Electrical Design for Controlling Tunable 4-/6-Channel LED Arrays

INTRODUCTION

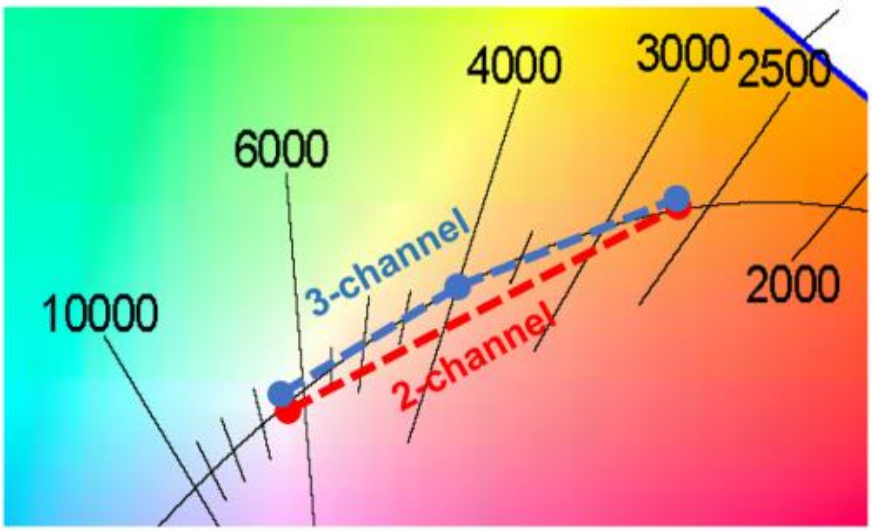
Color mixing several types of LEDs is a powerful tool to expand the capabilities of a lighting product. By combining several colors across the visible spectrum, many advanced features can be added to a lighting device to expand its capabilities beyond what a typical single-color device could achieve. Some of these features include:

- Adjustable color temperatures (CCT) for white light (black body color points), allowing users to choose warmer or cooler lighting.
- Non-BBL (black body line) colors for specialty applications, e.g. illuminating produce or other products in retail settings.
- Dynamic architectural or entertainment lighting with non-white colors.
- Optimization for specific light characteristics such as CRI, efficacy, circadian stimulus, TM-30, TLCI, or Standard Illuminant spectra.

The largest benefit of using color mixing in a lighting device is the flexibility it adds; many of these advanced features can be achieved simultaneously and with adjustability in the end product, so that optical characteristics of the light output can be adjusted in real time as required in the application.

CHANNEL COUNT

One of the most common color mixing applications is adjustable CCT along black body line color points. In some cases, this can be achieved with the simplest form of color mixing, which comprises two LED color points on the BBL of different CCTs. The biggest shortcoming of this type of design is that the mixed color falls below the BBL in the middle of the range, leading to red hue in the light output. This can be mitigated somewhat with the addition of a third color point in the center of the range, but the issue remains.



Cree LED / 4001 E. Hwy. 54, Suite 2000 / Durham, NC 27709 USA / +1.919.313.5330 / www.cree-led.com

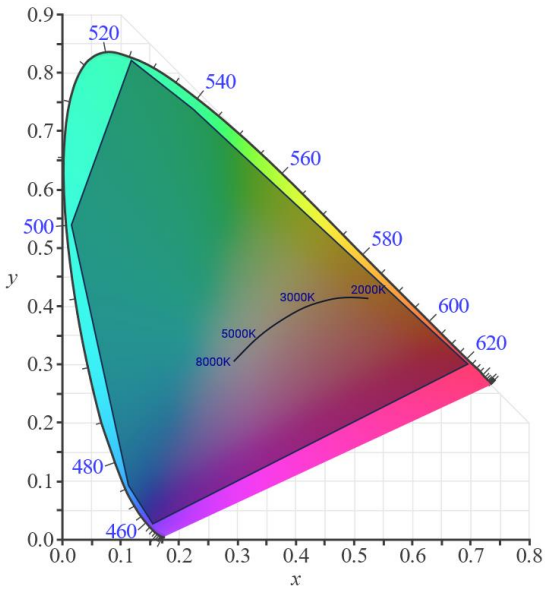
Regardless of channel count, this approach to color mixing is always limited to outputs on or near the black-body line.

Another common 3-channel mixing application is red/green/blue. This is commonly seen in entertainment lighting where saturated colors are the only output needed. While this approach is able to cover much of the color space, producing white light for general lighting is not typically possible due to extreme gaps in the spectrum that lead to low CRI and low efficacy. The long vector distances from the BBL color points also make tuning and adjustment very difficult and increase the requirements on channel resolution.

When an application requires white light outputs with excellent light characteristics, four channels is often the minimum needed. For specialized applications with extreme CRI, efficacy, or color space requirements, performance can be further improved by increasing to six channels.

CHANNEL MIXING

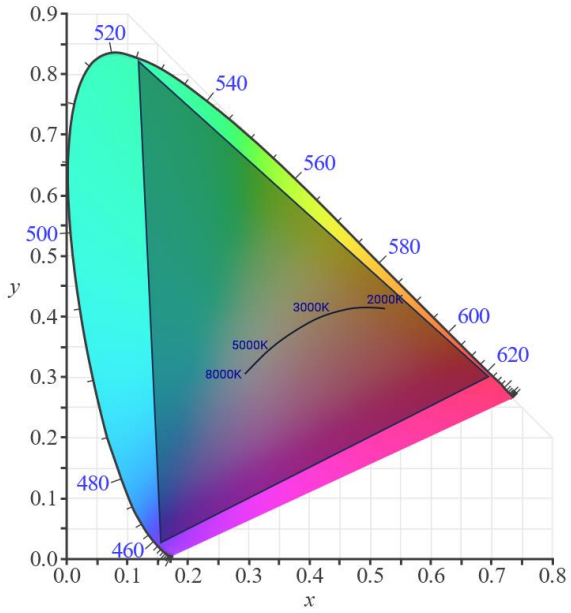
In this document the term “channels” refers to the individual LEDs or groups of LEDs that can be controlled independently. Typically, in a color-mixing design, each channel would consist of one or more LEDs of a single color, so the term “channel” in this document may also refer to the color point used in mixing. One of the color points used in a color-mixing application is sometimes referred to as an anchor point because it is a fixed position in the color space which can be used to “pull” the final output color in a certain direction. The achievable color space that the mixed output can create is found by plotting the area encompassed by the shape created by the anchor points, shown here in a 6-channel example that covers most of the space using royal blue, blue, cyan, green, lime, and red.



The arrangement of one color point per control channel is not always the case; in advanced color mixing applications a designer might mix multiple LED colors on a single control channel. This is typically done to create a virtual anchor point at a different point in the color space that is not otherwise achievable. It can also be done for CRI or efficacy reasons. This document will only focus on designs with one color point per channel.

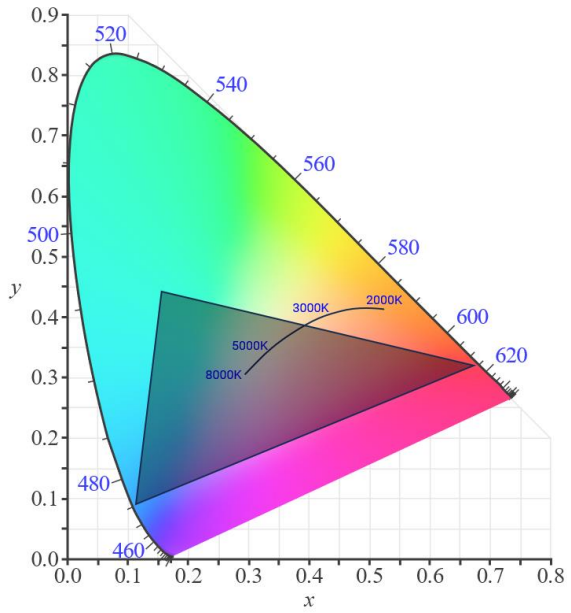
The selection of LED color points to use as the anchors is a complicated topic and has a lot of variability depending on the end goals of the design. A few examples of different color point selections and the reasoning behind them are shown below.

Red, green, royal blue



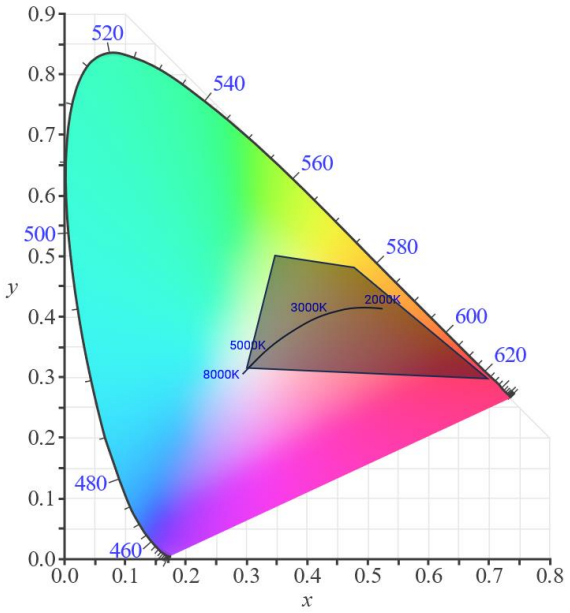
This arrangement might be chosen in an application where the goals are to maximize the available color space for entertainment lighting while minimizing the cost and channel count. Rendering white light with high CRI is not required. Efficacy is not a criterion.

Red-orange, blue, PC cyan, 4000 K white



This might be used in a CCT-tuning application where only a small portion of the color space is needed, covering a region of the black body line. By pulling the anchor points in closer to the target color point, efficacy generally improves. The choice of Phosphor-Converted (PC) cyan instead of green can help to improve CRI by pulling in a direction more opposite to the red-orange vector than green would. This allows both cyan and red–orange to be increased while not moving the color point significantly, allowing the R9 component of the CRI to come up and improve CRI Ra. R9 is typically the most critical section of CRI in LED color mixing and increasing the red amount tends to improve CRI the most.

Red, yellow, mint, 7000 K white



This might be chosen in a CCT-tuning application where very warm color temperatures are required with good efficacy. By choosing a cool white anchor point instead of blue, the lumen contribution from that channel increases significantly, bringing up efficacy, at the cost of losing some color space in the very cool or blue regions. Similarly, choosing mint over green as a counterpart to the red channel brings efficacy up in the warm white regions of the BBL that are critical. The choice of channel pairs that oppose each other, that is, they are on opposite sides from the desired output color, allows for very easy color control in the tuning phase. Adjusting 7000 K down and yellow up moves effectively in the “Northeast” direction or vice versa, and the same for the mint/red pair.

OFF-THE-SHELF SOLUTIONS

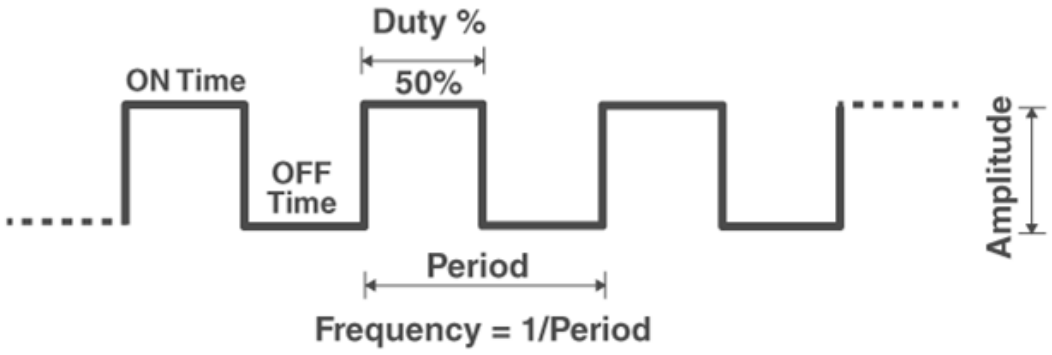
The primary barrier to creating high-channel-count color mixing applications is the availability of LED drivers. Four-channel LED drivers are available, but almost all of them are designed to operate red/green/blue/white LED strips, and the color mixing functions are usually limited to producing saturated RGB colors in fixed pre-programmed patterns geared towards entertainment lighting. Few off-the-shelf solutions offer the flexibility to use color points other than RGBW, or to mix them to produce variable-CCT white light. For this reason, a custom driver solution is often the best choice for high-performance color mixing applications.

LED DRIVE ELECTRONICS – AC TO DC

In typical single-channel LED drivers, the constant-current output to the LED is often tightly coupled to the AC-to-DC portion of the circuit. When creating multichannel drivers, it is more common to have a largely separate AC-to-DC section that provides a bulk DC supply to several copies of a DC-to-DC constant-current LED driver section. Only buck converters will be discussed here because this is the most typical in an AC mains-powered device. This document will focus on the DC-to-DC LED driver section and will assume that a bulk DC supply is available to power all channels. This can be accomplished with an off-the-shelf DC supply, which are widely available in common output voltages such as 12, 24, 36, or 48 V. It can also be done with an integrated AC-to-DC section designed into a custom driver, which is often done with a power factor correction (PFC) switching regulator IC.

CHANNEL ADJUSTMENT

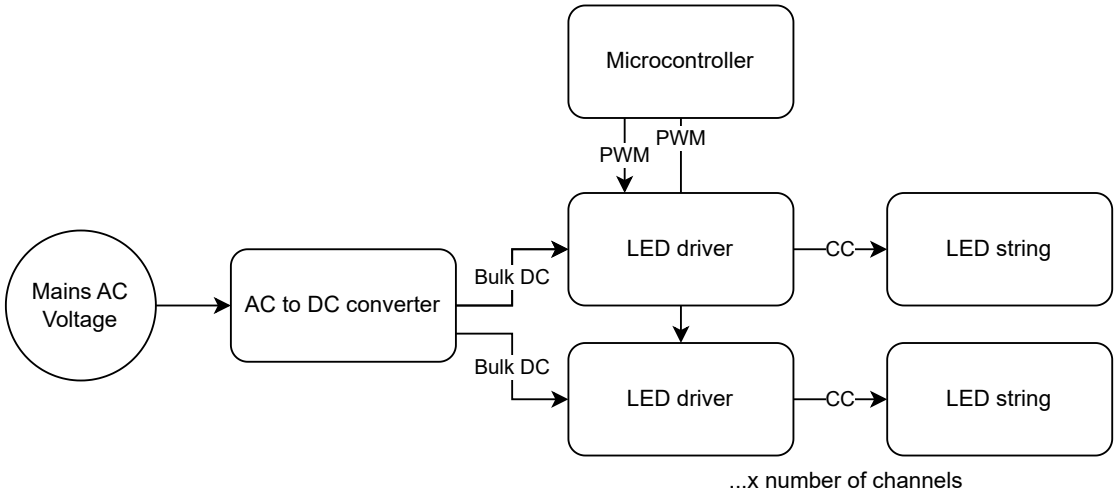
The most common technique for modulating a channel's output is pulse width modulation (PWM). This technique involves switching the output on and off at a frequency that is high enough to be imperceptible to the human eye. For example, an LED channel might be turned on at 100% for 100 microseconds, then turned off at 0% for 900 microseconds, and repeat, resulting in an average power of 10% of maximum at a frequency of 1 kHz. The end result is the perceived output is 10% of maximum and the light is steady without visible pulsing. The on/off ratio is referred to as the duty cycle. For lighting applications, 1 kHz would generally be considered the lowest allowable PWM frequency in order to avoid visible artifacts, and higher frequencies are preferable. In addition to the frequency, the other aspect of a PWM signal that is critical for lighting applications is the resolution. PWM resolution refers to the size of the number that defines the duty cycle; it is how many possible duty cycle settings there are. Resolution is often stated in terms of bits, describing the size of the binary value. For example, 8-bit resolution is 256 possible duty cycles, 10-bit is 1024, etc. In color mixing applications 10-bit is often the minimum usable resolution, and higher resolution is preferable. Other options for modulating a channel's output are possible, notably analog control which is possible with some driver designs.



Generally, a microcontroller is needed to generate the PWM signal. Other options including FPGA or ASIC could perform this task as well, but a microcontroller gives more flexibility in this role, particularly during development. This document will focus on microcontroller designs.

LED DRIVE ELECTRONICS – CC DRIVER

High-power LED driver designs are centered around a constant-current power supply. In color mixing applications, the driver has an added requirement that its output be adjustable with high resolution in order to hit precise mixing ratios among channels. This final building block gives an overview of the hardware needed for the multi-channel driver.



In some applications, a dedicated LED driver IC is available to create a suitable LED driver. In other cases, a constant-current output IC is not available, and the more common constant-voltage type must be adapted to form an LED driver. In either case, some basic criteria are used to select an IC:

- Integration of components – does the IC incorporate switching MOSFETs and/or Schottky diodes, or are those separate? If incorporated, the design can be more compact and cheaper, but this comes at the cost of less flexibility on the output current and may become a limitation.
- Switching frequency – how fast is the IC's switching frequency in its buck conversion? Faster frequencies allow for smaller inductors but may have a tradeoff with higher switching losses. Color mixing adds an extra consideration. If PWM will be used to adjust channel outputs, the driver's switching frequency should be considerably higher than the PWM frequency.
- Output current – the IC and supporting circuitry must be rated to provide the current necessary to achieve the design's desired light output.
- Maximum voltage – the IC and supporting circuitry must be rated to handle the input voltage needed to drive the LED strings.

If a design is running into limitations on current or voltage, these can sometimes be mitigated by adjusting the LED string arrangement. For example, if the string voltage is too high but there is remaining output current headroom, the LEDs can be rearranged into more parallel strings to reduce voltage while increasing current.

In the case of a dedicated LED driver IC, an added critical requirement is a dimming input to adjust the LED current.

- What type of dimming signal can the input accept? Will it work with PWM?
- Are there speed limitations on the PWM signal that will be problematic?

CC DRIVER EXAMPLE 1 – DEDICATED IC

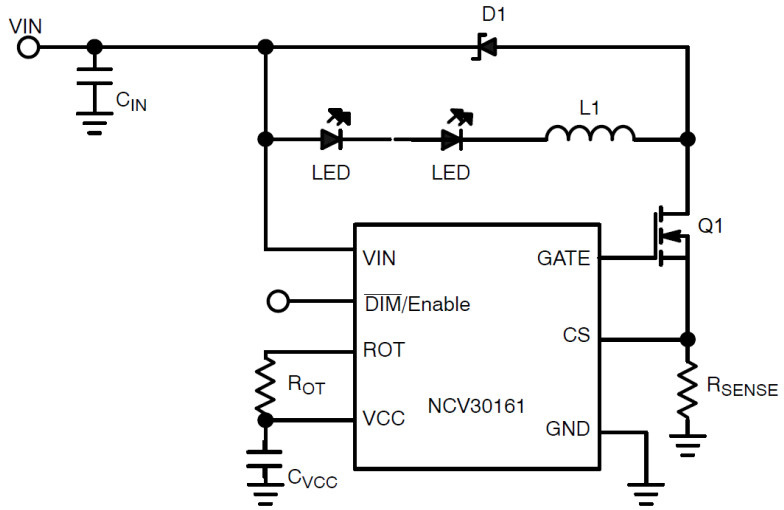
In one design example a driver is needed to power four channels of XLamp® XE-G LEDs at a peak current of 2 A. The design will perform color mixing to produce white light output with a user-adjustable CCT that ranges from 2700 K to 6500 K. The four color points used are red, green, royal blue, and PC yellow. Each string consists of 8 LEDs in series for a peak voltage of 26.4 V. A bulk DC supply of 36 V is chosen to be above the string voltage, be a standard value, and remain low enough to meet safety requirements as well as to allow a wide selection of LED driver ICs. The design specifications are as follows:

LED current	2 A
CCT range	2700 K – 6500 K
String count	4
Series LEDs per string	8

The IC search criteria are:

- Greater than 2 A output current.
- Greater than 36 V input voltage.
- Dimming input that allows >1 kHz frequency.

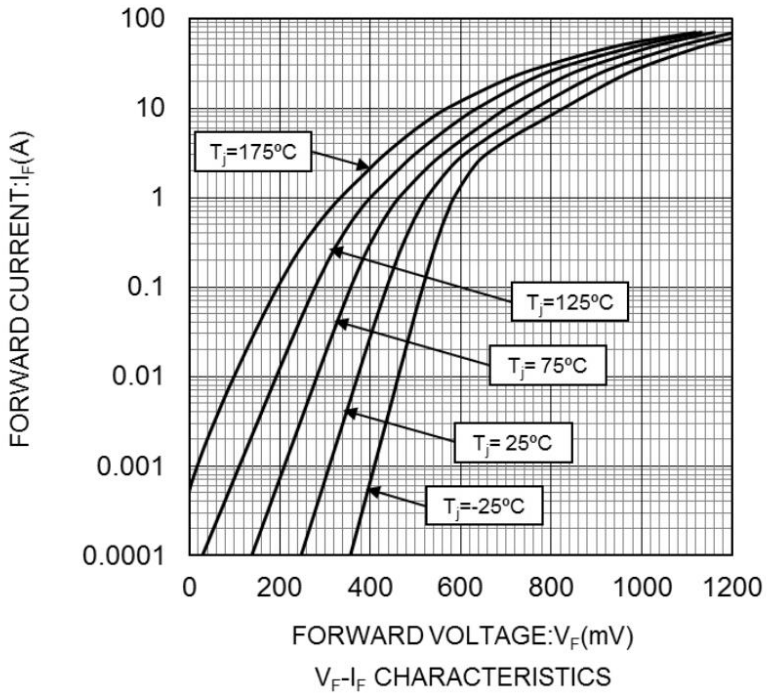
An IC that meets these requirements is the NCV30161 made by ON Semiconductor. From the IC's data sheet, it supports input voltages up to 40 V, PWM dimming up to 20 kHz, and has all power components external, which generally means the output current is only limited by the selection of the other parts in the power path. This IC presents an additional advantage that it allows a common anode connection for each LED string, which simplifies PCB layout and wiring. The basic LED driver schematic for each channel is taken directly from the example in the data sheet:



Since the IC's current sense voltage is 200 mV, R_{SENSE} will be set to 0.1 Ω to get a 2 A output current (R = V/I = 0.2 V/2 A = 0.1 Ω). Other critical components to create the driver include the inductor L1, the freewheel diode D1, and the switching MOSFET Q1.

FREEWHEEL DIODE

The freewheel diode is often the simplest to choose so it is an easy starting point. The only requirements are that it can handle the output current and the supply voltage in reverse bias. Both specifications should have some safety margin added. For this design the values are 2 A forward current and 36 V reverse bias. In addition to that, a lower forward voltage will lead to a more efficient system, so Vf should be minimized. For this reason, Schottky diodes are almost always the best choice. A diode that meets the requirements is the RB088LAM-60 made by Rohm Semiconductor. This diode has a reverse bias rating of 60 V and a forward current rating of 5 A, giving a good margin on both specifications. The data sheet shows that a Vf of around 0.5 V can be expected at 2 A.



V_F-I_F CHARACTERISTICS

MOSFET

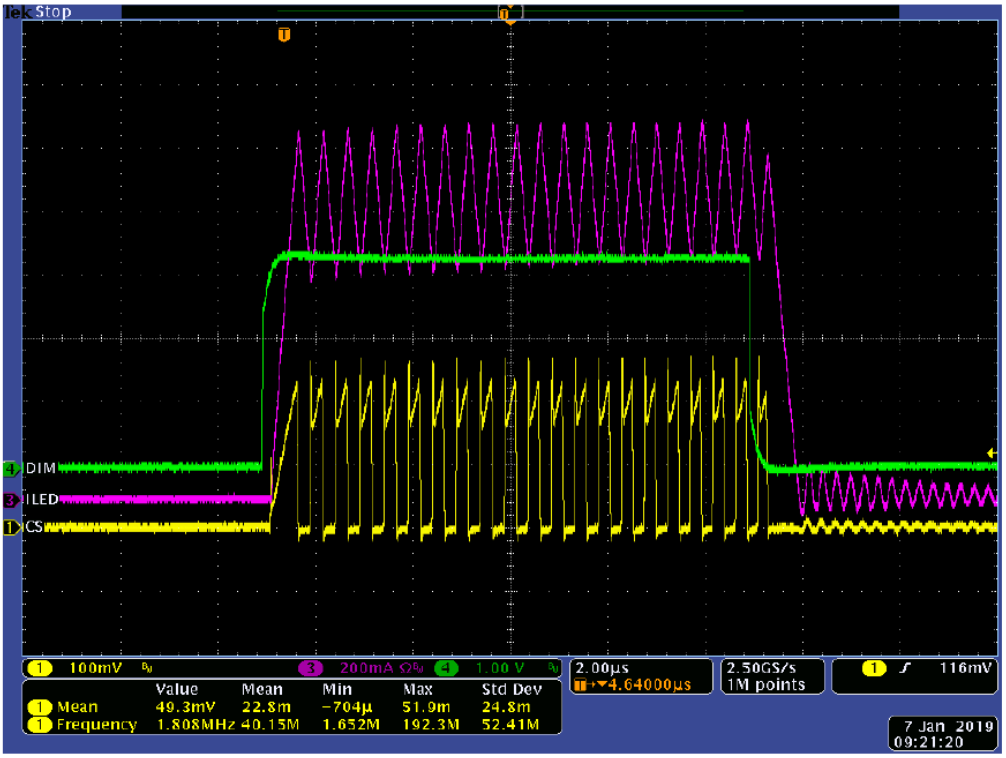
The MOSFET has similar requirements to the diode. Ratings must handle the output current on the continuous drain current rating and supply voltage on the Vds rating (plus margin). In addition to this, the Rds_on resistance should be minimized for efficiency. The driver IC's data sheet indicates that it uses the internal 5-V supply to drive the MOSFET gate, so there are no challenging requirements on the Vgs voltage rating. A MOSFET that meets these requirements is the SQ2318BES-T1_GE3 made by Vishay. This part has a Vds rating of 40 V which is sufficient for the 36 V supply, a continuous drain current of 8 A, and an expected Rds_on of 32 mΩ at the expected gate voltage.

INDUCTOR

The IC's data sheet gives guidance on how to choose the inductor value. The inductance can be a wide range of values, but the value will directly determine the switching frequency. The frequency chosen is a tradeoff. Larger inductances will lead to slower switching, which

is often more efficient, but this requires physically larger parts and if it goes too low, then PWM dimming will not work well. A smaller inductance brings the frequency up, but at the cost of lower efficiency. The IC can only switch up to 2.4 MHz, so the frequency must stay below this value. If the data sheet gives any example values, these are often the best starting point. The data sheet shows test results using a 3.3 µH inductor which results in LED ripple current around 30% and switching frequency of about 2 MHz:

12 Vin, 3.3 µH, 2 LEDs, 200 mΩ R_{SENSE}, 1 Khz F_{DIM}
Purple: LED Current, Yellow: CS Pin, Green: DIM Pin



Since the design will operate with a larger voltage difference from input to output than the test shown, and because there is plenty of room to slow down the switching frequency, a larger inductance is chosen. It will be easier to source components with common values, so values of 4.7 µH or 10 µH are good choices to raise the value. The predicted switching frequency can be calculated from the equations in the data sheet.

$$t_{ON} = \frac{L \times \Delta I}{VIN - V_{LED} - I_{OUT} \times \left(FET_{R_{DS}}(on) + DCR_L + R_{SENSE} \right)}$$

$$t_{OFF} = \frac{L \times \Delta I}{V_{LED} + V_{diode} + I_{OUT} \times DCR_L}$$

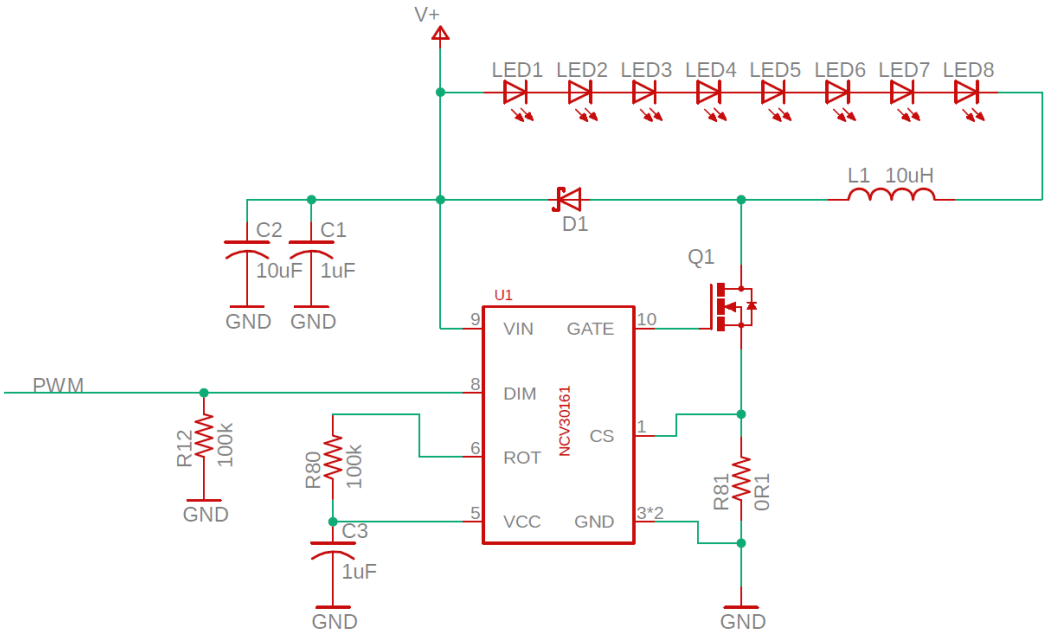
$$f_{SW} = \frac{1}{t_{ON} + t_{OFF}}$$

These equations require several prerequisites from other parts of the design, as well as characteristics of the inductor itself. For this reason, it is often necessary to choose a specific inductor as a starting point and check what the result will be. Based on a 10uH inductance and a minimum 2-A DC current rating, the SRP6030VA-100M made by Bourns is chosen to start as it has a 5-A current rating and a fairly low 62-mΩ DC resistance.

The equations also require that ΔI is known, which will be the ripple current seen by the LEDs. From the IC's datasheet, the IC sets the rising and falling thresholds at 180 mV and 220 mV, corresponding to a maximum and minimum output current of 1.8 A and 2.2 A (I = V/R). Given that the LEDs are rated at 3 A, the 2.2-A peak current is well within the specifications. All of the required values are now known.

L (inductance)	10 µH
ΔI (ripple current)	0.4 A
Vin (input voltage)	36 V
Vled (LED string forward voltage)	26.4 V
Iout (LED current)	2 A
Rds_on (MOSFET on resistance)	0.032 Ω
DCR_L (inductor DC resistance)	0.062 Ω
Rsense (sense resistor)	0.1 Ω
Vdiode (diode forward voltage)	0.5 V

With these values, the predicted t_{ON} is 0.43 microseconds, t_{OFF} is 0.15 microseconds, and the switching frequency is 1.7 MHz. These values fit well within the operating region of the IC, so the design can proceed with the chosen components. The schematic is taken directly from the IC data sheet and is built with the selected components. R12 is added to keep the dimming pin low during startup until the microcontroller has configured the PWM output; this prevents flashes upon powerup.



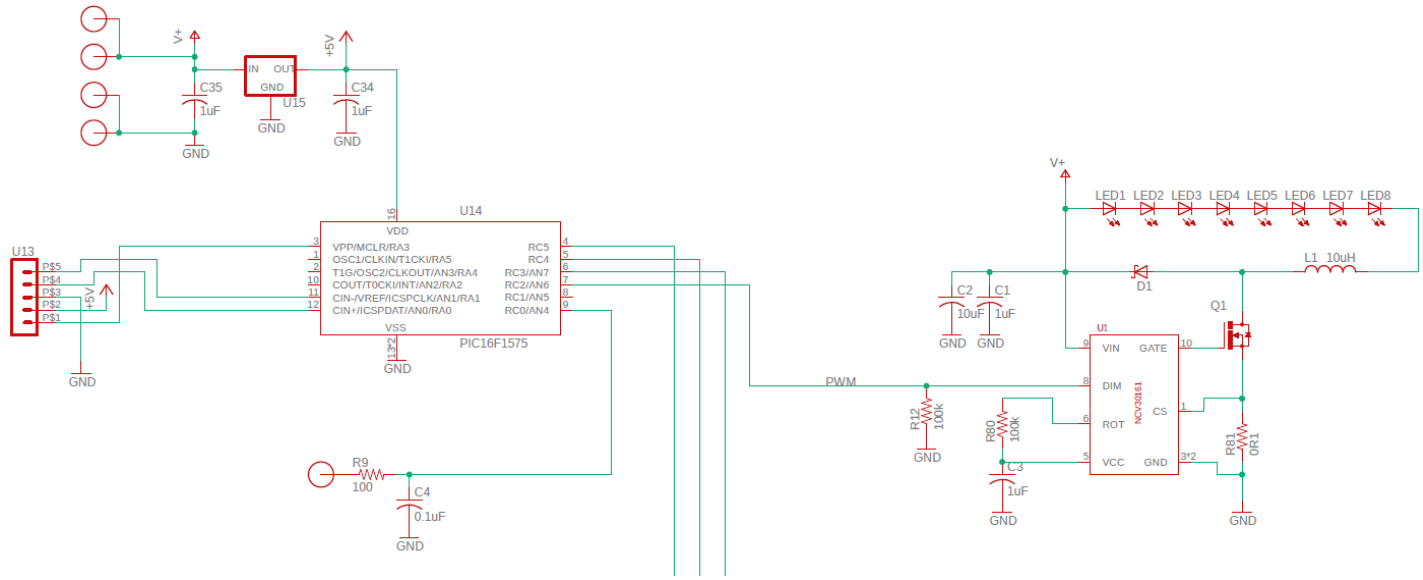
MICROCONTROLLER

For the design's LED drive functions, the only requirements on the microcontroller are PWM peripherals of enough quantity and the speed and memory to run the output calculations. In general, the computing demands for color mixing are quite low compared to the capabilities of modern microcontrollers, so not much effort needs to go into this second requirement. After the PWM hardware is established, most of the decisions around microcontroller selection are based on the other requirements of the design, such as the following.

- What user inputs are needed? Does the microcontroller need wireless capability for user interface? Does it need analog inputs to interpret 0-10 V dimmer signals? Does it need zero-cross hardware to interpret TRIAC dimming signals?
- Does it need low-power modes for standby power consumption goals? Are there limitations on the supply voltage that can be provided for it?
- What other tasks does it need to perform that may add requirements? Temperature monitoring? Non-volatile memory for user-configurable features?

In this design example, the user interface will be a knob mounted directly on the luminaire, so the only external interface hardware needed is an analog input (ADC) to measure a voltage from a potentiometer. With the only requirements being four PWM channels and an ADC, the PIC16F1575 made by Microchip can work in this design. This microcontroller is primarily chosen for its extremely high-resolution PWM peripheral which can go as high as 16-bit resolution. To provide it with a 5-V supply, the MCP1792 voltage regulator is chosen for its ability to handle the 36-V input voltage.

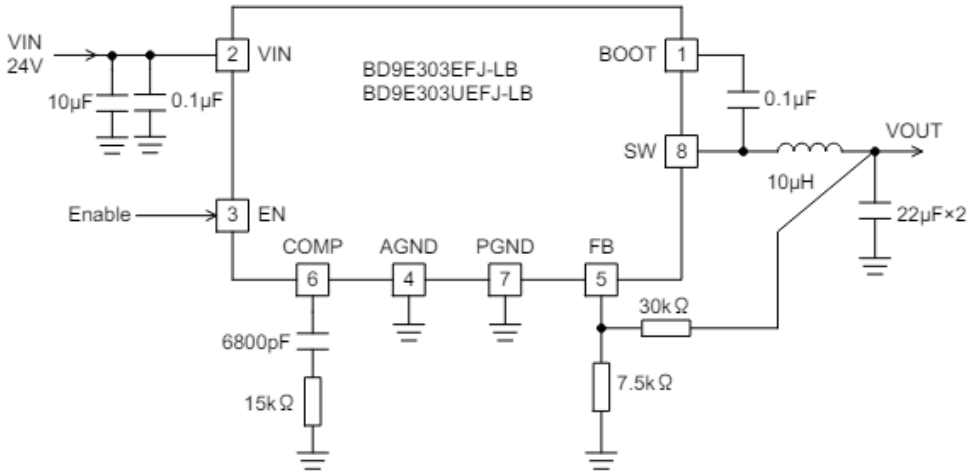
The final schematic includes four copies of the LED driver, the microcontroller and support components, and wire pads to connect inputs and outputs (three copies of the driver circuit not shown).



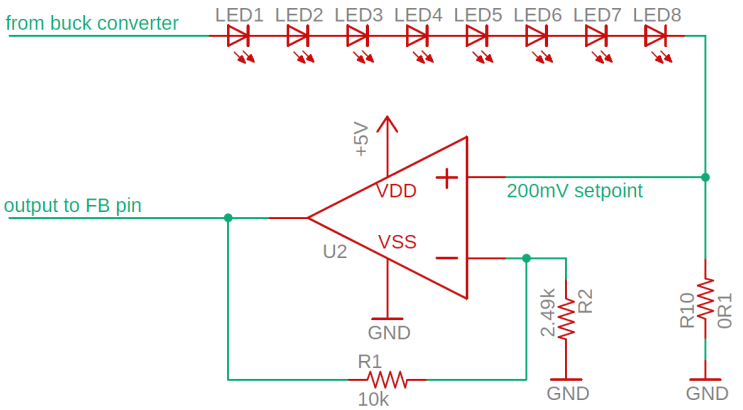
From this schematic a PCB layout can be created. The layout is largely determined by the physical and mechanical requirements of the design and so is beyond this document's scope.

CC DRIVER EXAMPLE 2 – CONSTANT VOLTAGE IC

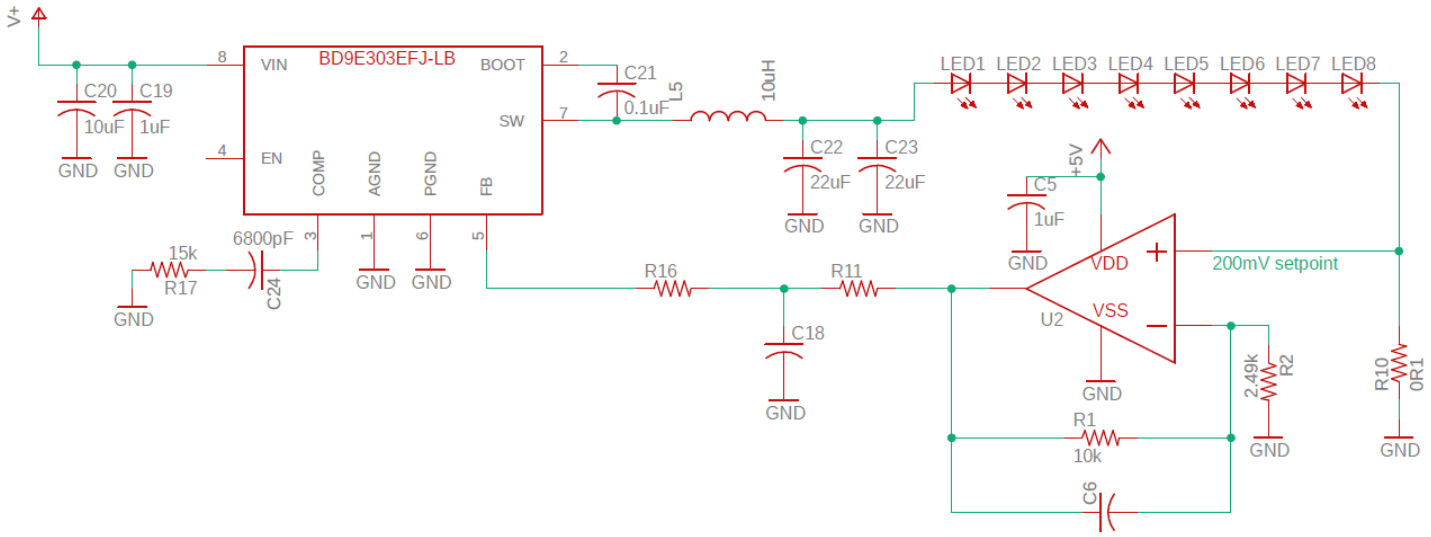
In some cases, a dedicated LED driver IC cannot be found that meets the design's requirements. To get around this, sometimes the more common general-purpose constant-voltage buck converter can be adapted to form a constant-current driver. This is done by adding circuitry to measure the LED current and feed a corresponding signal into the IC's feedback pin, instead of the normal use case of connecting it to a voltage divider measuring the output voltage. For this example, the same string configuration, output current, etc. as the previous example will be used. An example buck converter IC that can accomplish the goals of a 36-V input and a 2-A output is the ROHM Semiconductor BD9E303EFJ-LB. This IC comes with the advantage of integrated MOSFETs and diodes which can reduce system cost and size. From the data sheet, the standard application to produce a constant voltage output is below. The only critical component that may need significant effort to select is the inductor, which would be done in a process very similar to example 1.



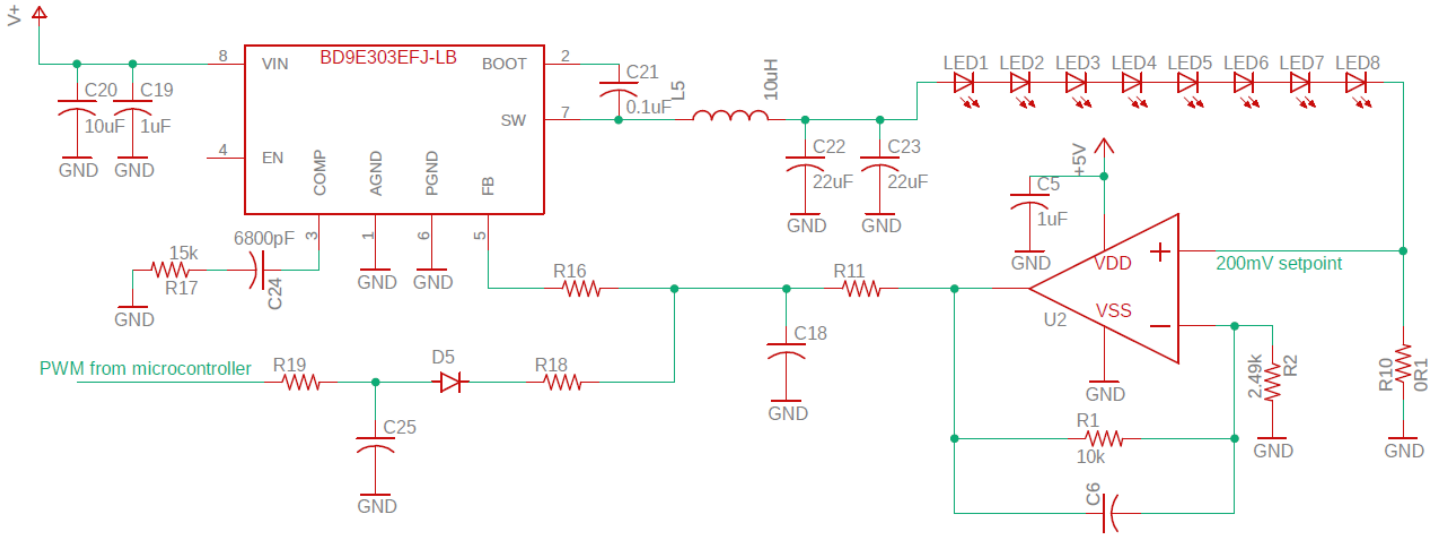
The IC regulates the output by maintaining the feedback pin (FB) at 1.0 V, and this voltage is set up to be ratiometric to the output voltage via a voltage divider. To convert this design to constant current, a signal needs to be created that is ratiometric to the LED current, and that outputs 1.0 V at the 2.0 A-current setpoint. This can be accomplished with an op amp set up in a non-inverting amplifier circuit and using a current-sensing resistor to measure the LED current.



In this circuit the 2.0-A LED current creates a 200-mV signal at the op amp's non-inverting input (V = I * R = 2 A * 0.1 Ω = 0.2 V). The voltage divider formed by R1 and R2 cause the voltage at the op amp's output to be 5 times higher than at its non-inverting input (gain = (R1 + R2) / R2 = (10 kΩ + 2.49 kΩ) / 2.49 kΩ = 5.0), meaning that the output will reach 1.0 V at the 2-A current setpoint. This signal is fed into the FB pin of the buck converter to create a constant-current LED driver. In practice, several filtering elements are often needed to tune this circuit into stable operation, and their values need to be adjusted for the specific design.



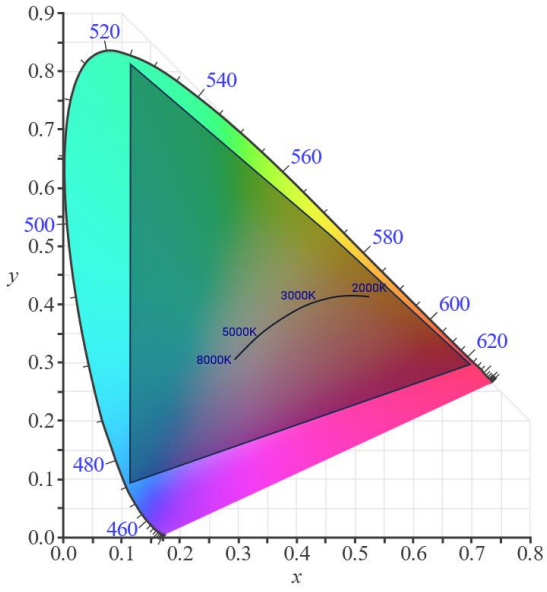
The IC has been successfully converted into a constant current driver to power the LED strings, but it is at a fixed 2.0 A output. To control the output from the microcontroller, the signal at the FB pin can be further manipulated by injecting voltage from the PWM signal. By raising the FB voltage and pushing it up to the 1.0 V setpoint artificially, the output is suppressed, and the current is reduced. First the PWM signal needs to be filtered to analog via an RC filter, and then it is fed into the signal path from the op amp via another resistor. To avoid pulling down on the FB signal when the PWM is low, a diode is added in the path as well. As these are sensitive analog signals, all values need to be tuned for the specific design, so values are not provided for many of the components in this signal path.



The constant-voltage IC is now a constant-current LED driver with PWM-adjustable output. It should be noted that the PWM signal is now inverted logic, the lower the duty cycle, the higher the LED current. Another thing of note about this design is that, when well-tuned, the LED output is a continuous current, not pulsed as in example 1. This typically leads to higher efficacy from the LEDs. With this type of design, it may be necessary to also control the enable input of the IC in case the analog circuit is not able to fully turn off the LED string. There is very little bandwidth requirement for this application so nearly any single-supply op amp with appropriate voltage ratings can be used. With the LED driver block complete, all other aspects of the design are the same as shown in example 1.

TUNING

To establish how to control the channels in the design, many data points are collected during the development phase to find channel currents that accomplish the appropriate light output at several locations along the CCT adjustment range. This is often done with common CCT targets (6500 K, 5700 K, 4000 K, 3000 K, etc.) but does not have to be. The only points that are critical are the extreme ends of the adjustment range. The number of points between the endpoints that are used is a tradeoff; more points take more time to tune and more memory to store but increases the accuracy of the output. For this design, the CCT range is 2700 K to 6500 K, and the achievable color space formed by the selection of red, green, blue, and PC yellow looks like the following.



Although the choice of PC yellow does not significantly expand the available color space, it helps considerably to fill in the spectrum when creating white light which improves CRI and efficacy.

The tuning points are chosen to be the common CCT values in the tunable range: 6500 K, 5700 K, 5000 K, 4000 K, 3500 K, 3000 K, and 2700 K. To find the tuning values, a mock-up of the LED board is built and tested in an integrating sphere. While measuring the combined light output from the LEDs, the drive currents to each channel are manually adjusted to tune the light output to the desired target. There are multiple criteria to consider while tuning – the resulting color point needs to be accurate, the lumen output needs to hit design goals as well as be consistent with the other color points (or as consistent as possible), the drive current to any channel cannot exceed the

2A design current, and potentially other parameters such as CRI need to hit goal values. This is an iterative process that can be time consuming and can be automated in some cases, but performing the task manually is not difficult with some practice. While performing the tuning step it helps to keep the locations of the anchor points in mind in the color space and picture those points "pulling" the output towards them when increasing current to that channel.

The tuning process can either be done with benchtop power supplies (one for each channel), or it can be done with a prototype of the driver circuit hardware. The latter has the advantage that the values will be more accurate when transitioned to the final design, but comes with the disadvantage that it requires building additional firmware, circuitry, and potentially software to allow the operator to send inputs to the microcontroller.

If using benchtop power supplies, the data to be collected at each tuning point are the currents to each channel:

CCT	Red String	Green String	Blue String	Yellow String
6500 K	0.201 A	1.046 A	0.422 A	1.259 A
5700 K	0.242 A	0.938 A	0.379 A	1.333 A
5000 K	0.231 A	0.837 A	0.328 A	1.457 A
4000 K	0.207 A	0.686 A	0.291 A	1.632 A
3500 K	0.256 A	0.657 A	0.237 A	1.662 A
3000 K	0.446 A	0.637 A	0.18 A	1.584 A
2700 K	0.673 A	0.6 A	0.123 A	1.521 A

These are then converted to the theoretical PWM values that will provide the same currents in the final design. If the points have been tuned using the actual hardware, then the PWM values are already known and no calculation is needed. In this design the PWM resolution is chosen to be 10 bits or 1024 values and the driver current is 2 A, so each tuning value can be converted to a PWM duty cycle by channel current / driver current * resolution -1 = PWM duty, so the calculation is current/2 A * 1024 – 1 = PWM duty. The subtraction of 1 is to change the value to the binary scale used in the source code where the range of values will be 0 to 1023. The result is shown below.

CCT	Red String	Green String	Blue String	Yellow String
6500 K	102	535	215	644
5700 K	123	479	193	681
5000 K	117	428	167	745
4000 K	105	350	148	835
3500 K	130	335	120	850
3000 K	227	325	91	810
2700 K	344	306	62	778

Note that these values are from a real R/G/B/PCY design and are the actual tuning points, but all designs have tuning affected by optics, LED bins, driver hardware, etc. and tuning should always be redone for a new design.

FIRMWARE

The primary function of the firmware in the design is to adjust the four channel output levels in response to the user input to produce a variable CCT, using the color points found in the tuning stage. There are many ways to accomplish the calculation from CCT setting to PWM duty cycles, three common ones are as follows.

- Polynomials
  - ◊ For each channel, the tuning points are converted into a polynomial that describes the channel output as a function of the desired CCT input. The firmware then calculates the polynomial for each channel when given a new CCT setpoint. This method has the advantage that it takes very little memory space, but the disadvantage is that the calculation can be slow and inaccurate on low-cost microcontrollers.
- Lookup tables
  - ◊ For each channel, the tuning points are converted into a list of output values, one for each possible input of the CCT setpoint. The firmware then just looks up the output value when given a new CCT setpoint. This has the advantage of being extremely fast since no calculations need to be made, but the disadvantage is that the tables take up very large amounts of program memory, especially if the resolution exceeds the hardware's native register size (e.g. 10-bit values on an 8-bit CPU). A hybrid of polynomial and lookup tables can also be created where the firmware does the slow calculation of polynomials upon startup and then saves them in RAM for fast lookup later.
- Linear interpolation
  - ◊ For each channel, the tuning points at all CCTs are known. When a CCT between tuning points is selected (e.g. 4200 K in this design), the firmware interpolates between the adjacent two points to find the needed PWM value along the virtual line between tuning points. This approach has low memory requirements (only the table of points above) but can usually be calculated faster than a polynomial of 2nd or higher order. If enough tuning points are used, it can be very accurate. The other advantage to linear interpolation is the ease of updates during development; all that's needed to adapt to a new LED color or CCT target is to update the table of values.

For this design, linear interpolation is selected as the best option. The interpolation calculation involves finding the two surrounding points of the desired CCT, calculating the span between them (ΔCCT), finding the corresponding PWM values and their span (ΔPWM), and finally doing a multiplication and division step to interpolate between the points. For example, calculating the green string at 4200 K:

- Base value is 350 (4000 K value)
- Upper value is 428 (5000 K value)
- ΔCCT is 5000 – 4000 = 1000
- ΔPWM is 428 – 350 = 78
- CCT setpoint offset is 4200 K (input) – 4000 K (surrounding lower point) = 200
- Output value is setpoint offset / ΔCCT * ΔPWM + base PWM = 200 / 1000 * 78 + 350 = 366 (rounded to whole number)

When implementing the math on a low-cost microcontroller, it is usually best to avoid using float variables to speed up the calculation. This means that all calculations need to be done with integer values only and any fractional remainders are dropped. To perform this calculation in practice with this limitation, the division and multiplication must be done in reverse order (though this makes the equation less intuitive) to avoid the loss of resolution in the output. The calculation then becomes setpoint offset * ΔPWM / ΔCCT + base PWM =

output PWM. Without rearranging, the first step of 200 / 1000 would result in a zero result as the remainder of 0.2 is dropped by integer math. In addition, it is good practice to split up the calculation into several lines of code to force the compiler to perform the calculations in a specific order. This prevents any compiler optimizations from rearranging it and potentially re-introducing the error.

In this particular implementation, the CCT selection by the user is input to the firmware via the ADC, which will produce a 10-bit value corresponding to the desired CCT. To reduce the calculations the firmware needs to perform, the CCT information should be completely left out, so that instead of calculating ADC -> CCT -> PWM it can simply be ADC -> PWM via the interpolation function. To do this the ADC range must be mapped to the CCT range, so the table becomes the following.

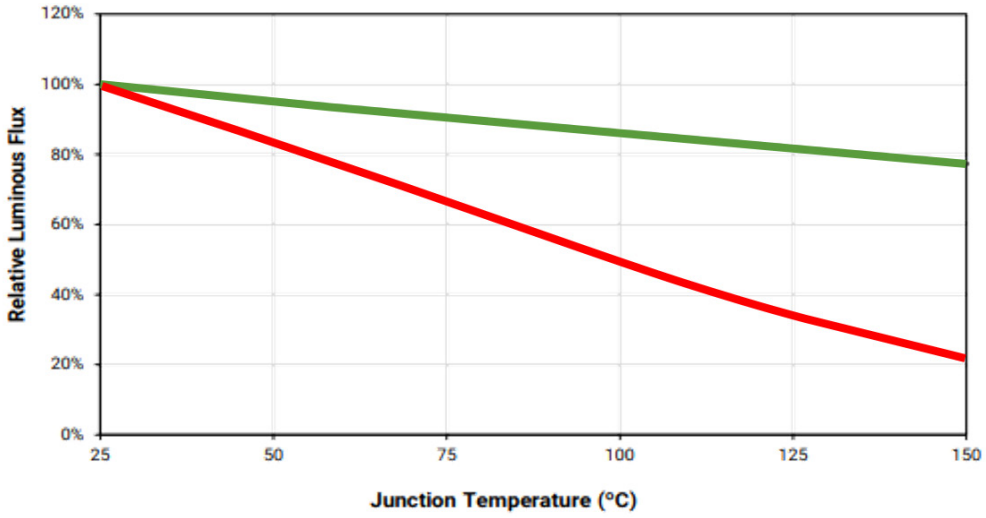
ADC Input	Red String	Green String	Blue String	Yellow String
1023	102	535	215	644
807	123	479	193	681
619	117	428	167	745
349	105	350	148	835
215	130	335	120	850
80	227	325	91	810
0	344	306	62	778

The new table has the CCT mapped linearly to the input, so that a change in input creates a proportional change in the CCT Kelvin value. Other mappings might be desired (e.g. extending the warmer CCTs to take up more of the adjustment range). If this is the case, it needs to be planned in advance since the example interpolation function shown below is expecting the tuning points to be equally spaced. Below is an example implementation of the interpolation function from ADC to four output channels written in C.

```
void linear_interpolate(unsigned int color)
{
  #define RESOLUTION 1024 //the PWM resolution
  #define ADC_RESOLUTION 1024 //the ADC resolution
  #define NUM_POINTS 7 //the number of tuning points
  #define SEGMENT_SIZE ADC_RESOLUTION/(NUM_POINTS-1) //the number of ADC counts between tuning points
  //input the tuning point values for each channel:
  static const unsigned int ch1_points[NUM_POINTS]=344, 227, 130, 105, 117, 123, 102;
  static const unsigned int ch2_points[NUM_POINTS]=306, 325, 335, 350, 428, 479, 535;
  static const unsigned int ch3_points[NUM_POINTS]=62, 91, 120, 148, 167, 193, 215;
  static const unsigned int ch4_points[NUM_POINTS]=778, 810, 850, 835, 745, 681, 644;

  unsigned int index;
  signed long outval; //make this a larger variable since it may multiply large numbers during the calculation
  signed int low_value, delta_value;

  for(int i=0; i<(NUM_POINTS-1); i++){
    if(color>=SEGMENT_SIZE*i && color<=SEGMENT_SIZE*(i+1)){ //find the tuning point below the desired setpoint
      index=i;
      break;
    }
  }

  low_value = ch1_points[index]; //find the PWM value associated the the lower tuning point
  delta_value = ch1_points[index+1] - low_value; //find the delta PWM between the two surrounding tuning points
  outval = color - SEGMENT_SIZE*index; //find the input offset
```

TEMPERATURE EFFECTS

If the design is expected to experience significant variability in the temperature of the LEDs, the effects on color tuning need to be considered. The biggest contributor to temperature-induced color inaccuracy comes from red, red-orange, or amber LEDs. This is because their luminous flux falls off with temperature at a much greater rate than other colors, as can be seen comparing XE-G red and green LEDs.



There are several ways to mitigate this, the simplest of which is to switch to PC red or amber parts, which will experience less thermal droop than the monochromatic counterparts. Second, a prototype of the design should be evaluated to find what the LED temperature will be in use, and to have the LEDs at that temperature during the tuning process. This usually works well for indoor designs where the room temperature is fairly consistent. For more demanding applications, sometimes active temperature compensation is required. This involves adding temperature sensing to the LED board and monitoring it with the microcontroller that is performing the PWM calculations. Then the faster-drooping channels can compensate for their temperature by increasing the output to counteract the thermal droop. This technique adds time and complexity to the tuning process but can produce excellent results if done well.

EXPANDING TO SIX CHANNELS

Some color mixing applications have more demanding requirements on CRI, the profile of the spectrum, or on the achievable color space. In these cases, it may be beneficial to expand to a six-channel solution. Designing a 6-channel driver is nearly the same as for a 4-channel; the constant current driver design, the tuning process, and the firmware are all done the same, just repeated for two additional channels. The exception to this is microcontroller selection. Stepping up from a requirement of four PWM channels to six significantly limits the options. Still, there are microcontrollers available with six or more PWM channels, such as:

- STMicroelectronics STM32C011F4
- Microchip PIC16F1778 (using 4 from the PWM peripheral and 2 from the CCP peripheral)
- Microchip ATmega1609 (although only 8-bit resolution which may be insufficient, especially if dimming is needed)
- Espressif ESP32-H2

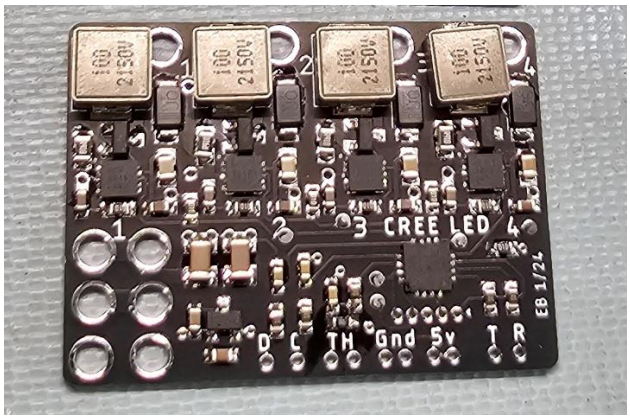
If no suitable microcontroller with sufficient PWM channels can be found, there are other options that work well.

- Using a PWM expander such as the NXP PCA9685. A device like this receives control commands from a microcontroller via I2C or similar, and independently operates many PWM channels based on those commands.
- Use multiple microcontrollers. In some cases, it can be beneficial to divide up tasks into two or more smaller microcontrollers which have a method of communicating between them. Two devices with only three or four PWM channels each can be used to create the six needed channels.
- Use software PWM. This technique involves writing firmware to control general purpose I/O pins to form extra PWM signals when there are no peripheral resources left. If the microcontroller is operating fast enough it can work, but care must be taken to ensure that no other processes can interrupt the PWM functions and cause them to glitch which might create visible flickers in the light output.

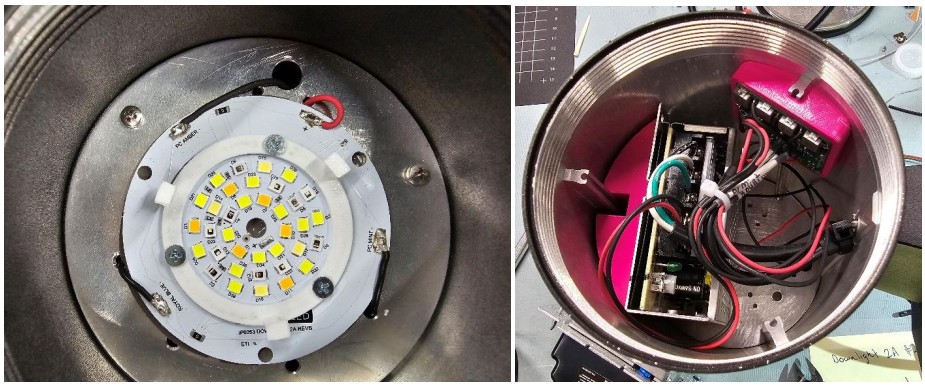
It is also possible to multiplex PWM hardware, but this is generally not recommended for color mixing as it degrades the resolution and creates limits on maximum duty cycle.

FIXTURE-LEVEL DEMONSTRATION

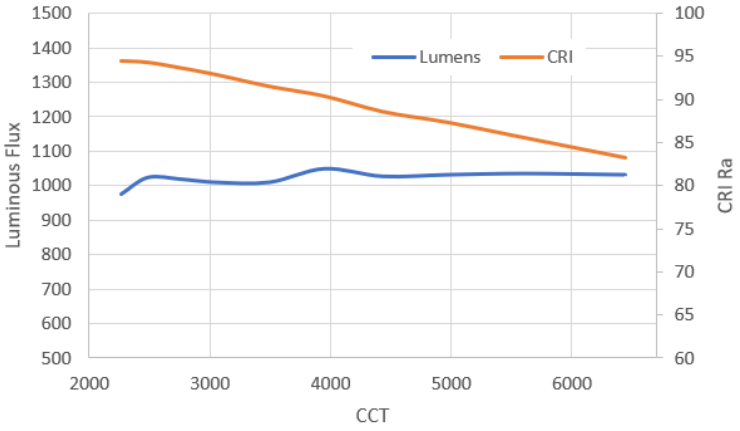
The circuit from design example 1 was adapted to a 4-channel color mixing downlight with Cree LED J Series 2835 LEDs to make a demo with adjustable CCT and brightness. A PCB was designed using four copies of the driver circuit using the NCV30161 and a PIC16F1575 microcontroller.



An aluminum-core PCB was designed for four strings of J Series 2835 LEDs and retrofitted into an off-the-shelf sconce downlight. The electronics were powered by a 36 V AC to DC power supply and installed in the back of the downlight.



The downlight was assembled into a demonstration to show color mixing, controlled by two linear potentiometers that allow users to adjust the CCT and brightness. The demo was able to be tuned to adjust across a wide range of CCTs while maintaining consistent brightness and high CRI.



CONCLUSION

By combining all these techniques of LED driver design, color point tuning, and firmware implementation, a custom color mixing application can be created for any lighting task. The designs and techniques shown here are just a few examples of the many ways that multi-channel LED drive can be accomplished, but they are methods that have been used successfully and can provide a good starting point for new color-mixing designs. All circuits, values, or methods need to be evaluated for their applicability on any new design and be adjusted as needed to fit the use case.