SCRIM LIGHT

By

Gary Liu

Mario Posso

Final Report for ECE 445, Senior Design, Spring 2017

TA: Jose Sanchez Vicarte

02 May 2017

Project No. 35

Abstract

The Scrim Light is a device that can provide a completely flexible and customizable lighting setup for users. By altering the brightness and color temperature through the onboard user interface, the user can create a profile for their specific application. The device provides several preset profiles, such as a gradient fading from one end to the other, or from the center outwards. Additionally, the user can program in their own customized modes for later usage. Finally, the device was implemented in a cordless, battery powered housing, allowing for the light to be conveniently used in the desired studio setting.

Contents

1. Introduction

Photography lighting is a complicated and many-faceted issue, often requiring thousands of dollars in equipment to achieve the desired lighting effect. Rick Kessinger, a local photographer in Bloomington, approached the 445 students with a problem; he was attempting to photograph automobiles in a particular fashion, and therefore needed a particular lighting setup in order to achieve that.
When photographing cars, light would typically need to be reflected off surfaces known as scrims in order to deliver a professional look and finish to the final product. Complicated setups often arise when the desired result looks to highlight the contours and shapes that make the vehicle unique, and small studios typically cannot afford the cost.

Rick proposed a prototype solution: The same effects can be achieved via long-exposure shots, but would require an appropriate lighting mechanism that can produce the specified gradient directly. His initial design consisted of a wall-powered LED strip which he could coat in a particular way to produce the gradient. This, however, was not an appropriate solution, as this would only produce a single non-adjustable gradient and made transportation around the studio very difficult.

Our proposed solution is a light that gives much more control to the user by allowing them to produce different gradients, such as fading from one end to the other, or from the center out and vice versa. The user has the ability to vary the brightness, temperature**,** and intensity of the light, adjust the starting position and profile of the gradient, and individually group LEDs together and adjust their brightness to produce a well-defined custom profile. The user is also able to store their current profile for non-volatile retention and later use as a preset mode. One of the more important parts of this solution is that the device is cordless and battery powered, as this allows for increased mobility when using the device in the studio. Our design is also modular, allowing the user to easily replace any malfunctioning LED unit and add more if the extra space is properly accommodated.

Overall, we were able to produce an intuitive and simple to use device that met the requirements that were originally set out. Our design for a custom bus protocol functioned as planned and enabled the device to operate with no visual latency. Some non-essential components of our design, however, failed to fully meet requirements.

1.1 Block Diagram

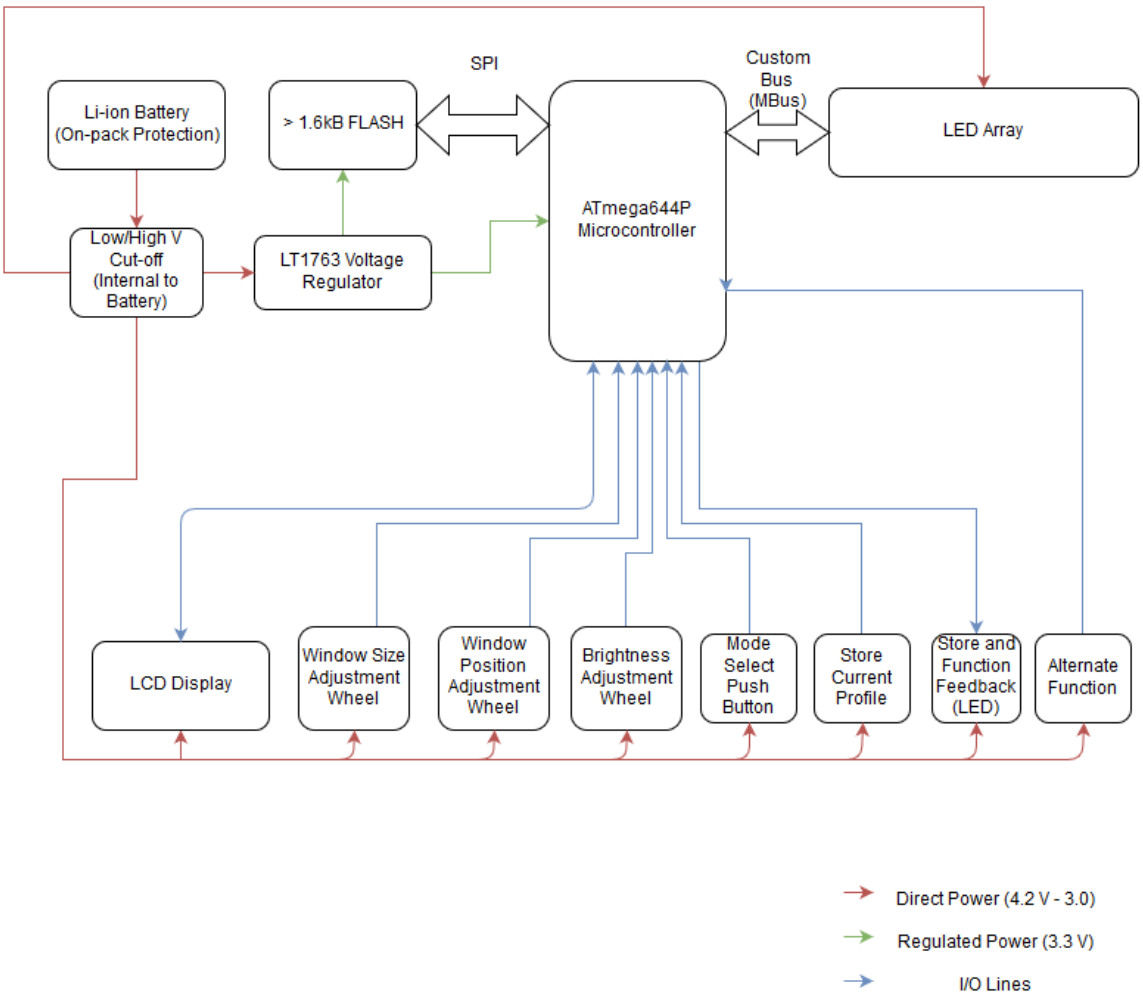

Figure 1.1: Overall design block diagram

1.1.1 Li-ion Battery with On-Pack Protection and Charger

Due to the high-current nature of the application (80 LEDs, each driven at a maximum forward current of 60 mA) and the portability requirements set forth by the end-user, Lithium-Ion cells were chosen as the ideal source of on-board power. The life-time of the device was set out to be at least 45 minutes, discharging at nominal 3.7 V at a discharge current not exceeding 5.5 A. We chose to purchase an industry-standard Li-Ion pack housing 3 cells with on-pack protection and an overall capacity of 7800 mAh.

We also chose an industry standard charger which is specifically designed for usage with our battery pack. The charger is able to charge the battery to full safely in 8 hours. It also ensures that the battery current draw drops dramatically before reaching any critically low voltage levels, which promotes higher cycle life for the Li-ion cells by maintaining a 50-65% depth of discharge. [2]

1.1.2 Voltage Regulator

The Flash memory (Serial NOR Flash) has a maximum input rating of 3.6 V, and the operating range of the device exceeds this requirement. To keep the cost down (not opting for a bigger memory or different type), the input range was met by a simple LT1763-3.3 regulator circuit. Additionally, we are using the regulator output to power the microcontroller so as to not over-supply the SPI communication into the flash, and for a more stable and decoupled operating voltage.

1.1.3 Flash Memory

A flash memory is the most readily available and low-cost non-volatile, rewriteable storage solution. A large part of the functionality of the device relies on the ability to retain and rewrite user profiles, and it is best to meet these with a low-cost, high data-retention flash memory. To properly store the profiles, the flash memory was chosen to have at least 1.6 kB, which stores 160 bytes per profile, and at least 10 profiles.

1.1.4 User Interface Elements

The LCD display communicates the current settings and mode selected to the user, and must be able to display at least 20 characters per line with 2 lines with backlighting. The LCD screen would be facing a different direction from the light, and its brightness is extremely low compared to that of the LED strip, so there will be no light contamination during operation. We understand, however, that some very light sensitive photography may warrant a stronger control on the screen, and as such we leave the time-out options to be set by the user.

The main handle also houses two buttons to cycle through the list of stored profiles, as well as a button to save the current profile.

In order to allow the user to intuitively control the brightness and color temperature of individual LEDs or set of LEDs, the device houses two rotary encoders (knobs) alongside the buttons described above. These knobs are capable of being pushed, like a button, to toggle their functionality and provide the user with more control. One knob controls adjustments of a selection window that dictates what LED or LED group is being targeted for adjustments in brightness or temperature, and the window size and window position can both be changed by this same knob. Similarly, the other knob controls the brightness and color temperature of the targets, and this enables the user to create a variety of unrestricted lighting patterns.

1.1.5 ATmega644P Microcontroller

This particular microcontroller was chosen because of the number of input and output pins required for our user interface elements as well as communication with the MBus protocol. Additionally, we needed the JTAG programming interface, as well as SPI communication with the flash memory.

The primary functionality of the microcontroller is to communicate with the LEDs, and send different addresses and values according to inputs from the user interface. It is also used to interface with the flash memory so that profiles can be loaded.

1.1.6 Custom Bus Protocol and LED Driver: MBus Modules

In order for the device to change without latency in response to the user interface, a custom bus protocol, known as MBus, was designed to operate at a minimum of 135 kHz. This is the primary design challenge within this project, and will be described in further detail in the design section. Additionally, each module contains an LED driver designed for operation with the custom bus protocol. This LED driver must provide up to a maximum of 60 mA of current to the LED, as well as provide linear steps of current as the digital brightness values communicated through the bus increase.

2 Design

The primary design choices were made in the design of the MBus protocol, the user interface, and the LED driver. This section will primarily focus on these design heavy blocks of the device, as the other blocks are more focused on purchasing parts that fit our requirements.

2.1 Design Procedures

2.1.1 Custom Bus Protocol: MBus

In order for our device to operate properly, it was determined that we would need a communication protocol that allowed for individually addressable LED modules to operate at a frequency that was high enough to eliminate visual latency when changing the profile of the light. We define unnoticeable latency to be a total latency less than the minimum visual latency of human vision (approximately 8 ms). This would mean that, with color temperature functionality added, the bus must be able to communicate a total of 120 bytes of data (3 bytes per LED module) within 8 ms. This requirement yields a total number of clock cycles (including acknowledgements and data alignment cycles) of

$$120x8 + 120 = 1080\ cycles$$

We can then obtain the *minimum* clock speed the bus needs to operate at as

$$\frac{1080}{.008} = 135\ kHz$$

Note that this clock speed is well above the typical I2C transfer speed (100 KHz). For this reason, we decided to opt out of using I2C as the bus communication protocol, and instead implement our own protocol and bus controller to seamlessly integrate with the LED Driver.

2.1.2 User Interface

The primary decisions for the user interface were made based on the best and most intuitive way to perform operations on the light. For that, we determined that it was necessary to have an LCD display, and several buttons and knobs to change the light. The LCD display was chosen because it is the most direct way to indicate to the user what state the light is currently in. The information it would provide is the current profile of the light, current brightness level, and save status. Buttons are an easy, tactile form of interaction with the light, and the knobs provide an intuitive method of adjusting the brightness, temperature, and window position and size.

We note here that the display element of the user interface did not function properly in the final build, and a work-around for displaying information was instead implemented as described below in section 2.2.2.

2.1.3 LED Driver

In order for the custom bus protocol to properly communicate brightness to the LEDs, it was necessary to create a system which allowed us to control the current driving an LED based on an 8-bit brightness value. We chose to use a BJT as it can act as a voltage controlled current source in the active mode. We also chose to use an R-2R ladder as a simple DAC because each module needed to have two of these LED drivers, and using an R-2R ladder would significantly reduce the cost of a single module while still providing the functionality we needed. We also needed to limit the current to the LED to a maximum of 60 mA, and provide linear steps in current as the brightness value changed.

2.2 Design Details

2.2.1 Custom Bus Protocol: MBus

The protocol is very similar to I2C, with some modifications as follows:

- There is no start condition. A start condition is effectively assumed automatically following a stop condition (note that this means the master cannot sleep the bus).
- The bus is write-only. Data flow modifiers were removed from the communication protocol, and the SCL line is solely controlled by the master (there can be no clock stretching).
- The bus is application specific, and as such does not support multiple masters and is not limited in speed by the masters/slaves it supports.
- The bus supports multiple-slave addressing and variable-size data streams. The application benefits greatly from allowing the brightness of multiple LEDs to be changed concurrently rather than in repeated fast succession, and leaving the data stream open for subsequent data removes the need for the bus to re-initiate communication by re-addressing the modules.
- The bus has a third line – Data Enable. This line allows the master to determine when the data stream terminates and closes.

A schematic of the LED module (MBus controller together with the LED drivers) can be seen in Figure 2.2. Note that in order to provide color temperature functionality, it was necessary to cascade two of the serial input parallel output registers together, and attach an LED driver to each. Thus, when wanting to communicate data to both LEDs, a 16-bit value would need to be sent along with the address of the module, and no particular address exists for each individual LED. A simulation waveform of the design in SystemVerilog can be seen in Figure 2.1. The simulation shows the addressing of two modules (addresses 01010101 and 10101010 to show that no aliasing effects occur) and their respective acknowledgements at the stop conditions. Note that the initialization of the bus causes a HIGH on SDA to be clocked at the start of the waveform.

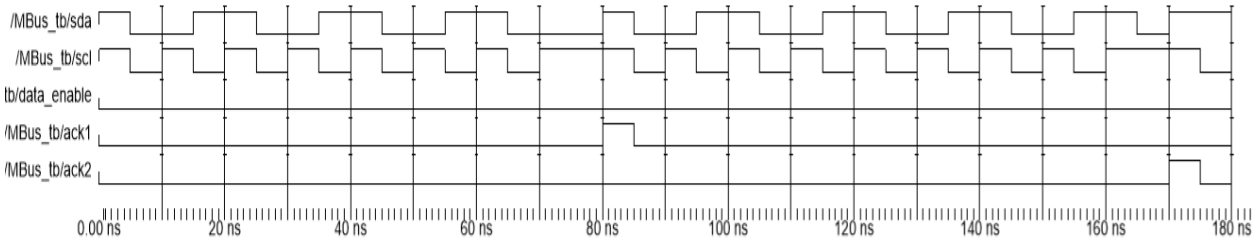

Figure 2.1 SystemVerilog Simulation of MBus Protocol

2.2.2 User Interface

The user interface primarily interfaces with the microcontroller so that the values of the LEDs can be altered depending on the inputs received from the buttons and knobs. A schematic is shown in Figure 2.3 depicting the connections of the buttons, knobs, and LCD display to the microcontroller, alongside the MBus and JTAG setups and the flash memory. The pins are connected to female headers to allow for interfacing with the UI elements. The LCD display is connected to a Molex flat flex cable connector.

The primary design work was done in software. In order to make sure that the signals from the user interface were properly received, it was necessary to de-bounce the inputs from the buttons and knobs to ensure that only a single correctly-identified interrupt was received when either were used. A hierarchy for the interrupts from the user interface was not necessary; CPU time for each service routine was itself interruptible, and as such poses no need for a preset hierarchy.

After experimentally determining the proper strength for redundancy checks during de-bouncing and ensuring that the interrupts were being received as expected, we needed to design ways for the user to understand how the light was operating when using the interface elements without the LCD display. As such, we incorporated some sequences for the light to perform when changing between modes. A list of the indication displays can be seen in the table below.

Table 1 Indication Displays

Power-On/Initialization	All modules are turned on/off in quick succession down the length of the device. If the flash memory passes an integrity check, the end of the succession is noted by a blue temperature for the entire device (red if failed).
Knob Function Change: Adjusting Brightness	All modules are flashed on/off twice in 1 second.
Knob Function Change: Adjusting Color Temperature	All modules are alternated between 3000K and 5700K twice in 2 seconds.
Knob Function Change: Adjusting Window Size	All modules turn on in succession, with the center of the device lighting first and extending to the ends before turning off while returning back to the center.
Knob Function Change: Adjusting Window Position	All modules are turned on/off in a sliding fashion down the length of the device and back.
Navigating Away With Unsaved Changes	All modules are turned on/off at 3000K (red) twice in 1 second.
Save Successful	All modules are turned on/off at 5700K twice in 1 second
Save Failed	All modules are turned on/off at 3000K twice in 1 second.

A software library was designed to interface with the flash memory according to its specifications in the data sheet. This allowed us to store and load profiles from memory. Two of the buttons allowed the user to cycle up and down through the list of 8 profiles, with 4 of them being preset as per the end-user's request. The third button saves the profile over the current one. As noted above, when attempting to switch away from an unsaved profile, the light will flash in a certain sequence, warning the user that they are about to switch away from the profile. If the user attempts to switch away again, the light will switch profiles without saving.

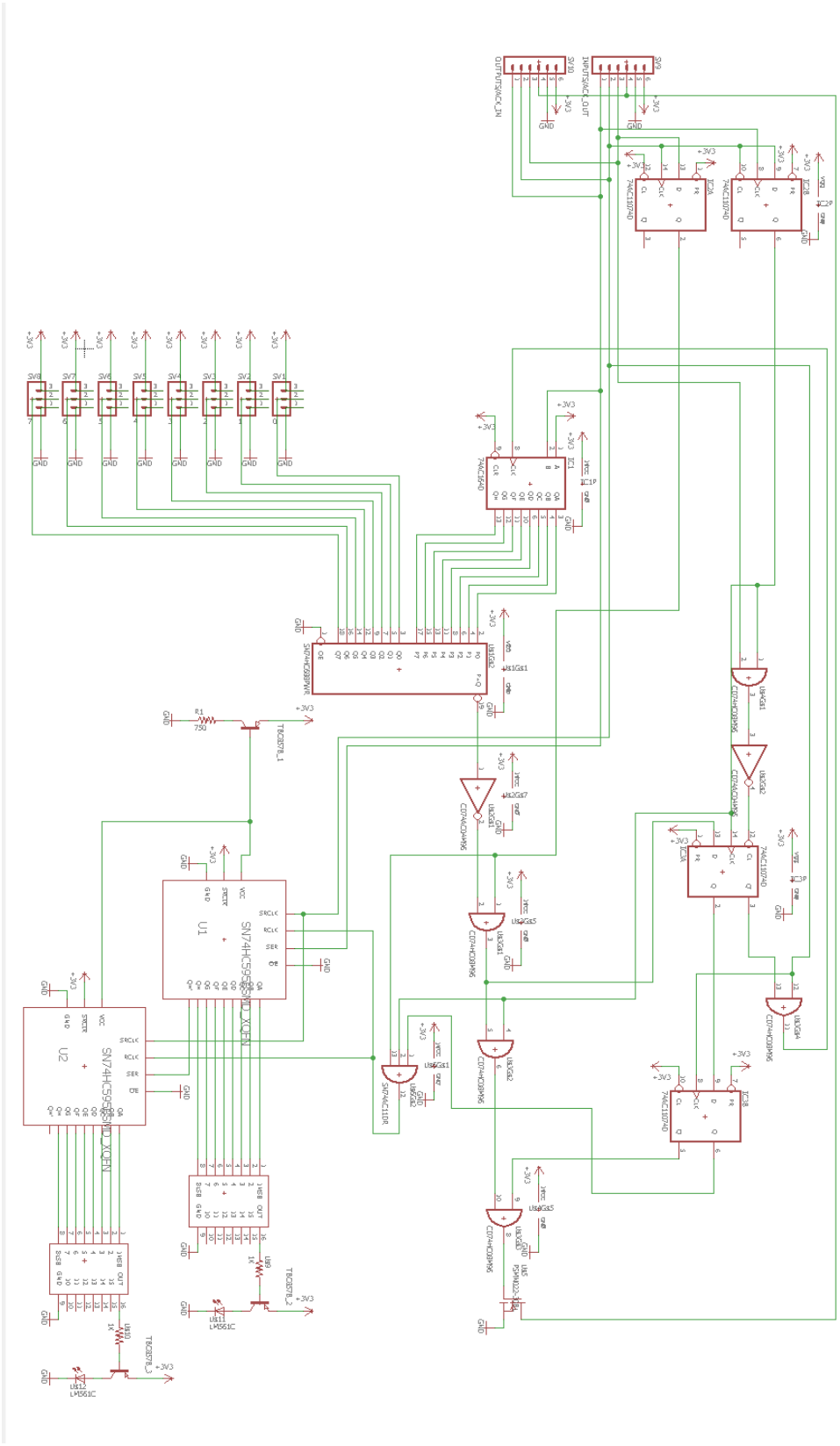

Figure 2.2 MBus Controller and LED Driver Schematic

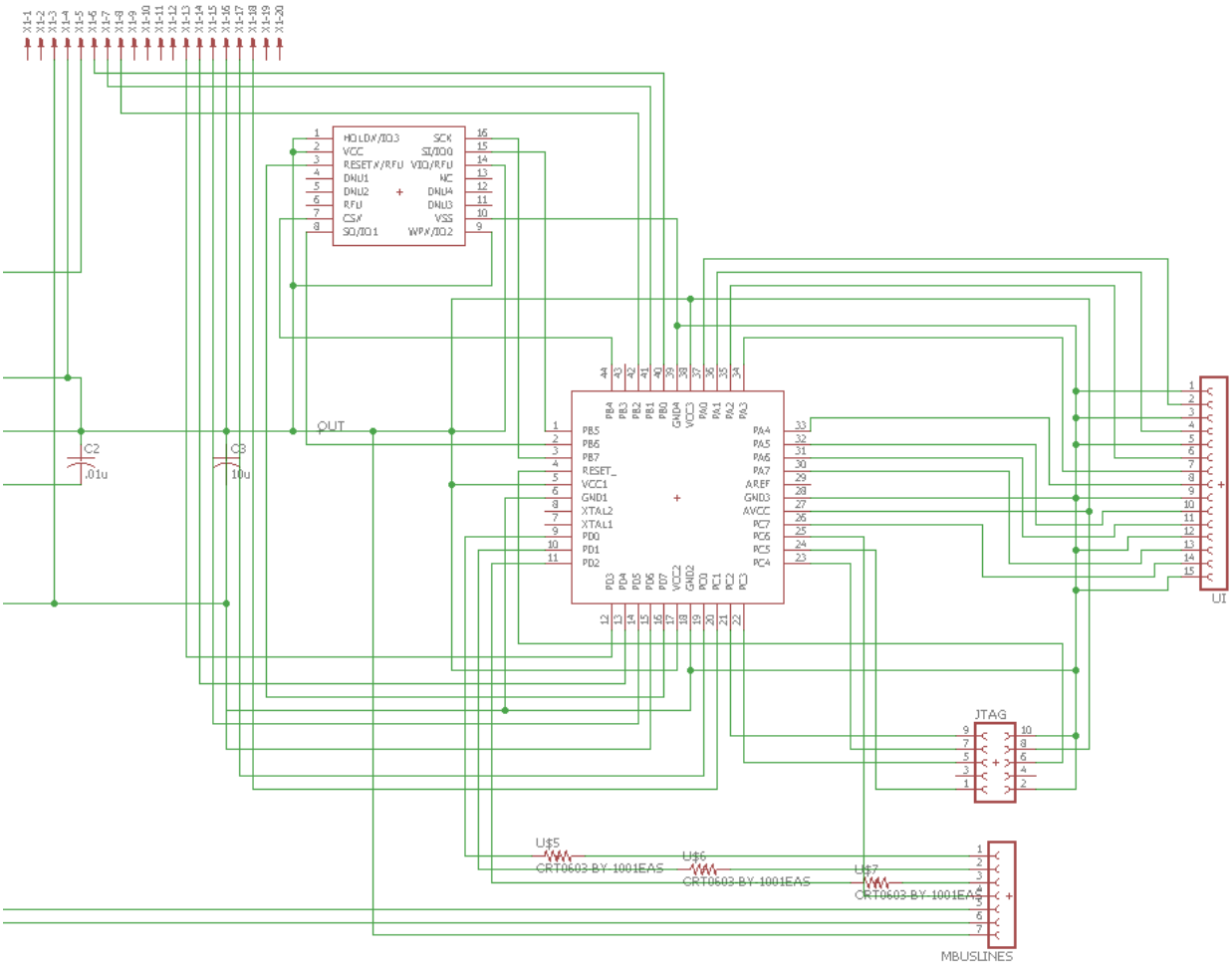

Figure 2.3 User Interface Schematic

2.2.3 LED Driver

The LEDs are driven by using an 8-bit register to store their brightness value. This 8-bit value is used as a parallel output to an R-2R Ladder DAC, which inputs into the gate of a PNP BJT in active mode (the setup is an approximate VCCS). Depending on the brightness value stored in the register, a specific current value flows through the LED. The LED driver must be able to provide a current to the LED of up to 60 mA when the R-2R ladder provides the minimum value of voltage, 0 V (this allows the LEDs to be on during power-on and by default), and regulate the current in linear current steps of fixed size down to 0 A when the R-2R is at the maximum V.

An 8-bit R-2R ladder can be iteratively reduced by applications of Thevenin's theorem to an equivalent circuit of Thevenin voltage given by

$$V_T = \sum_{i=1}^{8} \frac{V_i}{2^i} \qquad (2.1)$$

in series with a Thevenin resistance value of R. A diagram of the equivalent circuit can be seen in Figure 2.4. Using the small-signal model of a PNP BJT, we can obtain an expression for the base current I_B given by

$$I_B = \frac{2.9 - V_T}{R_{ladder} + R_B} \qquad (2.2)$$

from which we can obtain an LED current of

$$I_C = \beta I_B \qquad (2.3)$$

Solving for the requirement that $I_C = 60\ mA$ with $V_T = 0$ yields

$$R_{ladder} + R_B \approx 14\ K\Omega$$

In other words, the total input resistance to the base must be 14 kΩ. Note, however, that these calculations are rough estimates and do not account for the appreciable decrease in the DC current gain as the collector current increases or the battery voltage depletion from 4.2 V. We thus expect the actual values to be somewhat *below* the required range.

We simulated our design in LTSpice with the calculated value in order to obtain a more accurate representation of the physical implementation. The collector current as a function of the R-2R digital input shows an approximately linear relationship that is ideal for our requirements and application, but falls short of the required maximum draw of 60 mA at only 45 mA. The simulation plot can be seen in Figure 2.5.

We instead chose an input resistance of 1kΩ in order to improve the range provided by the BJT (a lower base resistance would allow for a higher base current that offsets the decrease in the DC current gain). Figure 2.6 shows the simulation results for this circuit, demonstrating that the requirement of 60 mA is met for an input voltage of 0 V. Note that as the current range increases, the linear behavior of the relationship starts to fade. For our application, the linearity in the 0-60 mA range is sufficient. Current ratings for the LED are also satisfied at the maximum voltage of 4.2 V, as shown in Figure 2.7.

Finally, we needed to ensure that the maximum output voltage of the R-2R ladder is matched to about .7 V below the battery voltage so as to maintain a full 256 digital value range in the BJT's forward-active region. We achieved this by stepping down the supply voltage to the 8-bit registers with a second BJT forced into active-mode. Although rough as a solution, the approximate .7 V step-down (together with minimal losses from the register) was close enough for our purposes.

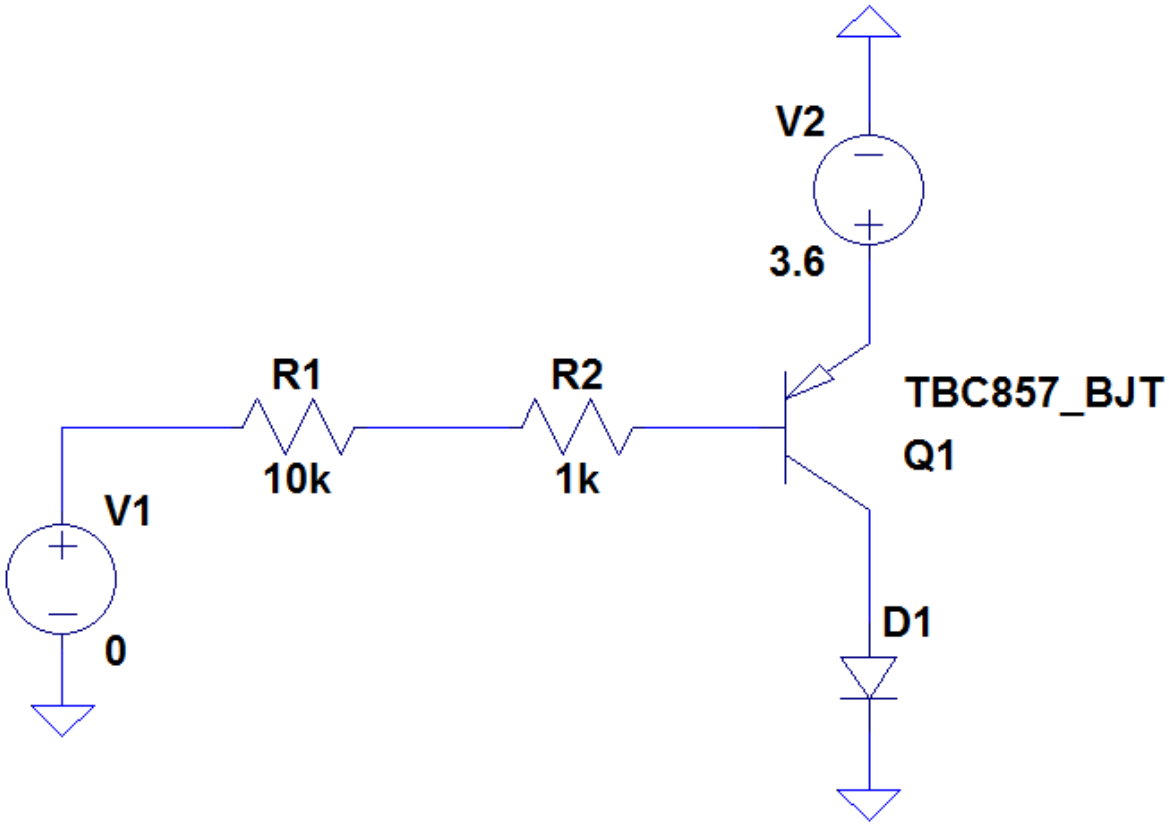

Figure 2.4 LED Driver Schematic

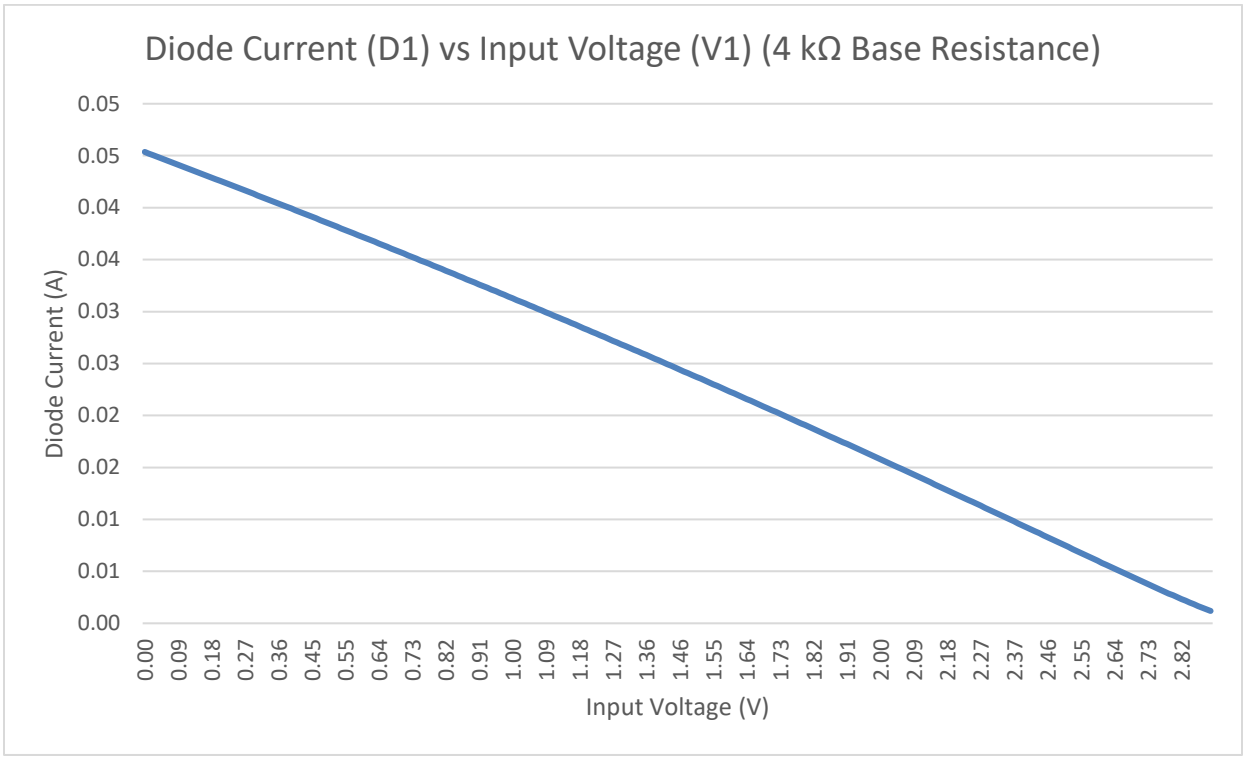

Figure 2.5 Diode Current (D1) vs Input Voltage (V1) for R_B = 4 kΩ

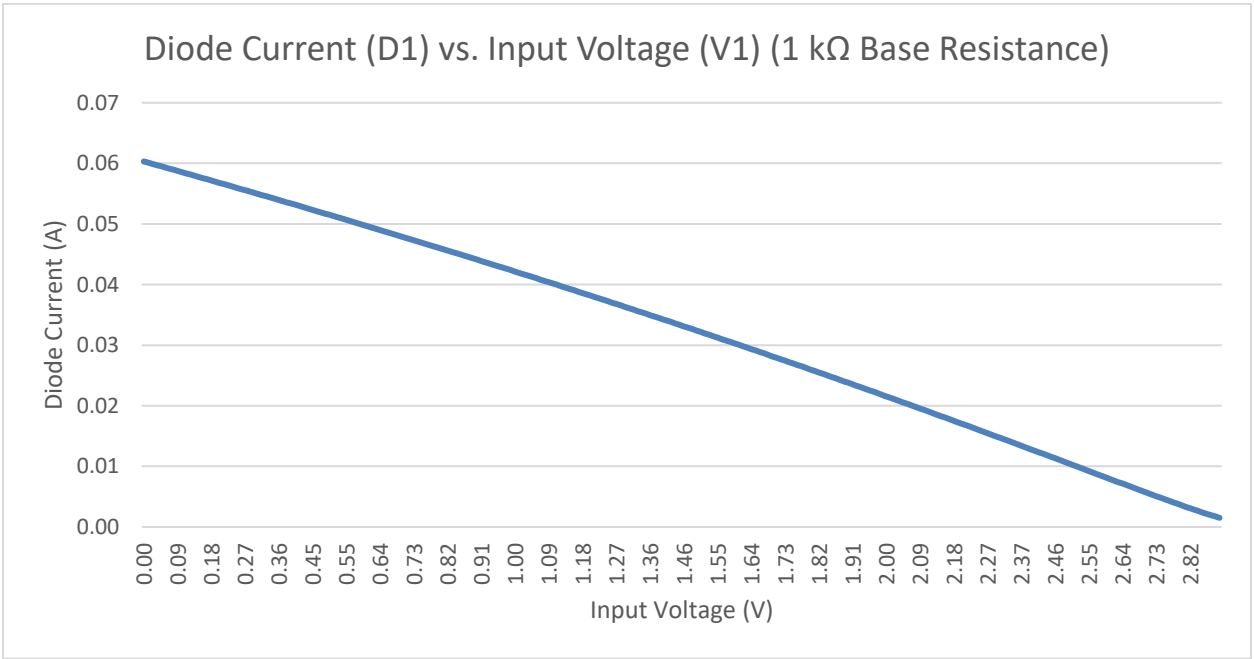

Figure 2.6 Diode Current (D1) vs. Input Voltage (V1) for R_B = 1 kΩ

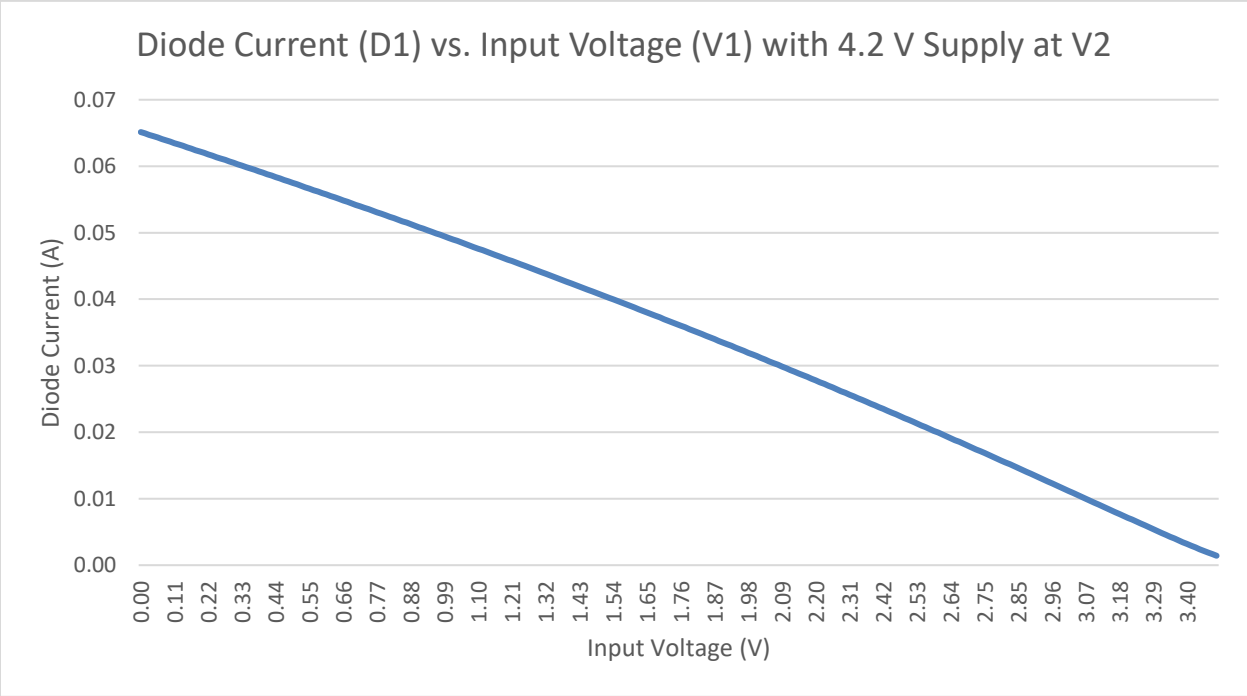

Figure 2.7 Diode Current (D1) vs Input Voltage (V1) with 4.2 V Supply at V2, R_B = 1 kΩ

3. Design Verification

3.1 Custom Bus Protocol

To verify the operation of the protocol, we first performed a SystemVerilog simulation of the schematic shown in figure 2.2. The resulting waveform is shown in figure 2.1. The waveform demonstrates that we are able to properly address two modules with different addresses, and receive an acknowledge signal from both. We also constructed the design on a breadboard, verified the operation of the protocol through an oscilloscope, and communicated to an LED with a simple Arduino program.

Additional verification of the operation of the protocol is largely visual, and done through programming of the microcontroller. Verifying that the protocol works in practice is as simple as programming the microcontroller in accordance to the protocol, and attempting to adjust the brightness of LEDs with different addresses. We can also attempt to adjust the color temperature of the LEDs by making one of the 3000 K or 5700 K LEDs brighter or dimmer, or by performing single-bit feeds into the registers which verifies the data packet size flexibility of the protocol.

We can just as easily verify the lack of visual latency. Once the functionality of the user interface was established, we continuously switched between two profiles very quickly, and experienced no noticeable latency. An LG V20's high-frame-rate camera (operating at 120 FPS, or approximately 8 ms frames) was used to record the switching process, and no latency was captured in the process.

3.2 User Interface

The verification of the operation of the user interface is again largely dependent on proper programming of the microcontroller. As mentioned above, we needed to make sure that the microcontroller was properly receiving interrupts from the user interface elements, and we did this by trial and error of different de-bouncing strengths.

One of the requirements that we failed to verify was the operation of the LCD display. Because the PCB connections of the Molex connector was incorrect, we were unable to provide sufficient power to the LCD display. In order to compensate for this, we programmed some additional functionality which allowed the user to see what was being changed through indications on the light, rather than through the display.

3.3 LED Driver

Verification of the LED driver was done first through simulation, and then through the actual performance of the LEDs once the modules were assembled. As demonstrated in section 2.2.3, while going through the design stages of the driver, we made sure to simulate and verify that our theory was correct. We were able to demonstrate that the LEDs produced linear brightness changes based on the 8-bit values that they received by leveraging a photoresistor within a smartphone to detect their luminosity. A plot of the data obtained is shown in figure 3.1. Note that even though the conversion between voltage to a measure of luminosity is inaccurate, the plot would only fail by a constant scaling factor and linearity would be maintained.

Lastly, in order to verify that the LEDs remained at a safe temperature, the light was left on for slightly over an hour. Since the unsafe temperatures of the LED would be hot to the touch, we simply made sure that the LEDs did not feel more than lukewarm.

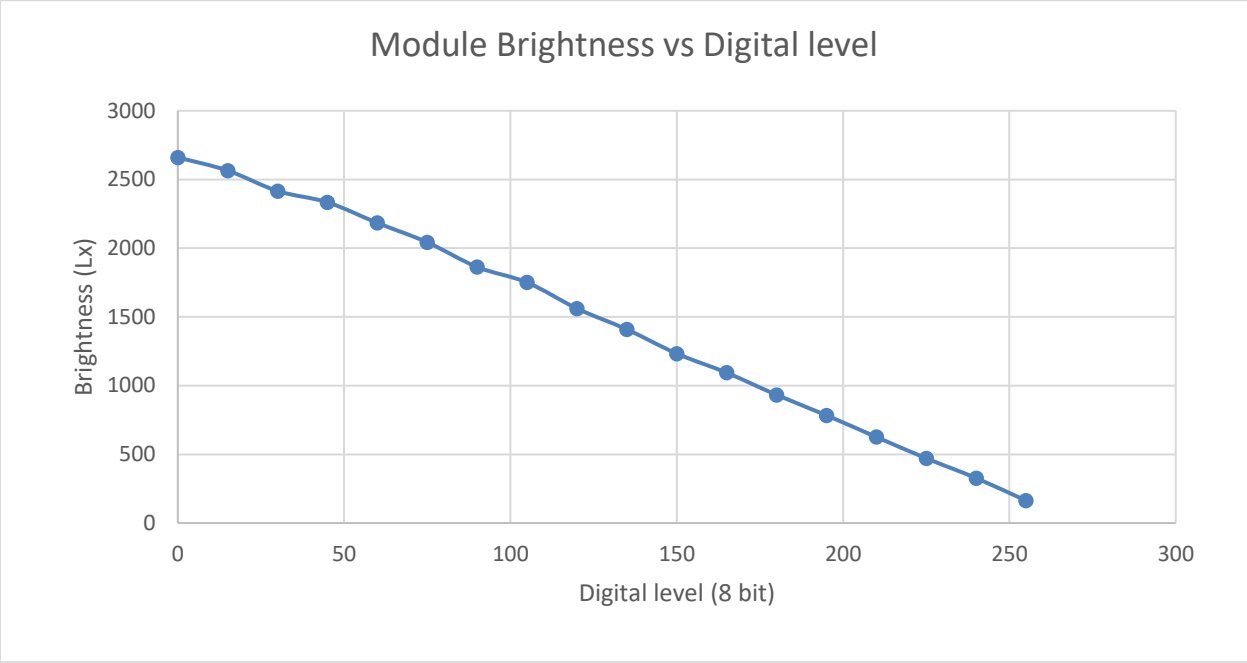

Figure 3.1 Module Brightness vs Digital Level for a single module

4. Costs

4.1 Parts

Table 2 Parts Costs

Part	Manufacturer	Retail Cost ($)	Quantity	Bulk Purchase Cost ($)	Actual Cost ($)
SN74HC595B 8 bit Shift Register	Texas Instruments	0.50	40	0.50	20.00
4816P-R2R-103LF R-2R Ladder	Bourns	1.85	80	1.85	148.00
PSMN022-30BL,118 N-MOSFET	Nexperia	0.84	40	0.84	33.60
74AC164D 8 bit SIPO Shift Register	Texas Instruments	0.65	40	0.65	26.00
S25FL128S 128 Mbit SPI Flash Memory	Cypress	2.42	1	2.42	2.42
ATMEGA644P 8 bit Microcontroller	Atmel	5.56	1	5.56	5.56
74AC11074 Dual D type Flip Flop	Texas Instruments	2.20	40	2.20	88.00
SN74AC11DR Triple 3 Input AND Gates	Texas Instruments	0.52	40	0.52	20.80
CD74HC08M96 Quad 2 Input AND Gates	Texas Instruments	0.52	80	0.52	41.60
SN54HC688 8 Bit Identity Comparator	Texas Instruments	0.88	40	0.88	35.20
CD74AC04M Hex Inverters	Texas Instruments	0.57	40	0.57	22.80
WHITE SMD LED, 3000 K LM561C	Samsung	0.50	40	0.50	20.00
WHITE SMD LED, 6500 K LM561C	Samsung	0.50	40	0.50	20.00
TBC857 PNP BJT	Toshiba	0.16	120	0.16	19.20
CRT0603-BY-1001EAS 1K SMD Thick Film Resistor	Bourns	0.55	120	0.55	44.00
Li-Ion 3.7V 1.5A Battery Pack	Tenergy	34.99	1	34.99	34.99
04026C105KAT2A 1uF SMD Ceramic Capacitor	AVX	0.25	1	0.25	0.25
LT17663-3.3 Voltage Regulator IC	Linear Technologies	12.19	1	12.19	12.19
C1206C103JARACTU .01uF SMD Ceramic Capacitor	Kemet	0.12	1	0.12	0.12

Part	Manufacturer	Retail Cost ($)	Quantity	Bulk Purchase Cost ($)	Actual Cost ($)
TCM0J106M8R 10uF SMD Tantalum Capacitor	ROHM	0.40	1	0.40	0.40
Smart Charger for Li-Ion Battery Packs: 3.7V	Tenergy	18.17	1	18.17	18.17
Break Away Male Headers – 40 pin	Sparkfun	1.43	14	1.43	20.02
Break Away Female Headers – 40	Sparkfun	1.50	14	1.50	21.00
ATAMEL-ICE Debugger	Atmel	133.70	1	133.70	133.70
2.54mm Standard Computer Jumper Caps 100 pack	Corporate Computer	6.39	4	6.39	25.56
20x2 Parallel Character LCD	Crystalfontz	15.86	1	15.86	15.86
Rotary Encoder	Sparkfun	2.95	3	2.95	8.85
Black Metal Knob	Sparkfun	1.50	3	1.50	4.50
Tactile Button Assortment	Sparkfun	4.45	1	4.45	4.45
20 pin Vertical FPC Connector	Molex	4.17	1	4.17	4.17
20 pin Flat Flex Cable	Parlex	4.05	1	4.05	4.05
Total					**$855.46**

4.2 Labor

For our project, we are estimating a salary of around $20 an hour. Assuming that we spend 20 hours a week, the same as a part time job, working on the project, with about 10 weeks of labor put into the project since the design review:

$20/hour x 2.5 x 200 hours to complete x 2 partners = $20,000

Additionally, we had the housing designed by the machine shop, which we estimate took about 10 man-hours. Estimating another hourly salary of the machine shop at 25 dollars an hour, this would add a cost of 250 dollars.

Totaling the cost from table 2 detailing the parts cost, we estimate a total cost of **$21,105.46.**

5. Conclusion

5.1 Accomplishments

Overall, we were able to properly communicate with and control the LED modules using our custom MBus protocol. There was minimal visual latency, and the LEDs behaved exactly as we expected when sending brightness values. We also found that the maximum brightness of the LEDs was quite great, meaning that there is more potential for producing the desired lighting setup. Color temperature functionality was successfully implemented, allowing us to produce light anywhere between 3000 K and 5700 K. The LED driver also produced enough variation of the brightness to distinguish between the maximum and minimum brightness, to properly produce a gradient.

The user interface elements also performed quite well, as we were able to program the microcontroller and precisely determine their behavior. They also produced the expected changes to the light, and were quite responsive when used. Our usage of the flash memory was successful as well, as we were able to grant the user access to 8 different profiles, with 4 being preset, and the other 4 being blank for the user's customization. The choice of 8 profiles was completely based on user-comfort (to avoid cycling through a long list), and the flash memory would support well over this amount.

Finally, we were successful in coordinating with the machine shop to produce our desired housing. Although it was somewhat bulky and heavy, it was a good prototype to demonstrate operation of the device, and how it would theoretically be used.

5.2 Failures and Shortcomings

Unfortunately, we were unable to deliver on the promised number of 40 LED modules over the entire 4 foot strip. Due to time constraints, difficulty of soldering, and the need to debug individual modules, we were only able to produce 16 operating modules. However, since we created the design of the device to be modular, and since our bus protocol supports up to 256 modules due to the 8-bit addressing, the only thing preventing the additional modules from being added is time and the work put into soldering them.

We were also unable to deliver on a properly working LCD display, due to making the wrong connections to the Molex flat cable connector. To fix this, all it would require is changing a connection from the regulated output from the battery to the unregulated output, allowing us to produce enough of a contrast on the LCD display.

During the testing of our device, we also encountered a very elusive bug that was seemingly random and that we were unable to solve; when communicating through MBus, some of the modules that were not being targeted would turn on to maximum brightness and return to the correct state after another packet propagated down the bus. We initially theorized that this was due to data misalignment on the bus, causing 0's to be clocked into the registers and turning the LED(s) to maximum brightness. This, however, did not explain why the modules would return to their correct state even after the packet-propagation that fixed them did not load into their registers. Our current and best explanation for this weird behavior is inappropriate tri-stating of the registers holding the brightness values; the ICs are

capable of disabling their outputs upon instruction by a dedicated pin, and we believe that this is the cause for the momentary maximum brightness that is observed when sending packets down the bus. The reasons for the disabling remain unknown.

5.3 Ethical considerations

Every component and part that we design or use should be RoHS compliant. This falls under the ACM Code of Ethics Section 1.2[1], which means that our product should avoid harming the end user. By making the product RoHS compliant, we would avoid using dangerous chemicals and substances in our design.

In terms of safety concerns, our greatest involves the usage of Li-ion batteries. These batteries could pose a threat if not charged and used properly. We have turned to industry to provide the necessary guidelines and equipment to safely house, use, and charge the batteries in the device. This includes protection from both thermal and voltage-induced runaway events, preventing potentially harmful explosions and fires. Also, we will work to make sure that there is no potential for sudden shorts and opens by designing our circuit and carefully verifying each portion, across all ranges of potential usage. By doing this, we ensure that no matter what the user does with the device, within reason, they would not be injured.

The user should also be mindful of how the batteries are handled, taking care to properly add and remove the batteries from the device when charging. Ensuring the terminals are connected in the correct polarity, and one at a time, will guarantee proper usage of the device. Also, the user should make sure to not leave the device in temperatures above 40 degrees Celsius. Finally, when charging the battery pack, only the specified charger for the pack should be used, and the user should make sure to remove the charger when the peak voltage is reached, and use a correct wall input voltage.

The device can also produce a brightness that could be potentially dangerous to the eye; the user should take care to avoid looking at the light for too long, or use the proper eye protection equipment when the device is at maximum brightness.

Finally, since the device is somewhat heavy, the user should take care to always keep the side without a handle towards the ground, to ensure that they do not injure themselves during usage of the device.

5.4 Future work

For further work, the primary goal would be completing the implementation of the LCD display in the user interface. This would require some changes made to the microcontroller PCB, as well as programming the microcontroller to properly interface with the display.

We would also like to improve the diffusive layer of our device to produce a better, more defined gradient, and reduce shadows from the components inside. To help with this, we would like to produce a more aesthetically pleasing and lighter housing through some other method like 3D printing.

Finally, assembling all 40 modules as intended would be the final goal. This would allow the light to produce its maximum potential brightness, as well as allow for more well defined gradients.

References

[1] ACM Council, "ACM code of ethics and professional conduct," in Association for Computing Machinery, 1992. [Online]. Available: https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct. Accessed: Feb. 25, 2017.

[2] I. Buchmann, "How to prolong lithium-based batteries," in Battery University, 2017. [Online]. Available: http://batteryuniversity.com/learn/article/how_to_prolong_lithium_based_batteries. Accessed: Feb. 25, 2017.

Appendix A Requirement and Verification Table

Table 3 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
The battery must power the device for at least 45 minutes at full brightness. 1. Must be able to provide a minimum of 20 Watt-hours (~5,500 mAh) at a voltage range of 3.6 – 4.2 V with a discharge current of 1C (5 – 6 A). The battery should have a protection circuit to prevent overcharge, over-discharge, and over current protection.	1. Turn the device on and set all LEDs to max brightness. 2. Use a Voltmeter to measure the battery voltage after 45 minutes and verify that the output is above 3.6 V.	Y
The charger must charge the battery from 3.6 V to full *safely* under 5 hours. Must be supplied directly from an AC wall outlet and be external to the device.	1. Use a Voltmeter to measure the battery voltage and verify that the output is below 3.6 V. 2. Charge the battery for a period of 5 hours and verify with a Voltmeter that the output has reached the maximum 4.2 ± .02 V.	Y
The flash memory must have a capacity of *at least* 1600 bytes (160 bytes per profile over 10 profiles). Must be SPI ready. Must operate quickly enough to avoid latency to the human eye when loading modes from memory. - Must provide read and write speeds of *at least* 160 Kbps clocked at 5 MHz	1. Cycle between modes, and verify that there is no visual latency when switching.	Y
The voltage regulator must be able to regulate the input levels to the flash memory to meet component requirements of 3.3V.	1. Supply a voltage within the range of 4.2 – 3.6 with a DC power supply. 2. Measure the output of the regulator with a Voltmeter and verify the output is at 3.3 V.	Y

Requirement	Verification	Verification status (Y or N)
LCD Display The LCD Display should readable in a low light environment. Should convey information regarding current state of the light. - Should show current brightness level, current profile, and save status. **Buttons** Profile Toggle Function - Profile toggling will cycle through the profiles as they are displayed on the LCD display and stored in the flash memory. Saving Profiles - User must have the ability to save the current profile into flash memory or to discard it. The user will have access to 8 different modes, and these modes should be able to be overwritten. **Rotary Encoders** Their rotation must be *effortless* but robust enough to prevent accidental turning. Window adjustment - The user should have access to a window of LEDs of flexible size and position that allows for their full customization. Brightness/Color Temperature Adjustment - The user should be able to adjust the brightness and color temperatures of the LEDs within the window that they have selected.	1. In a dark environment, display 40 different characters with the backlight on. 2. Hold at arm's length and verify readability. 3. Use the buttons to cycle through the various profiles offered by the device. 4. Verify that the information of the current profile is updated on the LCD Display. 5. Select an arbitrary window and adjust its brightness and color temperature with the rotary encoders. 6. Verify that the information of the current mode is updated on the LCD Display. 7. Save the current profile using the buttons. 8. Turn the device off, and attempt to load the saved profile. 9. Verify that the loaded profile is the same as the previously saved profile.	N, All Except LCD Display

Requirement	Verification	Verification status (Y or N)
MBus Protocol Must allow the microcontroller to establish communication with the LEDs using the MBus protocol. Should minimize latency as detected by a camera for more time sensitive future applications. - Should allow multiple LEDs to be addressed simultaneously. Should allow for color temperature adjustment through communication to multiple LEDs simultaneously. Color temperature should range from about 3000 K (warm, red light) to 5700 K (cool, bluish white). - Data packet size must be flexible (8 or 16 bits). Must be able to communicate information to all of the LEDs below the average latency of the human eye. - Must be able to operate at frequencies above 140 KHz.	1. Change the brightness of multiple LEDs and verify that there is no visible latency. 2. Change the color temperature and verify that there is no visible latency. 3. Reset the brightness of all LEDs simultaneously and verify that there is no visible latency through a camera to	Y
LED Driver Must be able to provide a linear digitally controlled brightness values for the LED. - The current values must then be linear, ranging from 0 to 60 mA because the LED brightness depends on the current value that is driving it. The LEDs should remain at a safe temperature as specified in the datasheet. - Must not exceed a maximum continuous forward current of 65 mA.	1. Vary the brightness values of the LED over the entire range. 2. Verify the linearity in the steps of brightness. 3. Operate the device at maximum brightness for the duration of the battery life, and ensure that the LEDs do not feel more than lukewarm.	Y

Appendix B Full User Interface Schematic