.2.2. DC/DC Buck Converter Module.....	8
3.3. User Interface .....	11
3.3.1. Android Smartphone and Application .....	11
3.3.2. Rotary Encoders and Power Switch .....	12
3.3.3. Battery Status Bar .....	13
3.4. Microcontroller Systems.....	14
3.4.1. Atmega328p Microcontroller.....	14
3.4.2. HC-05 Bluetooth Module .....	15
4. Verifications .....	16
4.1. Power System .....	16
4.1.1. NiCd Battery Charger Module.....	16

4.1.2.	NiCd Battery and Battery Status Bar .....	16
4.1.3.	DC/DC Buck Converter Module.....	17
4.2.	User Interface .....	18
4.2.1.	Android Application .....	18
4.2.2.	Rotary Encoders and Power Switch .....	18
4.3.	Microcontroller System .....	18
4.3.1.	Atmega328p Microcontroller and HC-05 Bluetooth Module .....	18
5.	Costs .....	19
	Adafruit NeoPixel Digital RGB LED Strip 144 LED.....	19
6.	Conclusion .....	20
7.	Appendix .....	21
7.1.	Requirements and Verifications .....	21
8.	References.....	32

# 1. Introduction

Light painting is a photographic technique, where the photographer sets the camera to have long exposure and moves the light around to paint the light onto an object. Currently, the photography equipment required to perform professional light painting is expensive. Our client, Rick Kessinger, needed an inexpensive solution for the lighting system for the light painting, that allows the photographer to be able to change the gradient, brightness, and color temperature easily.

From this, we set up our high-level requirements as follows:

- We must be able to control the brightness and color of each segment of the LED strip in precise detail.
- The LED state updates must be fast and smooth enough such that the blinking is not visible to the human eye, or a camera during a long exposure photograph.
- We must be able to power the LED system using a rechargeable battery for at least 15 minutes.

Since the design review, some small changes were made in the implementation and design. First, we have removed the linear voltage regulator, since it was not necessary to step down 5V to 3.3V, to power the microcontroller and Bluetooth module, as they can also operate on 5V input voltage. Also, another change in the design decision was to remove the physical battery status bar but integrate it into the Android application, because the physical battery bar on the Light Brush Pro system may disrupt the photography.

Figure 1.1 shows the modified block diagram of the Light Brush Pro system. It is divided into 4 main subsystems, microcontroller system, power system, user interface, and light system.

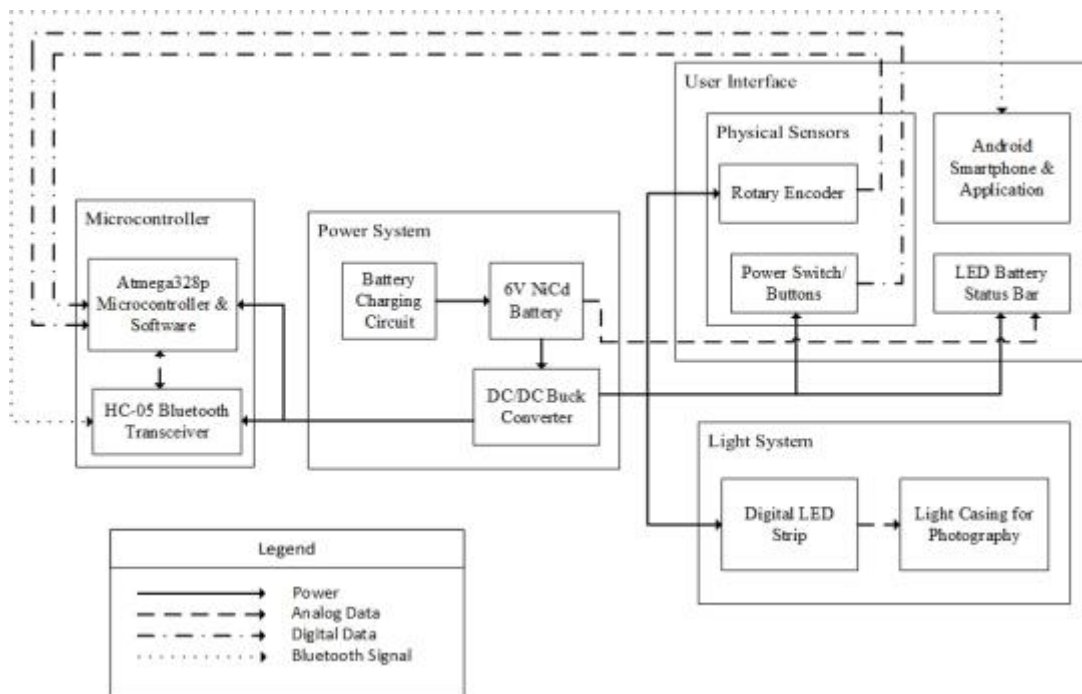


FIGURE 1.1. BLOCK DIAGRAM OF LIGHT BRUSH PRO SYSTEM

## 2. Design Procedures

### 2.1. Light System

The light System is composed of the digital LED strip module and the LED case module. Our client's prototype LED strip was analog and therefore would be constantly on full brightness. The fading effect was met by blocking out the light with multiple layers of diffusion papers. Also, the number of modes of the system was limited and changing gradient options required manual labor of moving around diffusion paper. Using an individually addressable digital LED strip allows us to enjoy more lighting modes while being more power efficient by turning off unnecessary LEDs.

#### 2.1.1. Digital LED Strip Module

The microcontroller determines the temperature, gradient, location, and brightness of the densely packed WS2812B digital LED strip. To make the gradient change as smoothly and specifically as possible, we chose the most densely packed LED with 144 LEDs/m. Since our system will be 1.2192m (4ft) long, 175 LEDs will fit into our system.

#### 2.1.2. LED Case Module

For light painting, it is crucial that the lighting is smooth and not pixelated, and therefore the LED case will have diffusion papers and plexiglass to diffuse the lights. Also our client wanted the case to have a handle at one end such that a wheeled LED holder can clamp on it.

### 2.2. Power System

#### 2.2.1. NiCd Battery Charger Module

Using the MAX713 NiCd Battery fast-charge controller, we will be able to fully charge our NiCd batteries in 2.5 hours. Our Battery fast charges at a standard charge rate of C/4. Nonetheless, overcharging the NiCd battery can still be hazardous. However, the MAX713 chip can terminate fast charge by detecting when the voltage slope becomes negative.

#### 2.2.2. NiCd Battery Module

Our high-rate NiCd battery must supply a steady 6V at a maximum current of 5A for at least 15 minutes. We chose the NiCd battery over the lead-acid battery and the lithium-ion battery because NiCd batteries are more stable and sturdy than li-ion batteries. Also, the NiCd battery is more charge dense than the lead-acid battery, making it smaller and lighter, perfect for our application.

#### 2.2.3. DC/DC Buck Converter Module

The TPS53353 buck converter takes 6V input from the NiCd battery and supplies a steady 5V to the LED strip, the microcontroller, the Bluetooth transceiver and a few other modules. The buck converter can handle a maximum output current of 20A, but for our application, we only need to draw 5A. We used a buck converter instead of a linear regulator so that this photography lighting tool can be used steadily for a long time.

### 2.3. User Interface

#### 2.3.1. Android Smartphone & Application

The main responsibility of the Android application is to provide the users with intuitive UI components for controlling the LED light. The application allows controlling the color temperature, brightness, color, and gradient cursor locations. Furthermore, it integrates Bluetooth settings, to select the Light Brush Pro device within the app, and save/load settings, to allow saving the current LED settings in database for future use. These components are specifically chosen to allow users to control LED light more intuitively. Also, the application sends data with smaller

bandwidth after every specific time interval to prevent congestion, which may prevent the transition in the LED light states to appear smooth.

### 2.3.2. Battery Status Bar

The battery voltage ranges from 7V to 5V and the analog input pin of our microcontroller can only measure voltages between 0v and 5V. A differential amplifier is needed to bring the battery output voltage within range of the microcontroller input.

### 2.3.3. Rotary Encoders and Power Button

The rotary encoder will be used to change the settings of the LED system and the power button will be used to turn on or turn off the LEDs. We will use 12-step rotary encoders that also have a push functionality. One rotary encoder, when not pushed, will alter the brightness. When pushed, it will change the color temperature. The other two rotary encoders, when not pushed, will change the location of the left and right ends of the solid light. When pushed, the two rotary encoders will change the location of the left and right ends of the gradient/faded light.

## 2.4. Microcontroller System

### 2.4.1. Atmega328p Microcontroller

Microcontroller processes all sensor and Bluetooth data and updates the LED light accordingly. Since the processing time of the microcontroller can cause the congestion of the Bluetooth signals coming from the Android device, the efficiency of the software's algorithm was very important. The efficiency was achieved by minimizing the number of LED pixels and variables that need to be changed. Furthermore, another important requirement of the microcontroller is to calculate correct RGB values for various color temperature and brightness.

### 2.4.2. HC-05 Bluetooth Module

Choosing the application protocol for the Bluetooth communication was very important because the latency within the microcontroller may cause congestion of the messages. To reduce the possibility of the congestion, we have reduced the bandwidth of the message as much as possible, while setting sufficient time interval between the previous and current message to be sent.

## 3. Design Details

### 3.1. Light System

#### 3.1.1. LED Case Module

In order to meet the specific dimension and weight requirements of the physical lighting case, we used Autodesk Inventor. There were a couple requirements to regard when designing the physical case. The house for the LED had to be at least 4ft long and 4 inches wide. Also, there needed to be approximately 2 inches of space between the acrylic glass diffuser and the LED strip for reasonable diffusion of the LED lights. Also, Rick wanted there to be a handle at the end of the case such that it could be clamped onto his photography equipment.

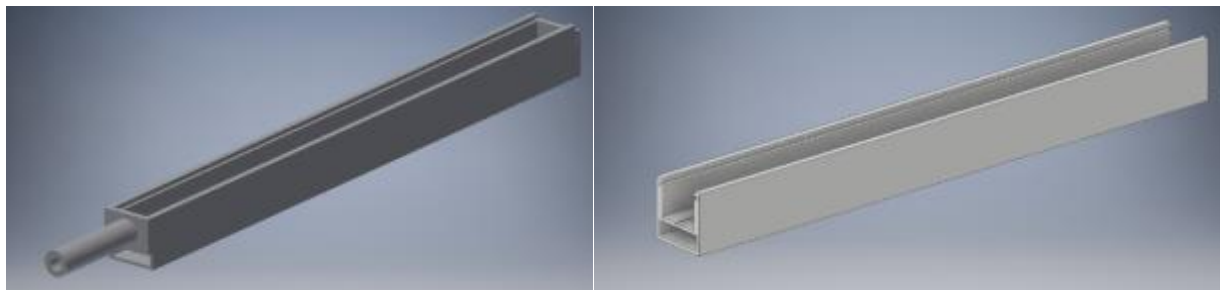


Figure 3.1.1.1. On the left is the assembled physical design of the Light Case, and on the right is the physical design of the body part of the Light Case.

The figure 3.1.1.2 below represents the cross section of the body part of the light case. The minor dent in the center of the design is where the LED strip will be housed. There are two 0.25 inch dents at the top of the figure where the acrylic glass can slide in. The rectangular hole in the bottom of the cross section is where our PCBs will be mounted.

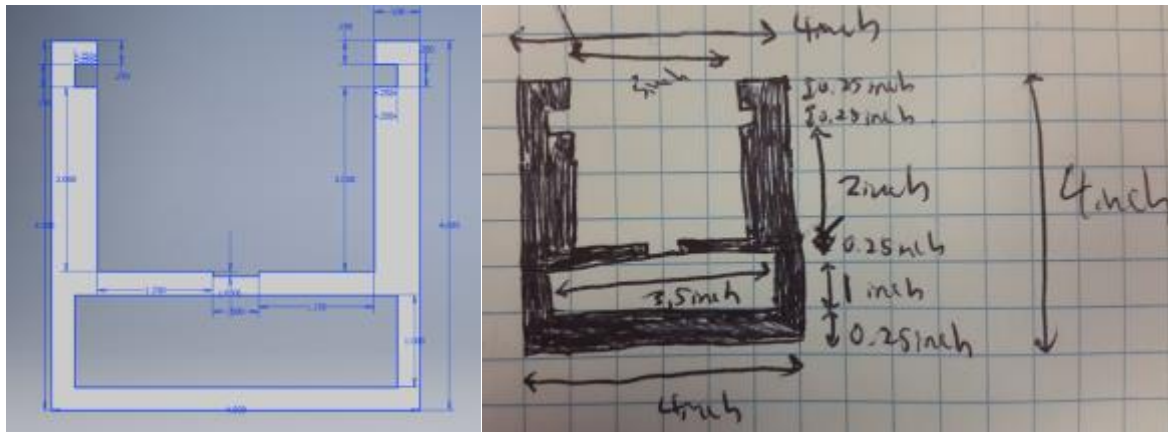


Figure 3.1.1.2. Cross section of body part of Light Case

For the final lighting case, Glen Hedin from the machine shop helped add an extra handle and a strap in order to make it easier to carry the LED system.

## 3.2. Power System

### 3.2.1. NiCd Battery Charger Module

The calculations below are based on the guidelines presented in the MAX713 NiCd Battery charger datasheet [1]. We need to charge 5 1.2V NiCd battery cells and therefore the minimum input voltage is

$$V_i = \# \text{ of cells} * 1.9 + 1.5V = 5 * 1.9 + 1.5V = 11V.$$

R1 in the schematic powers the chip and it is

$$R1 = \frac{V_{i_{min}} - 5V}{0.005A} = \frac{11V - 5V}{0.005A} = 1k\Omega.$$

In order to decide on the fast charging current, we need to take into consideration the battery capacity and the desired charge time. Our battery capacity is 2500mAh and we want to charge it in 2.5 hours, at a fast charge rate of 0.4C.

$$I_{fast} = \frac{\text{battery capacity}}{\text{charge time}} = \frac{2500mAh}{2.5hours} = 1A.$$

$R_{sense}$  sets the fast charge current into the battery.

$$R_{sense} = \frac{0.25V}{I_{fast}} = \frac{0.25V}{1A} = 0.25\Omega.$$

Since there are 5 cells, PGM1 should be connected to V+ and PGM0 should be open. Also, our charge time is 2.5 hours, and according to the datasheet, the voltage-slope cut off is enabled, then disabled when the voltage increase halts [1]. Next, the PNP BJT power dissipation is

$$PD_{pnp} = (V_{i,max} - V_{minimum_{battery}}) * I_{fast} = (11V - 5V) * 1A = 6W.$$

According to the 2N6109 PNP transistor datasheet, the maximum power dissipation of the transistor is 40W [1]. Since the maximum power dissipation is well above the power dissipation of our PNP transistor, the PNP transistor will operate well.

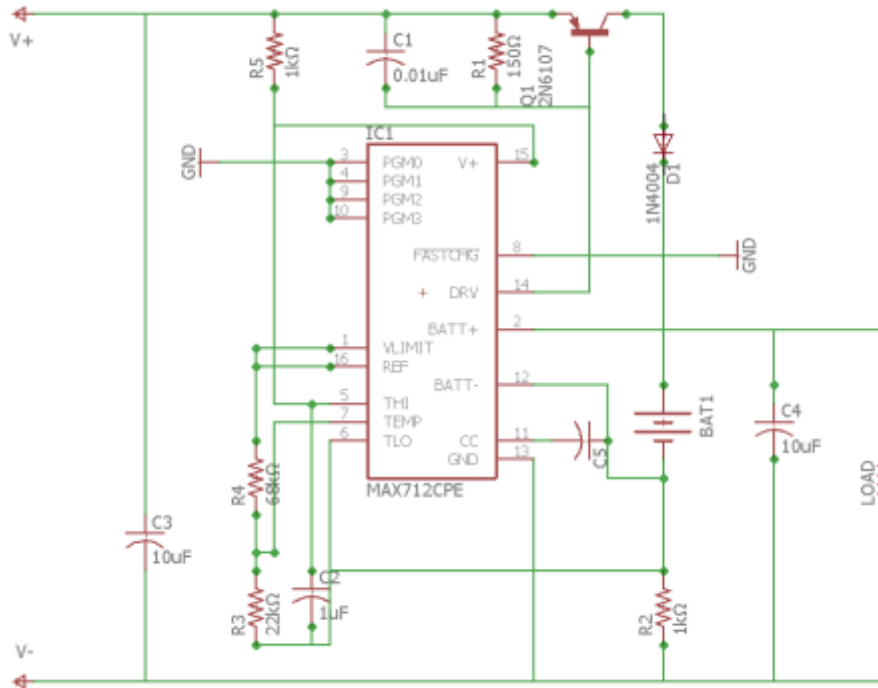


Figure 3.2.1.1. Schematics of NiCd Battery Charging Circuit

### 3.2.2. DC/DC Buck Converter Module

A major change in the final product from the design review is that we eliminated the linear voltage regulator module. Firstly, the microcontroller and the Bluetooth chip which we were planning to power with the 3.3V output from the linear voltage regulator, can be powered safely with the 5V buck converter output. The biggest reason for the change is because the data pin of the Digital Led Strip takes 5V as a high, and 0V as a low. If we power the atmega328P chip with 3.3V, the digital output pin of the microcontroller will only be 3.3V. By powering the microcontroller with a 5V buck converter output, we can send 5V digital signals to the data pin of the Digital Led Strip.



According to [2], the ripple current for the circuit shown in figure 2.3.3.4 can range from  $\frac{1}{4}$  to  $\frac{1}{2}$  and is determined by the inductance L. Also, increase in ripple current decreases the SNR (signal-to-noise ratio) but increases the energy loss. The inductance value is related to the output voltage by

$$L = \frac{1}{I_{IND} \cdot f_{SW}} \cdot \frac{(V_{IN} - V_{OUT}) V_{OUT}}{V_{IN}}$$

In order to achieve both adequate SNR and adequate energy loss, the ripple current is chosen to be  $\frac{1}{3}$ . Since

$I_{IND} = \frac{I_{OUT}}{3}$ , the value of the inductance becomes

$$L = \frac{3}{I_{OUT} \cdot f_{SW}} \cdot \frac{(V_{IN} - V_{OUT}) V_{OUT}}{V_{IN}}$$

According to the data sheet [2],  $f_{SW} = 500\text{kHz}$ . Also, the desired output voltage needed for other components in the overall light system is 5V, while the input voltage is 6V, specified by the battery voltage. Also, since the output current needed is  $I_{OUT} = 11\text{A}$ , the inductance can be calculated

$$L = \frac{3}{11\text{A} \cdot 500\text{kHz}} \cdot \frac{(6\text{V} - 5\text{V}) \cdot 5\text{V}}{6\text{V}} = 0.000455\text{mH}$$

Then, we can calculate the ESR using

$$ESR = \frac{V_{OUT} \cdot 10\text{mV} \cdot (1 - D)}{0.6\text{V} \cdot I_{IND}} = \frac{10\text{mV} \cdot L \cdot f_{SW}}{0.6\text{V}} = \frac{L \cdot f_{SW}}{60}$$

where D is the duty factor.

Since  $L = 0.000455\text{mH}$ , and  $f_{SW} = 500\text{kHz}$ , we obtain

$$ESR = \frac{0.000455\text{mH} \cdot 500\text{kHz}}{60} = 0.00379\Omega$$

According to [2], the recommended value for  $R_2$  is between  $1\text{k}\Omega$  and  $20\text{k}\Omega$ . So,  $10\text{k}\Omega$  was chosen as the value for  $R_2$ . Furthermore,  $R_1$  can be calculated using the equation

$$R_1 = \frac{V_{OUT} - \frac{I_{IND} \cdot ESR}{2} - 0.6}{0.6} \cdot R_2$$

Since  $V_{OUT} = 5V$ ,  $I_{IND} = \frac{I_{OUT}}{3} = \frac{11A}{3} = 3.667A$ , and  $ESR = 0.00379\Omega$ , the  $R_1$  is calculated as

$$R_1 = \frac{5V - \frac{3.667A \cdot 0.00379\Omega}{2} - 0.6}{0.6} \cdot 10k\Omega = 73.218k\Omega$$

Lastly, to calculate component values for the circuit shown in figure 2.3.3.4 we need to calculate the value of  $R_{TRIP}$  to specify the overcurrent value at which the controller restarts after a hiccup delay. The resistance  $R_{TRIP}$  can be calculated using

$$R_{TRIP} = \frac{\left( I_{OCP} - \left( \frac{1}{2Lf_{SW}} \right) \cdot \frac{(V_{IN} - V_{OUT})V_{OUT}}{V_{IN}} \right) \cdot 32 \cdot R_{DS}}{I_{TRIP}}$$

In order to prevent the current from damaging the components of the circuit, the overcurrent value,  $I_{OCP}$  was chosen to be 20A. According to [2], for  $I_{OCP} = 20A$ ,  $I_{TRIP} = 10\mu A$ , and  $R_{DS} = 1.6m\Omega$ . Using these values,  $R_{TRIP}$  can be calculated as

$$R_{TRIP} = \frac{\left( 20A - \frac{1}{2 \cdot 0.000455mH \cdot 500kHz} \cdot \frac{(6V - 5V) \cdot 5V}{6V} \right) \cdot 32 \cdot 1.6m\Omega}{10\mu A} = 93k\Omega$$

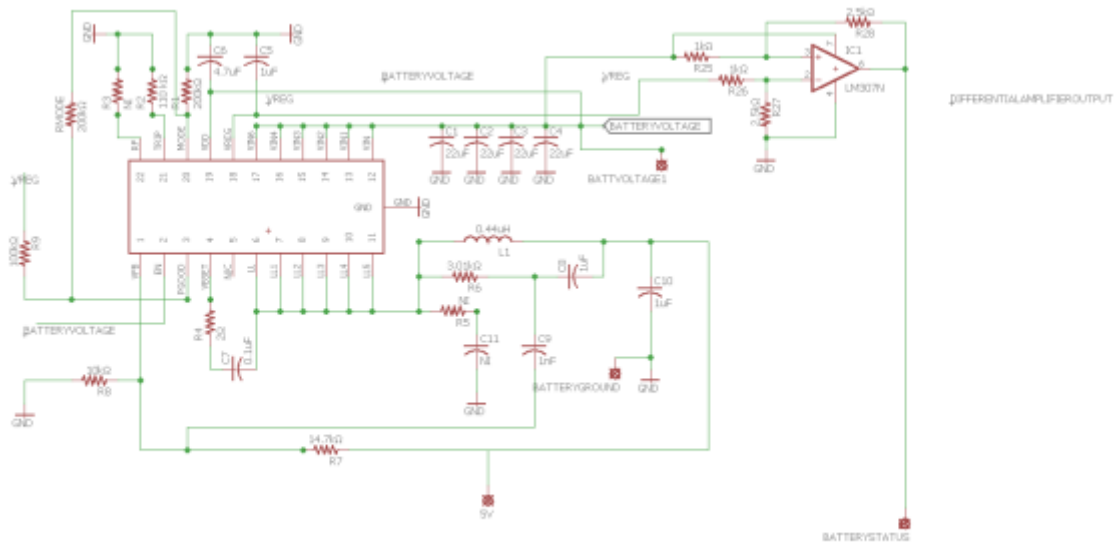


Figure 3.2.2.1 Schematics of NiCd Buck Converter and Differential Amplifier

### 3.3. User Interface

#### 3.3.1. Android Smartphone and Application

The application uses the `BluetoothAdapter` class available in Android API to interface with the Bluetooth hardware on the Android device. When the Bluetooth setting is turned on, the application invokes the `startDiscovery()` method to start scanning near Bluetooth devices, and it schedules `cancelDiscovery()` method to be invoked in 20 seconds, to allow the scan to have sufficient time to discover multiple devices, but also to stop wasting resources on scanning once the devices are found. When a Bluetooth device is selected, it creates another thread, where it creates a new `BluetoothSocket` object, and opens the input and output streams for communication with the Light Brush Pro device.

Figures 3.3.1.1-3.3.1.2 show the application screens displayed on an Android tablet. As shown in the figures, `SeekBar` views are mainly used to control the color temperature and the brightness. The `onProgressChanged` method of the view objects were overwritten, so that it reads the current cursor position and sends the progress value along with appropriate command to via the Bluetooth output stream. The design decisions on the Bluetooth message structure and rules are described in section 2.2.2 in more detail.

The gradient cursors are controlled by using the `GradientControlBar` shown in the figures. It is a custom view designed to display the RGB values currently displayed on the LED light and to allow intuitive control of the cursors. This view allows the users to move around the gradient cursor by simply dragging the triangle cursors shown in the view. As the cursor location changes, it sends Bluetooth message to the microcontroller to update the LED state using the rule described in section 2.2.2, and it also updates the light displayed on the gradient control bar.

Another feature that the Android application supports is saving and loading previous LED settings. When user enters the name of the setting and clicks save button, the application will save current state of the LED in SQLite database. And, once the user click on of the existing settings, it will load the corresponding LED state and update the current state accordingly.

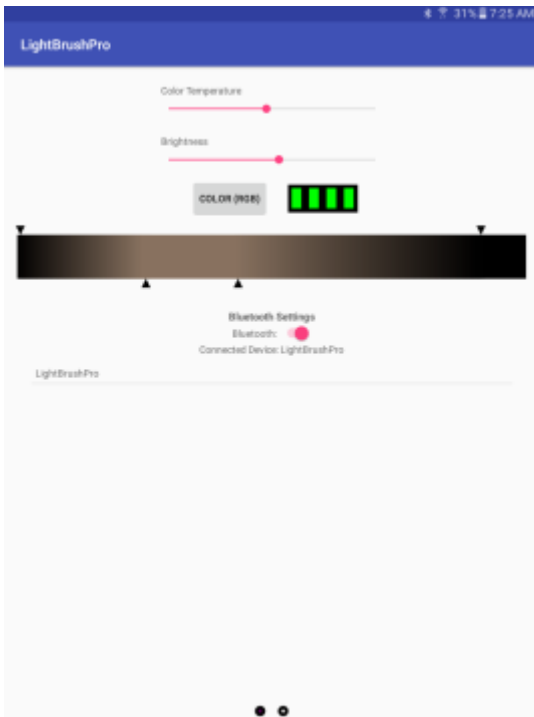


FIGURE 3.3.1.1. ANDROID APP WITH BLUETOOTH SETTINGS

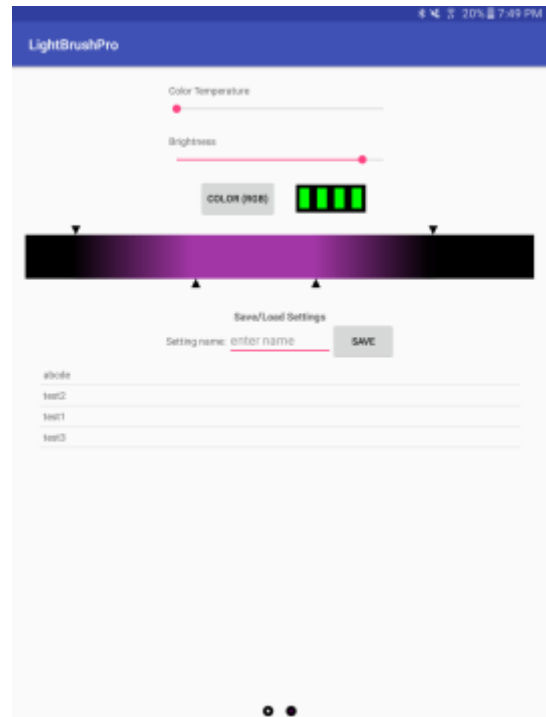


FIGURE 3.3.1.2. ANDROID APP WITH SAVE/LOAD SETTINGS

### 3.3.2. Rotary Encoders and Power Switch

Figure 3.3.2.1 shows how rotary encoders are connected to the microcontroller. Rotary encoder and power switch circuits were designed to increase the probability of microcontroller detecting each change in sensor value correctly. Because these are physical switches, the contact noise occurred every time the sensor values were updated. This contact noise provided possibility of detection false positive for the microcontroller. In order to prevent the false positive, we smoothed the sensor's output signal by placing an RC circuit as in this example project [3].

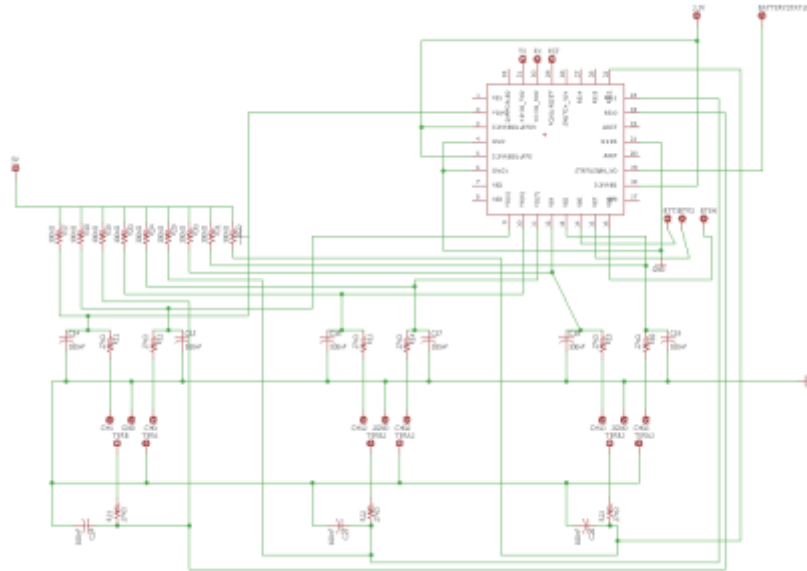


Figure 3.3.2.1 Schematics of Rotary Encoder Connected to Microcontroller

### 3.3.3. Battery Status Bar

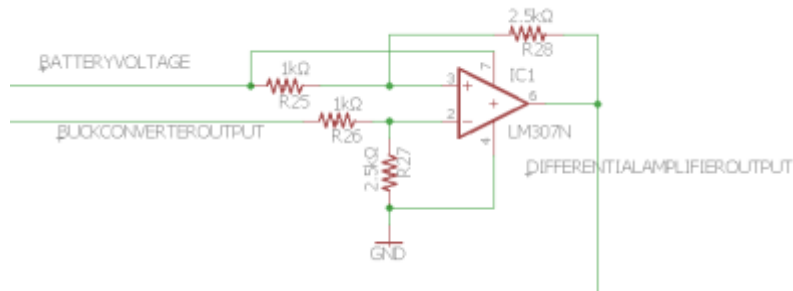


Figure 3.3.3.1: Differential Amplifier Circuit

$$\text{if } \frac{R_{28}}{R_{25}} = \frac{R_{22}}{R_{26}}, V_{OUT} = \frac{R_{28}}{R_{25}} * (V_2 - V_1)$$

V2 refers to the battery voltage, and V1 refers to the buck converter output in the figure above. The maximum battery voltage is 7V and the buck converter output is 5V. In order for the differential amplifier to output 5V,

$$5V = \frac{R_{28}}{R_{25}} * (7V - 5V)$$

$$\frac{R_{28}}{R_{25}} = 2.5$$

I set R28 to 2500 ohms and R25 to 1000 ohms. I set R22 and R26 accordingly as shown in the figure above. According to this configuration, when the battery is at its minimum voltage output 5V, the differential amplifier will output 0V. The output of the differential amplifier will always range between 0V and 5V [4].

It is important to mention that we used a voltage divider instead of a differential amplifier. Our assumption for using the differential amplifier was that the battery voltage would vary while the buck converter output would be very constant. However, due to imperfections in the buck converter feedback loop design, the output of the buck converter varied slightly as the input changed. Our system still worked fine. In order to minimize power loss due to current flow through the voltage divider, we used two 1 mega ohm resistors in series.

Using the derived information, we specified our software to display different number of battery status bar pixels for different voltage range. The mapping between the two are found in table 3.3.3.1.






Battery Status Bar	Voltage Range (V)
	> 3.5
	3.0 – 3.5
	2.95 – 3.0
	2.925 – 2.95
	< 2.925

Table 3.3.3.1. Battery Status Dedicated for Voltage Range

## 3.4. Microcontroller Systems

### 3.4.1. Atmega328p Microcontroller

As shown in figure 3.4.1.1, the microcontroller iteratively checks for any change in sensor value and power switch and for any available data in the Bluetooth serial input. Once the data is available, it processes the data by updating variables and sending data to the LED light. In order to reduce the processing time, the software was designed to store the previous value of any variable, and compares it to the new variable to update only parts of the LED's that are necessary to change, saving the time to update unchanged pixels.

To calculate the brightness given some RGB values, we divided the brightness option to 100 possible options, where 0 indicates LED off and 99 indicates the brightest option, and used these values divided by 99 as the multiplier to RGB value. This allowed very constant change in brightness of the LED lights as the brightness changed. For each color temperature, there is a specific RGB values that display the color temperature. The algorithm used to find the RGB values given the color temperature was found in [5]. This algorithm approximates the RGB curve for color temperatures for the interval 1,000K – 40,000K, to use only simple math functions such as natural logarithm and exponents to calculate the RGB value for given temperature.

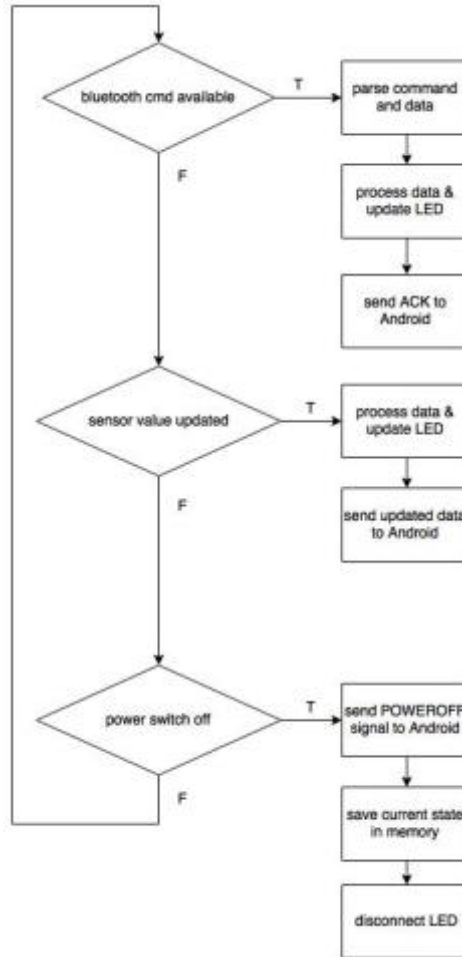


FIGURE 2.4.1.1. FLOWCHART OF MICROCONTROLLER SOFTWARE

### 3.4.2. HC-05 Bluetooth Module

For communication between the application and the microcontroller via Bluetooth, both systems send messages in a specific structure. The message is 2 bytes each, where the first byte contains the command ASCII character, while the second byte contains the necessary data for the command. The list of commands used in the design are shown in table 2.2.2.1. This specific structure is chosen in order to reduce the message processing time for the microcontroller as much as possible, since larger messages require the microcontroller to decode the message.

Command	Description
Q	Update brightness
T	Update color temperature
Z	Update gradientA cursor
X	Update solidA cursor
C	Update solidB cursor
V	Update gradientB cursor
R	Update color red
G	Update color green
B	Update color blue
S	Update battery status

Table 3.4.2.1. List of commands used in design

## 4. Verifications

### 4.1. Power System

#### 4.1.1. NiCd Battery Charger Module

The battery charger module was one of the easiest modules to verify. We fully discharged a 6V NiCd battery and then used a multimeter to make sure that the voltage across the battery was near to 5V. We connected the BATT+ pin of the battery charging circuit to the positive terminal of an ammeter. Then we connected the ground terminal of the ammeter to the positive end the battery. Lastly, we connected the negative end of the battery to the BATT- pin of the battery charging circuit shown in figure 3.2.1.1. We powered the battery charging circuit with a 12V source. The ammeter read 1A. After 2.5 hours, we checked the voltage across the 2.5Ah battery and it was 6.94 volts. It was fully charged.

#### 4.1.2. NiCd Battery and Battery Status Bar

In order to prove that our battery 6V battery stored 2.5Ah of charge to keep the system continuously powered, we discharged to 6V battery at 5A for 30 minutes while using a multimeter to track the voltage across the voltage every 3 minutes. We tested the battery status output module simultaneously.

While drawing 5A from the battery for 30 minutes, we also measured the voltage divider voltage every 3 minutes. We were successful in decreasing the battery voltage output to be between 0V and 5V, values the microcontroller analog input could handle. The table and graph below portray the battery voltage/battery status voltage over time. The results were comparable to the graph from the battery datasheet [6]. Since the battery is 2.5Ah, a 2C discharge corresponds to a 5A discharge.

Time (minutes)	Battery Voltage Output (V)	Battery Status Output (V)
0	7	3.5
3	6.2	3.1
6	6.0	3.0
9	5.95	2.98
12	5.93	2.96
15	5.90	2.95
18	5.92	2.96
21	5.91	2.95
24	5.85	2.925
27	5.5	2.75
30	5	2.5

*Table 4.1.2.1. Battery Voltage Output and Battery Status Output measured every 3 minutes*



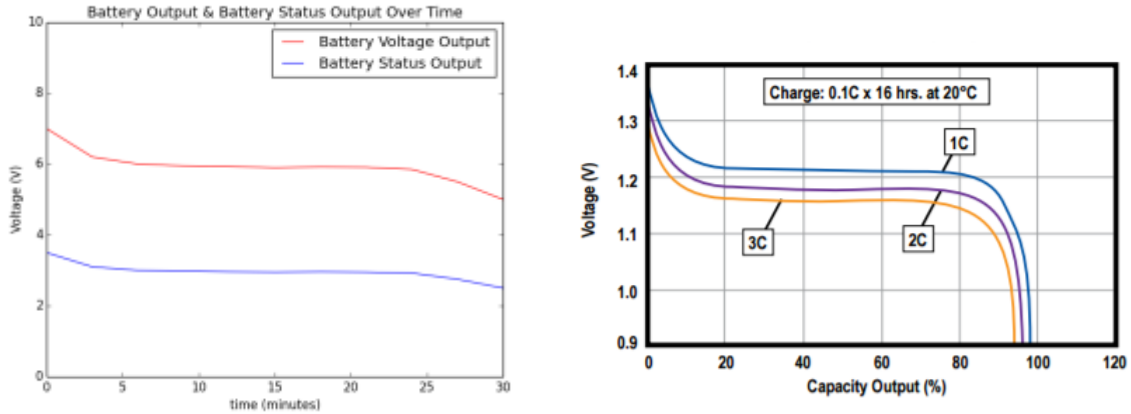


Figure 4.1.2.2. The graph on the left shows the battery output and the battery status output over time when drawing 5A. The graph on the right is from the NiCd battery datasheet for comparison [6].

#### 4.1.3. DC/DC Buck Converter Module

The requirement of the buck converter module was that it could provide 5V (+/-) from a 6V NiCd battery at a peak current draw of 5A. To test our system, we connected the input of the buck converter circuit to a DC power supply, and the output of the circuit to the digital LED strip. We also connected a multimeter in series between the buck converter circuit output and the LED strip input to make sure the current draw was 5A. After connecting oscilloscope probes across the output of the buck converter circuit and ground, we turned on the DC power supply to 7V. We made sure that buck converter output was lower than 5.5V and then decreased the DC power supply output voltage. While decreasing the voltage, we had in mind the high level requirement that the battery should be able to power the LED strip for at least 15 minutes. According to the table 4.1.2.1, the battery voltage would be 5.9V after 15 minutes of 5A discharge. Therefore, we decreased the voltage until 5.9V and saw that the buck converter output voltage was still quite above 4.5V. Therefore, we decreased the DC power supply voltage until 5.8V where the buck converter output 4.5V. The buck converter could output 5V (+/- 10%) for a battery voltage output between 5.8V and 7V. According to table 4.1.2.1, a fully charged battery would power the system at a peak current draw of 5A, for 24 minutes. An ideal buck converter circuit should always have a constant voltage output. However, due to a minor error in the buck converter circuit's feedback loop, the buck converter output shifted slightly as described above and in the figure below. However, the requirements were met and there was no problem powering our overall system.

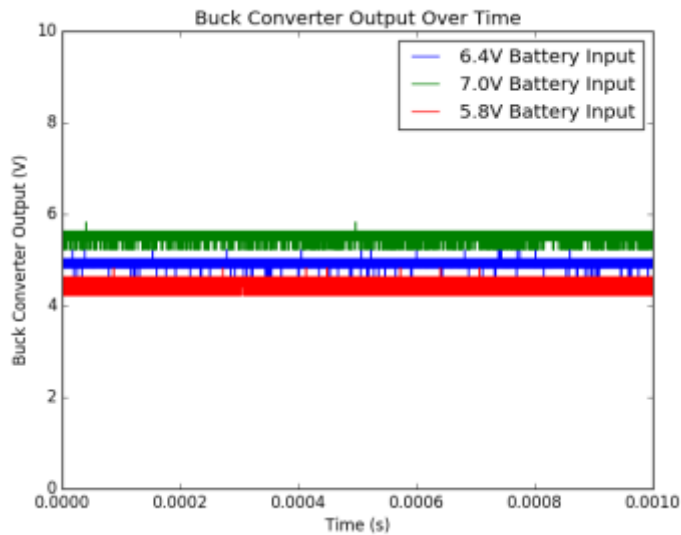


Figure 4.1.3.1. Buck Converter Output Over Time depending on Input

## 4.2. User Interface

### 4.2.1. Android Application

The requirement for the Android application was to display the identical color patterns shown in the LED in its gradient control bar. Table 4.2.1.1 shows the comparison between the two displays. For the first 3 items, the color displayed are very similar, while the LED's usually show brighter light with same color and pattern. However, for the last case, the gradient patterns match, but the color on the LED is slightly more bluish relative to the gradient control bar.

App		
LED		
App		
LED		

Table 4.2.1.1. Comparison Between the LED and Gradient Control Bar

### 4.2.2. Rotary Encoders and Power Switch

With microcontroller, we were able to detect change in the rotary encoder for every 30 degree rotation. Using the verification software, we verified that microcontroller detected changes 10-12 times for 360 degree rotation. Also, using the multimeter we verified that the power switch is 5V when pushed and 0V if released.

## 4.3. Microcontroller System

### 4.3.1. Atmega328p Microcontroller and HC-05 Bluetooth Module

The requirements for the microcontroller and HC-05 module are set to allow smooth transition between the LED state change and reliable remote control. Using the verification software, we have tested that the HC-05 module

can receive messages from an Android device from 50ft away. Also, we were able to verify that the transition between LED state change was always smooth when the message interval is set to 70ms.

## 5. Costs

Item	Part Number	Description	Unit Cost	Quantity	Total Cost
Digital LED Strip	Adafruit NeoPixel Digital RGB LED Strip 144 LED	1m digital LED strip consisting of 144 WS2812 based LEDs	\$22.99	2	\$45.98
Rotary Encoder	COM - 09117	Rotary encoder for controlling gradients and brightness	\$2.95	5	\$14.75
Black Metal Knob	COM - 10002	Metal Knob for rotary encoders	\$1.5	5	\$7.5
Microcontroller	ATmega328P - AU	32 pin microcontroller	\$2.01	1	\$2.01
NiCd Battery	PS-CX	Power source for design	\$8.5	6	\$51
Bluetooth Transceiver	HC-05 RS232	Bluetooth transceiver for remote control of LED system	\$7.88	1	\$7.88
Dc-Dc Buck Converter	TPS53353	Buck converter for steady voltage stepdown	\$7.74	1	\$7.74
PNP Transistor	2N6107	PNP Bipolar Power Transistor	\$0.97	1	\$1.61
NiCd Battery Fast Charge Controller	MAX713	NiCd Battery charging IC	\$9.12	1	\$9.12
Total Price			\$ 147.59		

Table 3.1.1. Cost Analysis of the Components Needed

Name	Hourly Rate	Hours Invested	Total Cost
------	-------------	----------------	------------

Kyoungyeon Cho	\$25	200	\$12,500.00
Jihong Min	\$25	200	\$12,500.00
Seungjun Cho	\$25	200	\$12,500.00
<b>Total Cost</b>	-	-	\$37,500.00

Table 3.2.2. Wages for Team Members

## 6. Conclusion

Although there were many unexpected obstacles on the way, we pushed through to mend and maneuver our project towards meeting our requirements. We could use a battery to power the entire system stably, while using the mobile application and rotary encoders to control the brightness and color of each pixel in the 4ft long LED strip in precise detail. Although the product was fully functional, the process in making was bumpy. We used 3 PCBs which were messy and had several connection errors. It was burdensome to debug and reroute the PCB connections using jumper wires. Also, even after verifying that the entire system functioned perfectly when stationed on a capacious bench, there were many milestones left before fitting all the PCBs into the tiny PCB slot of the Lighting Case. Also, less than an hour before the demo, we accidentally powered the microcontroller and LED strip with a 12V source when they could only handle 5V. The LED strip and microcontroller got fried, and therefore we had to reconstruct the PCB and order a new LED strip before the presentation.

There are a few uncertainties that we can work on to improve the stability and functionality of our product. Firstly, as mentioned above, our buck converter output varied slightly upon the input. If we improve the feedback loop of the circuit, the buck converter output would be steadier. Also, there was a slight mismatch between the LED RGB color expression and the desired color. We can work on this by performing calibration between the two displays. Our project didn't have space for battery casing because the battery was bigger than we expected. Therefore, we can redesign the case to accommodate a larger hollow handle for the battery to fit into. Also, the installation of our electrical system inside the case was messy with lots of wires. Our PCB design was separated into three parts, and it caused us to use a lot of wires to connect PCBs and other modules. We can create a more compact single LED in order to eliminate the messy wire connections.

Throughout the project, we followed the IEEE code of Ethics [7]. Our project confronted several technical issues. To acknowledge and debug errors, we asked for technical assistance from available resources and course staffs. These are consistent with IEEE code of Ethics 7: *"to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others"*. Our project uses 6V Ni-Cd Battery as a supply for the system. Special attention and care is required for the use of Ni-Cd Battery. Considering the potential rust and electric hazard of the battery, we designed the case of our project to have private space only to store battery inside the case. Unfortunately, the size of battery was bit bigger than expected. For future work, we will improve the design of the lightning case to place the battery in proper place. Our effort meets with the IEEE code of ethics 9: *"to avoid injuring others, their property, reputation, or employment by false or malicious action"*. Lastly, all our team members collaborated and contributed to the project, helping each other to understand the technical content of the project and improve the functionality. This is consistent with the IEEE code of Ethics 10: *"to assist colleagues and co-workers in their professional development and to support them in following this code of ethic"*.

## 7. Appendix

### 7.1. Requirements and Verifications

Component	Requirement	Verification
Battery Charger	<p>Requirement 1: The temperature of the circuit will be between 0°C and 40°C while charging the batteries at a charge rate of 0.4C.</p> <p>Requirement 2: The charger must fully charge the battery within 2.5 hours.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"><li>1) Fully drain the battery by connecting the battery to a DC electronic load and set it to constant current mode at 10A for 10 minutes.</li><li>2) Verify that the battery is fully discharged by probing it with a voltmeter. The battery should measure 5V when fully discharged.</li><li>3) Connect the battery to the charging circuit.</li><li>4) Using thermometer, measure the temperature at the surface of the charger IC until the battery is fully charged, to check that the temperature is always less than 40°C.</li></ol> <p>Verification 2:</p> <ol style="list-style-type: none"><li>1) Repeat steps 1) 3to ) in verification 1 for battery charger module.</li><li>2) After 2.5 hours, measure the voltage over the battery using the voltmeter to check that the value is between 6.8V and 7V, indicating the battery has been fully charged.</li></ol>

<p>NiCd Battery</p>	<p>Requirement 1: The 6V battery must store at least 2.5Ah of charge to keep the system continuously powered.</p> <p>Requirement 2: The battery pack must be mobile enough to be mounted on the led case. It must weigh less than 1kg.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Fully charge the battery using any NiCd AA battery charging device.</li> <li>2) Connect the battery to a DC electronic load and discharge at a constant 5A for 30 minutes.</li> <li>3) For 30 minutes, measure the voltage over the battery using a bench voltmeter to check that the voltage is always greater than 5V.</li> </ol> <p>Verification 2:</p> <ol style="list-style-type: none"> <li>1) Place the battery pack on a calibrated laboratory digital bench scale. Make sure nothing is connected to the battery pack.</li> <li>2) Measure the weight of the battery pack using scale to check that the weight is less than 1kg.</li> </ol>
---------------------	--	---

<p>DC/DC Buck Converter</p>	<p>Requirement 1: The buck converter must provide 5V (+/-10%) at a peak current draw of 5A from a 6V NiCd battery.</p> <p>Requirement 2: The buck converter must maintain a thermal stability of below 85°C.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Set the bench power supply output to 7 Volts with output off. Then connect the power supply output to the buck converter circuit input.</li> <li>2) Connect the output of the buck converter circuit to the input of a DC electronic load with a constant current draw of 5 Amps.</li> <li>3) Use a bench multimeter to make sure the output voltage of the buck converter circuit is always between 4.5V and 5.5V.</li> <li>4) Repeat steps 1~4 above with a bench power supply output of 5.8V.</li> </ol> <p>Verification 2:</p> <ol style="list-style-type: none"> <li>1) During the Verification 1 process above, use a thermometer to measure the surface temperature of the buck converter IC.</li> <li>2) Make sure the that temperature is below 85°C at all times.</li> </ol>
-----------------------------	--	---

<p>Atmega328p Microcontroller &amp; Software</p>	<p>Requirement 1: The delay between the change in user input and microcontroller starting to write data to the LED strip must be less than 400ms for all user input including the android application.</p> <p>Requirement 2: When changing the LED's state, the time it takes for the overall LED strip to fully change its state should be less than 100ms.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Connect the microcontroller to the Atmega328p programmer device, and connect RXD and TXD pins of the HC-05 device to pins PD[0] and PD[1] of the microcontroller.</li> <li>2) Open the Arduino IDE software on a computer, and connect the programmer device to the computer using a USB cable.</li> <li>3) Load the delay test program to the microcontroller using Arduino IDE.</li> <li>4) Load the delay test application to an Android smartphone using Android Studio.</li> <li>5) Using Settings app, turn on the Bluetooth on the Android device and establish a connection with the HC-05 device.</li> <li>6) Open the delay test application app on the Android device.</li> <li>7) Once the delay test application is loaded, touch the button to send a bluetooth packet to the microcontroller.</li> <li>8) Open the serial monitor on the Arduino IDE. Then, check on the output panel to check that the recorded delay is less than 400ms.</li> </ol> <p>Verification 2:</p> <ol style="list-style-type: none"> <li>1) Repeat steps 1-7 from verification 1 of for the microcontroller module, but load LED test program and application instead of the delay test program and application.</li> <li>2) Open the serial monitor on the Arduino IDE. Check the output panel of the serial monitor to check that the recorded delay is less than 100ms.</li> </ol>
--	--	--



<p>HC-05 Bluetooth Transceiver</p>	<p>Requirement 1: The communication with the Android smartphone should not fail within the distance of 15ft.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Repeat steps 1-6 from verification 1 of for the microcontroller module, but load distance test program and application instead of the delay test program and application.</li> <li>2) Connect a LED to pin PD[5] of the microcontroller, and connect a 10kΩ resistor between the ground and the LED.</li> <li>3) Place smartphone within 50cm distance from the microcontroller.</li> <li>4) Once the distance test application is loaded, touch the button to start sending bluetooth packets to the microcontroller periodically.</li> <li>5) Slowly move the smartphone away from the microcontroller at 10 cm per second, until the LED light of connected to the microcontroller turns on.</li> <li>6) Measure the distance from the smartphone to the microcontroller to check that is is greater than 15ft.</li> </ol>
------------------------------------	--	--

<p>Rotary Encoder</p>	<p>Requirement 1: Turning 30 degrees (+/- 10%) in the gradient location encoder should move the starting or endpoint of the gradient by one pixel.</p> <p>Requirement 2: To allow effortless control of rotary encoder, the minimum force necessary to turn the rotary encoder needs to be low, and less than 5N.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Connect the microcontroller to the Atmega328p programmer device, and connect the each port of rotary encoder to pins PB[2:0] of the microcontroller.</li> <li>2) Open the Arduino IDE software on a computer, and connect the programmer device to the computer using a USB cable.</li> <li>3) Load the encoder test program to the microcontroller using Arduino IDE.</li> <li>4) Spin the encoder clockwise by 360 degrees.</li> <li>5) Open the serial monitor on the Arduino IDE. Check the output screen of the Arduino IDE to check that the printed value, which is identical to the number of pixels changed, is between 11-13.</li> </ol> <p>Verification 2:</p> <ol style="list-style-type: none"> <li>1) Tape 3m string on the side of the rotary encoder cap, and wrap the string around the encoder cap 10 times.</li> <li>2) Connect the spring force gauge to the string.</li> <li>3) Slowly increase force of pulling the spring force gauge away from the encoder until the encoder starts to spin.</li> <li>4) When the encoder starts to spin, measure the value of the force gauge to check that the force is less than 5N.</li> </ol>
-----------------------	---	---

<p>Power Switch &amp; Buttons</p>	<p>Requirement 1: The power switch and buttons circuits should output LOW (0V) signal when pressed or switched on, and output digital HIGH (5V) signal otherwise.</p> <p>Requirement 2: The switch and buttons should be debounced, and once the push button is pressed, the changed value should be kept for at least 0.75ms (see calculations) so that the microcontroller is guaranteed to sense the changed state of the button.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Connect V_in pin of the power switch circuit to 5V constant voltage source, and set the switch to LOW mode.</li> <li>2) Using voltmeter, measure the output of the power switch circuit to check that the output voltage is 0V.</li> <li>3) Switch the power switch.</li> <li>4) Using the voltmeter, measure the output of the power switch circuit to check that the output voltage is 5V.</li> <li>5) Connect V_in pin of the button circuit to the 5V constant voltage source.</li> <li>6) Using voltmeter, measure the voltage output of the button circuit to check that the voltage output is 0V.</li> <li>7) While pressing the button, measure the voltage output using the voltmeter to check that the value is 5V.</li> </ol> <p>Verification 2:</p> <ol style="list-style-type: none"> <li>1) Connect V_in pin of the power switch circuit to 5V constant voltage source, and set the switch to LOW mode.</li> <li>2) Connect the output of the power switch circuit to the oscilloscope to measure the voltage output over time.</li> <li>3) Switch the power switch twice.</li> <li>4) By observing the oscilloscope screen, check to see that there is no spike indicating bouncing of the switch.</li> <li>5) Repeat 1-2, but connect the button circuit instead of the power switch circuit.</li> <li>6) Press the button, and release the button immediately.</li> <li>7) By observing the oscilloscope screen, check to see that there is no spike indicating bouncing of the switch and that the switch value is kept at 5V for at least 0.75ms.</li> </ol>
-----------------------------------	--	--

<p>Android Smartphone &amp; Application</p>	<p>Requirement 1: The android application must be able to correctly display the battery status and state of the light system, such as gradient pattern, color, and brightness.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"><li>1) Repeat steps 1-6 from verification 1 of for the microcontroller module, but load smartphone test program and application instead of the delay test program and application.</li><li>2) Connect the D_in port of the LED strip to PD[2] pin of the microcontroller.</li><li>3) Start the smartphone test application, and once the smartphone test application is loaded, touch the button on the application to signal the microcontroller to generate arbitrary LED pattern.</li><li>4) Compare the pattern displayed on the LED strip and the expected light pattern displayed on the smartphone are identical to humans.</li></ol>
---	--	---

<p>Battery Status Bar</p>	<p>Requirement 1: The circuit needs to be able to map the voltage output of the NiCd battery into a value in 0-5V range with a tolerance of <math>\pm 10\%</math>.</p> <p>Requirement 2: The number of LED's turned on must be proportional to the remaining charge in the battery. That is, if some percentage x of the battery is remaining, only the rounded integer value of x percent of the total number of LED's should be turned on to indicate how much battery is actually remaining.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Fully charge the battery using any AA NiCd battery charging device.</li> <li>2) Connect the battery status circuit to the battery.</li> <li>3) Using oscilloscope, measure the voltage output of the battery status circuit over time.</li> <li>4) Also, connect the battery to the DC electronic load to draw 5A current for 30 minutes.</li> <li>5) Observe the oscilloscope screen to check that the shape of the voltage curve is similar to the that of the discharge curve shown in the NiCd battery datasheet. Also, check that the battery status circuit output voltage is proportional to the output voltage of the battery.</li> </ol> <p>Verification 2:</p> <ol style="list-style-type: none"> <li>1) Fully charge the battery using any AA NiCd battery charging device.</li> <li>2) Connect the battery status circuit to the battery.</li> <li>3) Connect the microcontroller to the Atmega328p programmer device, and connect the output of the battery status circuit to the ADC6 pin of the microcontroller.</li> <li>4) Open the Arduino IDE software on a computer, and connect the programmer device to the computer using a USB cable.</li> <li>5) Load the battery status bar test program to the microcontroller using Arduino IDE.</li> <li>6) Connect the battery to the DC electronic load to draw 10A current for 15 minutes.</li> <li>7) Open the serial monitor on the Arduino IDE. Check the output screen of the serial monitor to see that the time it takes for one LED to turn off is approximately 3 +/- 10% minutes.</li> </ol>
---------------------------	---	--

<p>Digital LED Strip</p>	<p>Requirement 1: It must be possible to control the white light temperature from 3000k to 5600k.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"><li>1) Set the bench power supply to 5V constant DC with power off. Then, connect the power supply output to the voltage input pin of the digital LED strip.</li><li>2) Connect the ground pin of the digital LED strip to the ground on the power supply.</li><li>3) Power an arduino with a laptop and upload a LED-test program to the arduino. The test program should make the LED strip output white lights with temperatures ranging from 3000k to 5600k, according to an LED color chart.</li><li>4) Connect the analog output pin of the arduino to the data input pin of the LED strip.</li><li>5) Download a mobile application that measures color temperature with the built in camera to make sure the LED strip is outputting the colors desired colors.</li></ol>
--------------------------	---	--

<p>Light Casing</p>	<p>Requirement 1: The lights produced by the LEDs must be fully diffused when it reaches the surface of the housing such that it is impossible to spot a single LED.</p> <p>Requirement 2: The case should be 4ft long, and 4 inches wide, and lighter than 15kg.</p>	<p>Verification 1:</p> <ol style="list-style-type: none"> <li>1) Set the bench power supply to 5V constant DC with power off. Then, connect the power supply output to the voltage input pin, and the data input pin of the digital LED strip.</li> <li>2) Connect the ground pin of the digital LED strip to the ground on the power supply.</li> <li>3) Turn the voltage output of the power supply on and turn off all the lights in the room.</li> <li>4) Make sure that it isn't possible to differentiate between any individual pixels when all the LEDs are turned on.</li> </ol> <p>Verification 2:</p> <ol style="list-style-type: none"> <li>1) Use a measure tape to measure the dimensions of the LED case.</li> <li>2) Cautiously put the LED case on a calibrated laboratory digital bench scale.</li> <li>3) Set the scale to kilograms and make sure that the case is lighter than 15kg.</li> </ol>
---------------------	---	--

## 8. References

- [1] M. Integrated, "MAX712/MAX713 NiCd/NiMH Battery Fast-Charge Controllers," [Online]. Available: <http://pdfserv.maximintegrated.com/en/ds/1666.pdf>. [Accessed 24 02 2017].
- [2] T. Instruments, "TPS53353 High-Efficiency 20-A Synchronous Buck SWIFT™ Converter With Eco-mode™," 02 2016. [Online]. Available: <http://www.ti.com/lit/ds/slusak2c/slusak2c.pdf>. [Accessed 24 02 2017].
- [3] Editorial Team, "How to Use a Rotary Encoder in an MCU-Based Project," 1 November 2015. [Online]. Available: <https://www.allaboutcircuits.com/projects/how-to-use-a-rotary-encoder-in-a-mcu-based-project/>. [Accessed March 2017].
- [4] ElectronicsTutorials, "The Differential Amplifier," ElectronicsTutorials, [Online]. Available: [http://www.electronics-tutorials.ws/opamp/opamp\\_5.html](http://www.electronics-tutorials.ws/opamp/opamp_5.html). [Accessed 25 03 17].
- [5] T. Helland, "How to Convert Temperature (K) to RGB: Algorithm and Sample Code," [Online]. Available: <http://www.tannerhelland.com/4435/convert-temperature-rgb-algorithm-code/>. [Accessed March 2017].
- [6] P. S. Corporation, "Nickel Cadmium and Nickel-Metal Hydride Batteries and Chargers," 2010. [Online]. Available: [http://www.power-sonic.com/images/power-sonic/technical/1277751010\\_20106027-NiCdCatalog-Lo.pdf](http://www.power-sonic.com/images/power-sonic/technical/1277751010_20106027-NiCdCatalog-Lo.pdf). [Accessed 24 02 2017].
- [7] Alpha, "SparkFun," [Online]. Available: <https://www.sparkfun.com/datasheets/Components/TW-700198.pdf>. [Accessed 24 02 2017].
- [8] T. Instruments, "TPS799-Q1 200 mA, Low Quiescent Current, Ultralow Noise,," 06 2015. [Online]. Available: <http://www.ti.com/lit/ds/symlink/tps799-q1.pdf>. [Accessed 24 02 2017].
- [9] Atmel, "AVR Instruction Set Manual," 2016. [Online]. Available: <http://www.atmel.com/images/Atmel-0856-AVR-Instruction-Set-Manual.pdf>. [Accessed 24 February 2017].
- [10] Atmel, "ATmega328P," Atmel, [Online]. Available: <http://www.microchip.com/wwwproducts/en/ATMEGA328P>. [Accessed 24 February 2017].
- [11] Worldsemi, "WS2812B Intelligent Control LED Integrated Light Source," [Online]. Available: <https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf>. [Accessed 24 02 2017].
- [12] D. Hildebrandt, "Light Painting Part One – the Photography," [Online]. Available: <https://digital-photography-school.com/light-painting-part-one-the-photography/>. [Accessed 8 February 2017].
- [13] IEEE, "IEEE Code of Ethics," IEEE, 2017. [Online]. Available: <http://www.ieee.org/about/corporate/governance/p7-8.html>. [Accessed 8 February 2017].
- [14] "Nickel Cadmium Batteries Handling Precautions," [Online]. Available: [http://www.taylor-edge.com/reference/Batteries/Ni-Cd\\_Precautions.pdf](http://www.taylor-edge.com/reference/Batteries/Ni-Cd_Precautions.pdf).