Color Science Demonstration Kit from Open Source Hardware and Software

Michael W. Zollers, Synopsys, Inc., 377 Simarano Dr., Marlborough, MA 01752

ABSTRACT

Color science is perhaps the most universally tangible discipline within the optical sciences for people of all ages. Excepting a small and relatively well-understood minority, we can see that the world around us consists of a multitude of colors; yet, describing the "what", "why", and "how" of these colors is not an easy task, especially without some sort of equally colorful visual aids. While static displays (e.g., poster boards, etc.) serve their purpose, there is a growing trend, aided by the recent permeation of small interactive devices into our society, for interactive and immersive learning. However, for the uninitiated, designing software and hardware for this purpose may not be within the purview of all optical scientists and engineers.

Enter open source. Open source "anything" are those tools and designs -- hardware or software -- that are available and free to use, often without any restrictive licensing. Open source software may be familiar to some, but the open source hardware movement is relatively new. These are electronic circuit board designs that are provided for free and can be implemented in physical hardware by anyone. This movement has led to the availability of some relatively inexpensive, but quite capable, computing power for the creation of small devices.

This paper will showcase the design and implementation of the software and hardware that was used to create an interactive demonstration kit for color. Its purpose is to introduce and demonstrate the concepts of color spectra, additive color, color rendering, and metamers.

Keywords: Color Science, LEDs, Education, Outreach

1 INTRODUCTION AND MOTIVATION

I've given a lot of optics demonstrations on a lot of different optical subjects through my involvement with the New England Section of the Optical Society of America (NES/OSA). Optics is a complex discipline and many of the topics that an optical engineer finds interesting or fundamental are not easily explained to a non-engineer. Consequently, some demos work well; others do not. To that end, the NES/OSA is always trying to find ways of improving the demos we provide to better engage a wide variety of age groups -- from pre-school aged children to the curious grandparent and everyone in-between -- in such a way that they learn important optical concepts while having fun.

The NES/OSA demo suitcase [1], many suitcases actually, contains several demos that cover different facets of color science. These demos are some of the most popular that we use, perhaps due to the nearly universal experiences we all have with color. We are surrounded by color, and yet very few understand why it is so. The generation of color through the process of mixing paints and dyes is taught at an early age, but most of the other key concepts of color science are missed. The demos within the suitcase endeavor to rectify the shortcomings of this education. One such demonstration teaches additive color by allowing the eager learner to superimpose any combination of red, green, or blue light to make a variety of different colors.

2 GOALS

The goal of this project is to replace this simple additive color mixing demo. While it is easy to see what happens during the demo, we lack many of the visual aids and intensity controls to make or explain anything more than the seven possible combinations with the three, fixed-intensity, light-emitting diodes (LEDs). Simply put, the principle goals of this new demonstration are as follows:

- Demonstrate additive color using a variety of primary colors with variable intensity.
- Demonstrate and explain the color gamut for the chosen set of primaries.

Optics Education and Outreach III, edited by G. Groot Gregory, Proc. of SPIE Vol. 9188, 91880C © 2014 SPIE · CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2060833

- Be portable and interactive.

Additionally, the following list contains secondary goals for the kit.

- Explain the correlation between spectral power distributions and additive color.
- Demonstrate and explain metamerism.
- Demonstrate and explain color rendering.

3 THEORY

In the demo system's simplest form, the user chooses a set of light sources from a list and then picks a color to be made from these primaries. The set of primaries define the color gamut, and the chosen color can either be in-gamut or out-of-gamut. If the chosen color is out-of-gamut, we need to display the closest, in-gamut color. This section describes the mathematical models used to make these desire a reality.

The mixing of various levels of a fixed number of light sources is an application of Grassmann's law [1], shown in matrix form for tristimulus values in Equation 1.

$$\begin{bmatrix} X_T \\ Y_T \\ Z_T \end{bmatrix} = \begin{bmatrix} X_1 X_2 X_3 & & X_N \\ Y_1\, Y_2\, Y_3 & \cdots & Y_N \\ Z_1 Z_2 Z_3 & & Z_N \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ \vdots \\ k_N \end{bmatrix} \tag{1}$$

If the desired color, $[X_T, Y_T, Z_T]$, and the color of the N primaries, $[X_1, Y_1, Z_1] \cdots [X_N, Y_N, Z_N]$, are known, simply solving the matrix for the coefficients $(k_1 \cdots k_n)$ gives the ratio of powers required of each of the primaries. Since the device works by changing the power output of LEDs, the coefficients all must be greater than 0 and the positive values are scaled such that $Y_T = 1$ for all cases, as shown in Equation 2. This assures that the output, regardless of the number of LEDs currently powered, is constant in luminance; only the color changes.

$$\begin{bmatrix} X_T \\ 1 \\ Z_T \end{bmatrix} = \begin{bmatrix} X_1 X_2 X_3 & & X_N \\ 1\ \ 1\ \ 1 & \cdots & 1 \\ Z_1 Z_2 Z_3 & & Z_N \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ \vdots \\ k_N \end{bmatrix} \tag{2}$$

There are many cases in which this system of equations has no solution, or an infinite number. If only one LED is illuminated, the color gamut is a point, and all other colors are impossible to produce. If two LEDs are illuminated, the color gamut is a line, and only colors that are linear combinations of the two primaries are possible. If three or more LEDs are illuminated, the color gamut is an area and a solution may be possible. Examples of these three possibilities are shown in Figure 1.

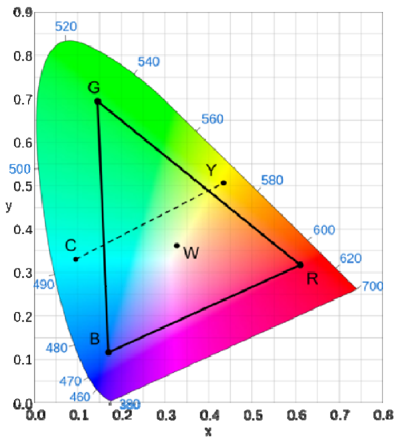

Figure 1. A depiction of various types of possible color gamuts on the 1931 CIE Chromaticity diagram.

In Figure 1, point W is the only color that can be made if only that primary is used. The dashed line between points C and Y represents the color gamut of those two primaries; all other colors cannot be realized. If points B, G, and R are used, then the area within the triangle represents the color gamut. Any color within this region is possible as a linear combination of the primaries; therefore, point W can be said to be in-gamut, while point Y is out-of-gamut and cannot be realized. If points R, G, B, and W are used as primaries, the gamut is the same area – the convex hull of the set of points – but the system of equations in Equation 1 are over-defined, so many solutions are possible.

Consider the four-primary case; there are four coefficients and three unknowns, which is over-defined. The simple approach is to use a least-squares method to find a solution [3]. To begin, generalize Equation 2 as Equation 3, where *C* is the desired color, *P* is the matrix of the four primaries' tristimulus values, and *k* is the coefficient matrix.

$$C = Pk \tag{3}$$

The least-squares solution involves multiplying both sides of Equation 3 by the transpose of *P*, denoted as P^T, and solving for *k*, which is shown in Equation 4.

$$k = (P^T P)^{-1} P^T C \tag{4}$$

This solution is very quick to produce, but can yield negative coefficients for the LED powers. Since negative LED output cannot be realized, a better algorithm must be utilized. One approach is to use the Nonnegative Least Squares algorithm [4]. This algorithm arrives at a solution in which all of the coefficients are greater than or equal to zero.

A more general solution, and the one that was implemented, is to use an optimization algorithm, such as Nelder-Mead method [5]. By using a generalized optimization algorithm, considerations other than inequality constraints on the coefficients can be used, such as maximizing the color rendering index, or minimizing the total system power. With a small number of LEDs, the optimization converges quickly enough to a solution that there is no perceivable lag. In this implementation, the optimization begins with 0.5 applied to all of the coefficients and iterates to a solution.

4 IMPLEMENTATION

This demonstration kit consists of a set of colored light sources that are optically combined such that they mix and are electrically dimmed to form any of the colors within their gamut through the use of an interactive control system. The components of this system are described here as well as the software that was used to create the interactive experience.

4.1 The LEDs and Optical System

The first and most important components of the system are the light sources. Since colored, high-power LEDs are readily available, these were chosen for their saturated color. LuxeonStar.com [6] will populate a printed circuit board (PCB) with seven Philips Luxeon Rebel [7] or Rebel ES [8] LEDs of colors and flux bins of my choosing. I chose the highest flux bin for the blue LED, as it has the lowest total luminous flux of the other color choices, then chose bins for the other colors that closest matched the blue LED's flux. Table 1 shows the LED color and flux bin choices for all seven LEDs.

LED Color	*LED Part #*	*Minimum Luminous Flux (lm)*	*Dominant Wavelength (nm)*	*Spectral Half-Width (nm)*
Blue	LXML-PB01-0040	40	470	20
Cyan	LXML-PE01-0070	70	505	30
Green	LXML-PM01-0070	70	530	30
Amber	LXML-PL01-0040	40	590	20
Red-Orange	LXML-PH01-0050	50	617	20
Red	LXML-PD01-0040	40	627	20
5000K White	LXW8-PS50	180		

Table 1. Part number, minimum flux, dominant wavelength, and spectral FWHM for each of the LEDs used.

Thermal management of the LEDs and their PCB was particularly important for this project. This system needs to operate for an entire science fair, or about 4 hours. This means that the LEDs need to stay cool, lest the light output degrade or the PCB become hot enough to burn an unfortunate youth. Passive cooling is desirable for cost and lack of moving parts. Luckily, I was able to repurpose a heat-sink that was designed for a computer CPU. With the fan removed, the remaining finned heat-sink is a perfect size for the PCB and the wires can be fed through the fins to the back of the device. Figure 2 shows the PCB after it was screwed to the heat-sink. A thermal transfer paste was used between the PCB and the heat-sink.

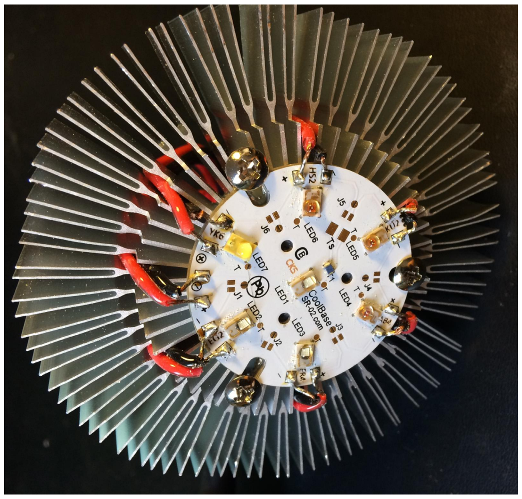

Figure 2. The PCB and heat-sink

To facilitate mixing of the seven LEDs, a simple approach was taken. A 2" white PVC pipe was cut to a length of 12". The resulting cylinder perfectly fit over the PCB and secured onto the fins of the heat-sink. I anticipated that the PVC would be too specular for good mixing, so I painted the interior and exterior of the plastic with a matte white spray paint. The complete light engine is shown in Figure 3.

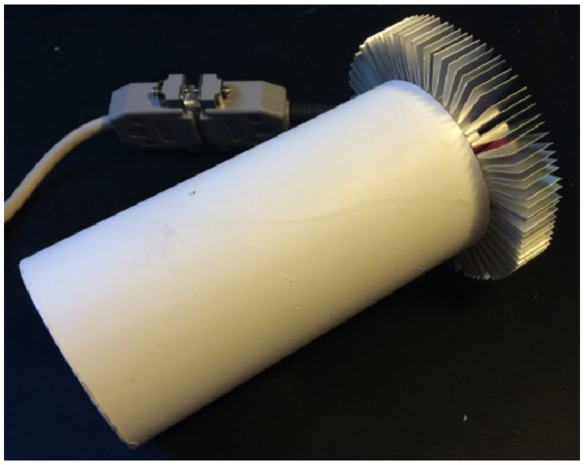

Figure 3. The light engine with the mixing tube.

Sadly, the matte paint proved too specular, and produces results consistent with Figure 4 on the wall. The colors mix in certain regions, but the overall pattern is not mixed. As a result, and to increase the intensity of the beam pattern as projected onto a wall, I chose to use a commercially available collimating lens assembly made by Ledil [9]. Figure 5 shows the lens in place and Figure 6 shows a representative beam pattern on a wall.

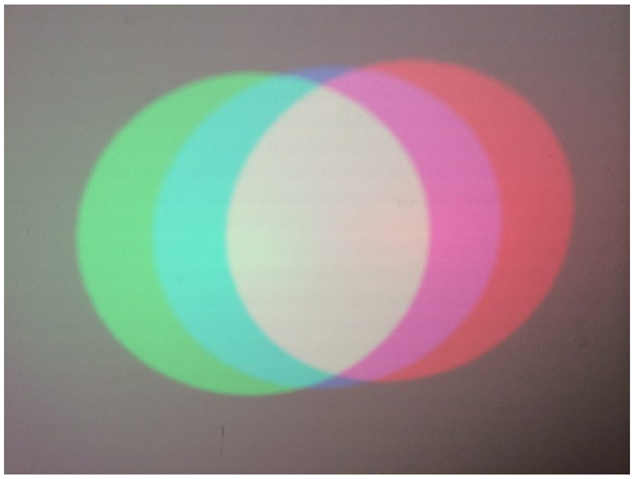

Figure 4. The light output from the light tube mixing system. The output does not mix well.

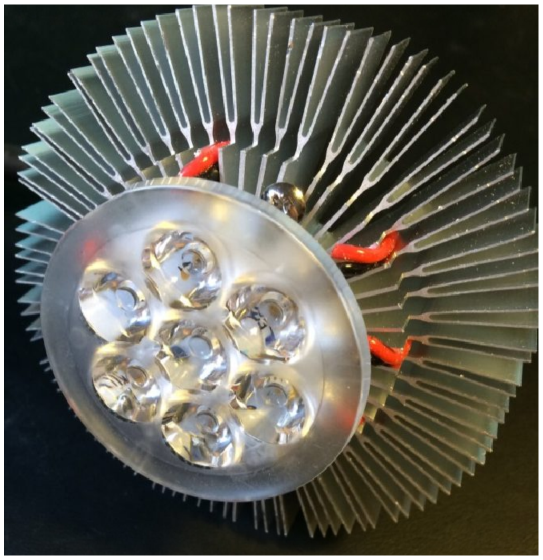
Figure 5. The light engine with the Ledil collimating lens in place.

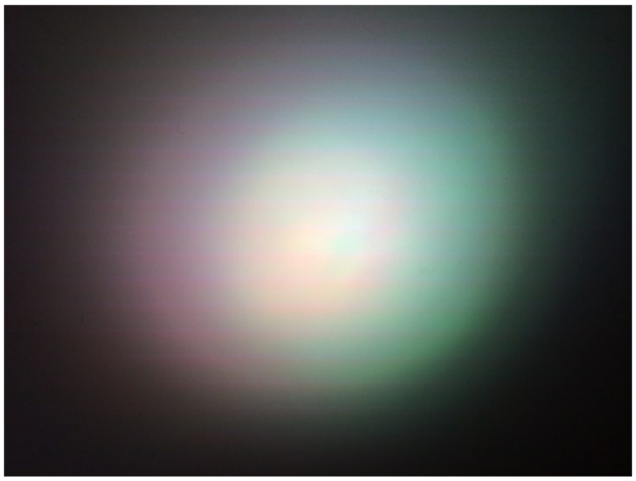

Figure 6. Color mixing result with the Ledil lens. Mixing is improved, but not good.

4.2 Control System

The system to enable/disable and dim various combinations of LEDs consists of electrical hardware and a software control system. The software tells the hardware the appropriate amount of current for each LED.

4.2.1 Electrical Hardware

Two versions of the electrical hardware have been created to date. The initial design uses very generic components and required an elaborate electrical and software design. The second design uses an industry-standard protocol for hardware and software communication. This simplifies things, as specialized hardware was available and didn't need to be created from scratch.

4.2.1.1 Version 1: Generic Components

Under normal operating conditions, LEDs desire constant current. In order to properly power and dim them, I chose to use a commercially available LED driver from LuxDrive [10]. I chose the 350mA driver to reduce thermal load and keep the total flux low enough that there was no risk of eye damage due to excessive luminance. The drivers nominally accept a 12V DC voltage and can be dimmed by applying between 0 and 5V to trigger the internal dimmer circuit. Providing the constant voltage can be achieved by an off-the-shelf power supply. Controlling and sending the proper signal to the internal dimmer required much more consideration and design.

Early in the design, I decided that I wanted to make sure that the drivers were not accidentally powered at a very low level for long periods of time, as I believed that this would be hard on them. As such, I decided to put relays in each channel that can completely remove power from the driver when it is off. I experimented with several 8-channel relay boards (7-relay boards were not available) and found that I did not like the clicking noise of a mechanical relay, so solid state relay boards were chosen. In the end, I chose MOSFET-based switch boards.

As I am not an electrical engineer, designing a complex control circuit with a software interface is not an option. If that was the only option, then this project is dead in the water. Instead, I decided to use microcontrollers developed by Arduino [11]. Arduino is an open hardware platform that contains a micro-controller with many digital and analog inputs and outputs. There are many Arduino boards available from many manufacturers. They are all compatible and largely interchangeable since the hardware design specifications are available for free.

Unfortunately, none of the Arduino boards contain a variable analog voltage output, which is needed to dim the LED drivers. However, they do have the ability to control digital to analog converters (DACs) through several communication protocols. I chose a 12-bit DAC [11] that uses the I^2C communication protocol to convert a digital signal of 0 to 4095 to 0 to 5V. Controlling 7 I^2C devices is not natively available on any current Arduino boards. However, Arduino compatible I^2C controllers [13] are available that allow up to 8 devices to be controlled by one Arduino board. A schematic of the control circuit for a single channel is shown in Figure 7. A list of the hardware used for this configuration is shown in Table 2.

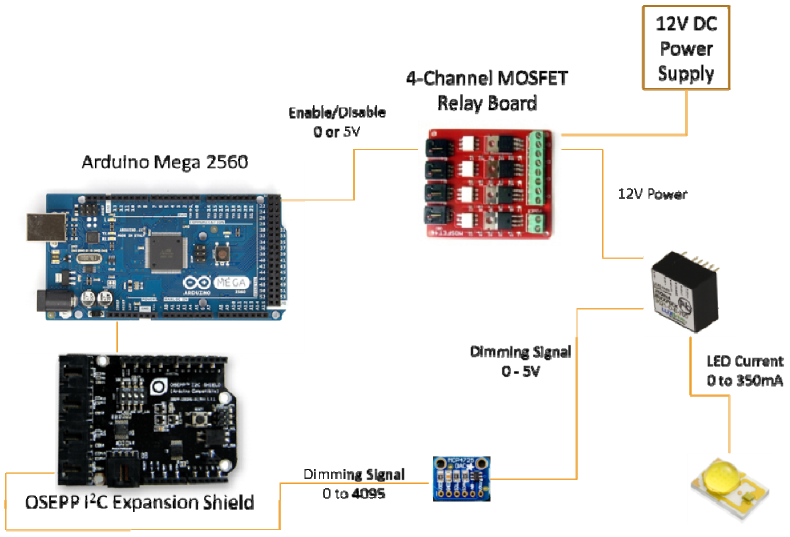

Figure 7. Schematic of one channel of the first hardware configuration.

Quantity	*Item*
1	Arduino Mega 2560
1	OSEPP I^2C Expansion Shield
7	MCP4725 Breakout Boards [14]
7	LuxDrive 3021-D-E-350 LED Drivers
2	4 Power FET Switches [15]
1	12V, 40W DC power supply

Table 2. Parts list for the first hardware configuration.

4.2.1.1.1 Version 1: Lessons Learned

The first opportunity to test the system was during the presentation of a workshop on color. The following list details what was learned from this experience:

- It will sometimes be necessary to operate the system from a laptop or other computer that is being projected onto a screen. A kiosk mode is still necessary, but the hardware and software should be configured for a “presentation” mode as well.
- There is significant lag between choosing the color to be displayed and the production of the color. This is largely due to a mismatch in communication speed between the Arduino and the controlling laptop.
- The relays are not necessary and are problematic. Dimming at level 0 is adequate to turn off the LEDs and removing the relays simplifies the design.

Armed with the experience from the workshop, I reworked the hardware to overcome some of its shortcomings.

4.2.1.2 Version 2: DMX Solution

As learned during the trial run, the relays did not function properly. Also, there was a bit of lag introduced by the serial communication. In the time since the original electrical design and the second implementation, other off-the-shelf hardware resources became available that allow me to rethink the entire electrical design.

LuxDrive has available a 4-channel controller [16] that uses the same LED drivers and allows for dimming control using a standard communication protocol developed by the entertainment lighting market: DMX [17]. It only allows for 8-bits of dimming (256 levels), but has the potential to be much faster. To use these DMX dimmers, a DMX controller is necessary. Velleman makes a DMX controller with a USB interface [18]. This can be attached to any computer with a USB port and software on the system will send the dimming commands to the specified channel. A schematic of the version 2 hardware is shown in Figure 8 and the parts list is shown in Table 3.

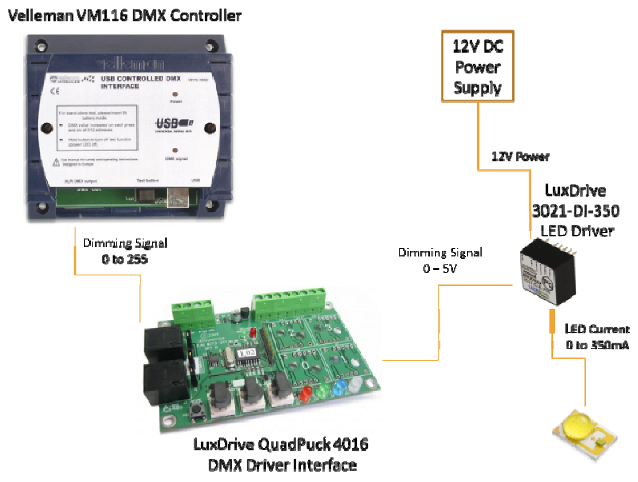
Velleman VM116 DMX Controller
12V DC Power Supply
12V Power
LuxDrive 3021-DI-350 LED Driver
Dimming Signal 0 to 255
Dimming Signal 0 – 5V
LED Current 0 to 350mA
LuxDrive QuadPuck 4016 DMX Driver Interface

Figure 8. Schematic of the second hardware configuration.

Quantity	*Item*
1	Velleman VM116 DMX Controller
2	LuxDrive QuadPuck 4016 DMX Driver Interface
7	LuxDrive 3021-D-I-350 LED Drivers
1	12V, 40W DC power supply

Table 3. Parts list for the second hardware configuration.

4.2.1.2.1 Version 2: Lessons Learned

The version 2 hardware was tested at another color workshop. This configuration performed much better than the first, without lag or issues with the relays. However, the system never achieved the desired colors except when only a primary color was requested. There are two potential reasons for this:

1. 8-bit dimming is insufficient for this type of system, or
2. Non-linearities between input current and light output cause a color shift.

In practice, I assume that both contribute, but a simple calibration step should provide enough information to either fix the problem or point toward the culprit.

4.2.2 Calibration

The light output of the LEDs is non-linear with input current [7]. Also, different colors of LEDs have different flux vs. current curves. In order to produce the correct color, these details must be correctly accounted for in the control system. At the very least, the control system needs a set of flux vs. dimming level for each of the LEDs in the system. These values can be normalized to the flux output of the dimmest LED to make the software controls a little easier.

The system for controlling current is already in place, but a system for detecting light output is necessary. Ideally, I would use a photopically-corrected photodiode to measure the on-axis intensity of each LED vs. current. Since the intensity distribution of each of the LEDs is roughly the same – Lambertian – the ratios of on-axis intensities of the LEDs is the same as the ratios of the LED fluxes. Sadly, finding a well-corrected photopic detector for a reasonable cost is not easy. Ultimately, I settled on an I^2C compatible board with a ROHM BH1750FVI 16-bit ambient light sensor. The spectral response, as shown in Figure 9, is not a perfect match with the photopic curve, but is better than most other sensors in that price range.

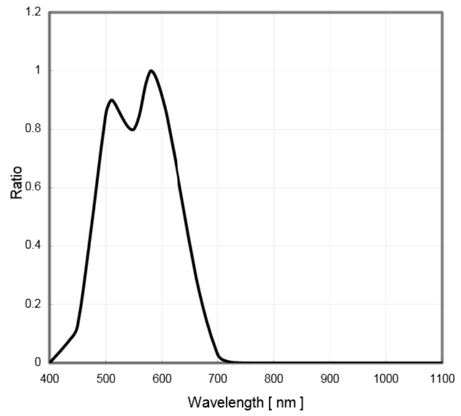

Figure 9. Spectral response curve for the ROHM BH1750FVI 16-bit ambient light sensor.

Finally, in order to generate the calibration tables, a small program was written that changes the input current for each of the LEDs and queries the on-axis intensity at the photodiode and writes the values to a file. This can be done for either the 8-bit DMX system or the 12-bit DAC configuration.

4.2.3 Software Controls

The software controls are the final piece of the system. It serves the following roles in the system:

1. It acts as the interface for the user, accepting the desired color and primaries.
2. It converts the color and primaries into the electrical currents necessary to create the color and sends the information to the hardware control system.
3. It provides the user with feedback about any color science that can be displayed on the interface.

In presentation mode, a laptop computer will host the software, but in kiosk mode, the platform upon which the software resides needs to be determined. Either the kiosk system needs to have a dedicated laptop or a small computer must be used to control the system via a keyboard, mouse, and monitor. The most notable system as of late is the Raspberry Pi system [19]. This is an ARM-based computer that runs a variety of operating systems, such as Linux. The Raspberry Pi has a can send signals to the Arduino board through a serial bus. Also, the Pi can be used to interface with the DMX controller through the USB ports.

The initial software interface was written in Visual Basic. VB was chosen due to the programmer's familiarity with it and the need for a quick solution for the presentation mode. For the kiosk mode configuration, a more portable solution was needed, so the QT toolkit [20] for C++ was used. By using QT, the same (or similar) program can be compiled for Microsoft Windows for the presentation mode as well as for Linux for the kiosk mode on the Raspberry Pi.

The software requires two types of inputs: the list of primaries and the desired color. Figure 10 shows a simple layout that was implemented. The buttons on the right enable/disable specific primaries. As primaries are added to the active list, the locations of the primaries are indicated on the diagram and the color gamut is shown by drawing lines between the primaries that make up the convex hull of the points. To compute the convex hull, the gift wrapping algorithm was applied. [21]

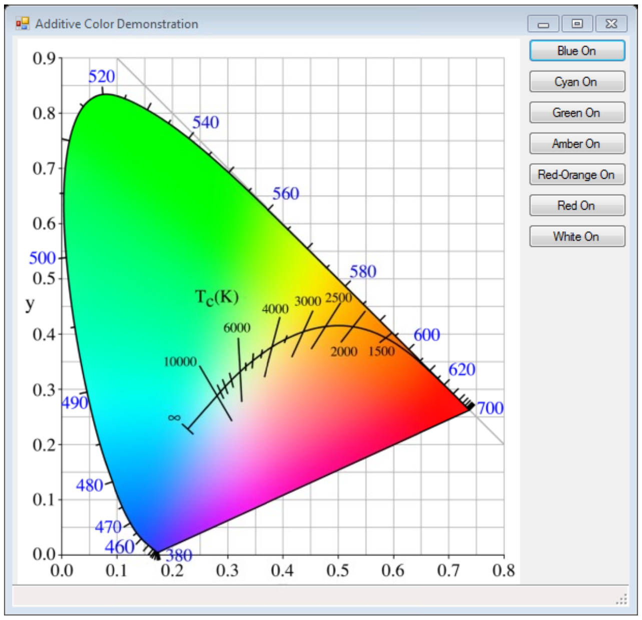

Figure 10. User-interface as implemented in Visual Basic.

The user clicks in the 1931 CIE Chromaticity diagram to specify the desired color. If the color is in the gamut of the current primaries, then the software calculates the necessary ratios of the current primaries, uses the calibration tables to determine the necessary current levels for the LEDs and sends that information along to the hardware. If the color is not in the current gamut, then the closest in-gamut color is computed and its calibrated current is determined and sent to the hardware. Figure 11 shows a workflow for the software calculation.

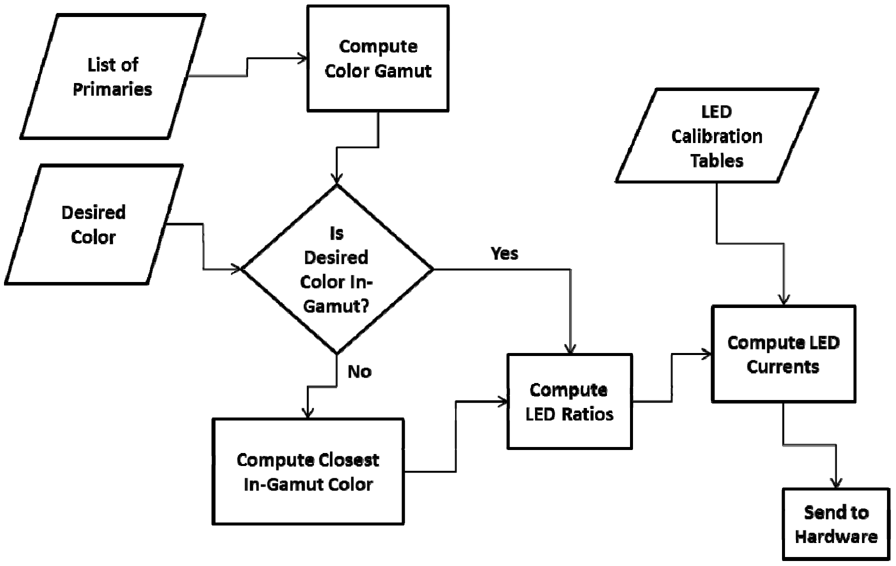

Figure 11. Workflow diagram of the software calculation.

With the exception of the white LED, the spectral power distribution (SPD) of each of the primaries is assumed to be a Gaussian distribution with the dominant wavelength and spectral half-width as given in Table 1. The tristimulus values for each LED are calculated from these SPDs. For the 5000K white LED, the SPD was assumed to be a 5000K blackbody for the purpose of calculating tristimulus.

5 CONCLUSIONS AND FUTURE WORK

An additive color demonstration kit has been created to serve the outreach efforts of the NES/OSA. The color science, the hardware, and the software have been described. The system can be used during a workshop presentation or as a kiosk-style demo on a tabletop with a touchscreen monitor. It has successfully been used at three color workshops given by the author and other members of the NES/OSA. Currently, the desired color and the output color do not match well; more work needs to achieve better correlation. Also, the level of color mixing achieved at the LEDs is less than desirable. An integrating sphere solution will be tested to see if this can better combine the colors.

REFERENCES

[1] Gregory, G. G., Biss, D. P., and Darnell, B., "Optical demonstrations through science fairs," Proc. SPIE 7783, Optics Education and Outreach, 77830K (2010).

[2] Wyszecki, G. and Stiles, W. S., [Color Science: Concepts and methods, quantitative data and formulae, Second Edition], John Wiley & Sons, New York (2000).

[3] Weisstein, E.W., "Least Squares Fitting--Polynomial." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html (23 August 2014).

[4] Lawson, C.L. and Hanson, R.J., [Solving Least-Squares Problems], Prentice-Hall, Chapter 23, p. 161 (1974).

[5] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, .B.P, "Section 10.5. Downhill Simplex Method in Multidimensions". [Numerical Recipes: The Art of Scientific Computing (3rd ed.)]. New York: Cambridge University Press (2007).

[6] "Rebel 7 LED Round Assemblies." https://www.luxeonstar.com/luxeon-rebel-7-led-round-assemblies (24 August 2014).

[7] "LUXEON Rebel Color Portfolio Datasheet DS68 20140527." Philips Lumileds Lighting Company (2014).

[8] "LUXEON Rebel ES Datasheet DS61 20140630." Philips Lumileds Lighting Company (2014).

[9] "Ledil OY C11670 Anna-50-7-S-OSL 7/3/2014." Ledil Oy (2014).

[10] "Luxdrive 3021/3023 BuckPuck Wide Range LED Power Module." LuxDrive by LEDdynamics. May, 2013 – Rev. 3.1. (2013).
[11] "Arduino Mega 2560." Arduino. http://arduino.cc/en/Main/ArduinoBoardMega2560 (24 August 2014).
[12] "Microchip MCP4725 12-bit Digital-to-Analog Converter with EEPROM Memory in SOT-23-6." Microchip DS22039D. 03/26/09 (2009).
[13] "OSEPP I^2C Expansion Shield." OSEPP. http://osepp.com/products/shield-arduino-compatible/i%C2%B2c-expansion-shield/ (24 August 2014).
[14] "MCP4725 Breakout Board – 12-Bit DAC w/I2C Inteface." Adafruit. http://www.adafruit.com/products/935 (2014).
[15] "4 Power FET Switches (Improved Version)." YourDuino.com. http://yourduino.com/sunshop2/index.php?l=product_detail&p=60 (24 August 2014).
[16] "LuxDrive 4016 QuadPuck 4-Channel DMX Driver Interface." LuxDrive by LEDdynamics. January, 2005 – Rev. 1.2. (2005).
[17] "DMX512." Wikipedia. http://en.wikipedia.org/wiki/DMX512 (24 August 2014).
[18] "USB Controlled DMX Interface." Velleman Inc. http://www.vellemanusa.com/products/view/?country=us&lang=enu&id=522065 (24 August 2014).
[19] "Raspberry Pi." Raspberry Pi. http://www.raspberrypi.org/ (24 August 2014).
[20] "QT." Digia. http://qt.digia.com/ (24 August 2014).
[21] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., "33.3: Finding the convex hull". [Introduction to Algorithms (2nd ed.)]. MIT Press and McGraw-Hill. pp. 955–956 (2001).