TAOS TSLx257-Colorimeter Demo
Theory & Design

by
Phil Pilgrim, Bueno Systems, Inc.

Introduction

The TAOS colorimeter demo unit is designed to recognize shades from a pre-established set of up to 100 different colors. It communicates with an Adaptive Micro Systems one-line Betabrite display unit. The demo unit is designed to be incorporated into a wheel-of-fortune upon which various colored, numbered "paint chips" are affixed, one at each wheel position. When the wheel is spun and alights upon one of the chips, the sensor demo will identify which color it is and send the color's corresponding number to the Betabrite unit for display.

This document, rather than being a "how to" on the device's use, attempts to chronicle the decisions made during the course of its design and how they impact the final product.

Lighting and Sensor Geometry

No optoelectronic project can even begin without careful attention to lighting. Lighting is everything, and this is especially true with color sensing. Several criteria were established prior to designing the light source. These included:

1. The light had to be *bright*. The demo would be used in exhibit halls typically lighted by high-intensity discharge lamps. The unit's own light source had to be able to swamp out any ambient light.
2. The light had to be *broad-spectrum*. Colors illuminated with the light had to look as natural as possible.
3. The light had to be *small* and *cool-running*. The whole demo, lamp and all, needed to fit into a small enclosure.
4. The light had to be *long-lived*. One doesn't want bulbs burning out right and left during a trade show.

Criteria 1 and 2 almost dictate an incandescent lamp. Unfortunately for incandescent lamps, all four criteria taken together are mutually exclusive. And numbers one through three *always* rule out number four. Clearly a compromise was in order.

Tried and rejected were:

1. Small flashlight heads with krypton bulbs. *Advantages:* Extremely compact, extremely bright, white light. *Disadvantages:* The reflectors are of poor quality and hard to mount, and the bulbs have short lifetimes.
2. Multiple lens-end flashlight bulbs. *Advantages:* No reflector needed. Simple screw-in design. *Disadvantages:* Unpredictable alignment of globe with base. Bulbs pointed every which way. Poor focussing.
3. Halogen reflector lamps. *Advantages:* Reflector is part of bulb. Extremely bright, white light. *Disadvantages:* Too hot. Too large – even the MR11s.
4. Medical instrument lamps. *Advantages:* Small, high-quality reflector lamps. Bright white light. *Disadvantages:* Very expensive with short lifetimes.

After much searching, a lamp and reflector combo were found at Carley Lamps (310-325-8474). Their #880 xenon-filled, bi-pin bulb mated with their #1907 parabolic reflector form an almost perfect compromise. Features are:

1. Adequately high color temperature: 2750 Kelvin.
2. Long life: 1000 hours.
3. Small size: T 1½.
4. 5V operation at 440ma: Can be powered from logic supply!
5. Precision-placed filament: Crucial for good focus.
6. High-precision, well-polished reflector: Again, crucial for good focus.
7. Inexpensive: Around $5 each for the bulb and the reflector.
8. Easy to mount: Reflector is solid aluminum and can be drilled with mounting holes.

The biggest disadvantage is that the #880 is not guaranteed to be stocked, although it has been so far, and their turnaround for production is fairly short, even for small orders.

Carley offers reflectors in both the setscrew and non-setscrew styles. The setscrew styles have a larger ID for the lamp, which must then be glued into a metal sleeve. The non-setscrew style was deemed the better choice, so long as a way could be found to focus the lamp and hold it in place.

The answer to this need came from some Samtec surface-mount strip-line sockets laying about the shop. These had a 2mm spacing – same as the #880 bulb. Plus the pins could be pushed all the way through, making it possible to slide the bulb in and out for best focus and still grip it tight enough to hold it in place. By mounting this socket on a circuit board screwed through another piece of fiberglass and into the rear of the reflector, an adequate mount could be fabricated for the whole illuminating assembly.

The next decision involved the optical geometry. One of the big bugaboos with color sensing is *specular* reflection. This is the reflection from a shiny surface whose angle of reflection is equal to its angle of incidence. Generally the full spectrum of the incident light is reflected, not just that represented by the color of the subject. Specular reflection is what gives a shiny car its bright highlights under the multiple bare light bulbs of a used car lot. So for color sensing, we want to eliminate it.

There are two ways to eliminate the effects of specular reflection: 1. Illuminate the subject at an angle from the sensor's optical axis. By illuminating at an angle, the specular reflection bounces away at the same angle and won't enter the sensing chamber. 2. Use polarizers. Since specular reflection retains the polarization of its source, putting one polarizer in front of the source and another at right angles in front of the sensor will reduce it to almost nothing. But the diffuse color reflection is randomly polarized and won't be so severely affected. The disadvantage with polarizers is that illumination efficiency drops to around 25%, being reduced by 50% twice. For this reason, it was decided to go with angular illumination. However, this is not without it's own disadvantage. Particularly with a single lamp, it places more stringent requirements on maintaining the sensor assembly a fixed distance from its subject. This is because the incident light spot will move horizontally – possibly off-axis – as the assembly is moved in and out.

The following illustration summarizes the resulting sensor geometry.

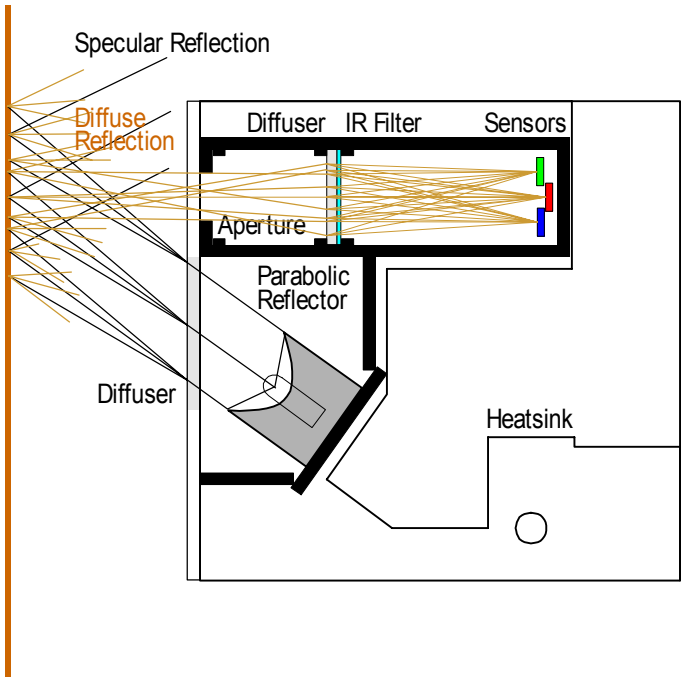

Figure 1. Lighting and Sensor Geometry.

Here we see the parabolic reflector forming a roughly parallel beam of light. It passes through a mild diffuser made of translucent vinyl in order to smooth it out somewhat – not so much to spread it. The diffuse color reflection is allowed to enter the sensor chamber, where it passes through another diffuser and an IR filter (1mm Hoya CM500). The reason for the diffuser in the sensor chamber is that the individual sensors are so far apart. It's the only way they all have a chance of "seeing" the same points of the subject.

The inside of the sensor chamber is flat black and sealed as much as practical against light leaks. Both measures are necessary to prevent stray light from corrupting the sensors' outputs. The flat black minimizes internal reflections, which would allow light coming through the aperture at a steep angle from somewhere outside the circle of interest to be reflected onto the sensors.

Also note the heat sink. This is an oddly-shaped piece of 1/16" sheet aluminum put in place to carry heat away from both the reflector and the 5V regulator. It fits into grooves inside the aluminum housing, thus allowing the entire housing to dissipate the 2-3 watts of heat generated.

Electronics

With a 5V, 440ma bulb dominating power consumption, the first decision was what kind of power supply to use. A linear regulator in such a circumstance could get quite hot, so the initial plan was to use an external 5V switching supply. CUI Stack makes a nice model that looks like a diminutive wall transformer. Subsequent testing, however, showed that it produced 5.16V. With the addition of load-dependent IR losses in the supply cable, it was ruled out as a reliable power source. A reliable constant voltage is crucial to this demo for two reasons: 1. Fluctuating voltage will cause the lamp brightness to vary. 2. The TSLx257 sensors have a high power supply rejection ratio. But most analog to digital converters available in microcontrollers are ratiometric to the supply voltage. So, while the sensor output may remain constant with varying supply voltage, its digitally converted value will not.

For these reasons, it was decided to use local linear regulation. The LM2940-5V is a low-dropout regulator with good regulation (to less than 1/2 LSB over 5V for an 8-bit conversion) and built-in reverse polarity protection. Therefore it could be used with a 6VDC wall transformer with no series protection diode and relatively small dissipation. By mounting it to the underside of the heat sink, it could share its thermal advantages.

The microcontroller posed more of a challenge. A major objective was to make the demo look as simple as possible. Initially it seemed desirable to convert the sensors' voltage output to more than 8 bits, perhaps as many as 12. Because the TSLx257 sensors have a fixed dynamic range, a 12-bit ADC would provide good resolution, even if the full-scale output were kept well shy of the full 5-volt span (to avoid saturation). But no microcontroller with 12-bit converters could be found with less than 28 pins, and a 28-pin microcontroller was deemed overkill for the job at hand. Plus, even the 28-pin devices were either UVPROM or one-time programmable, which would make program changes after delivery much more difficult than, say, a flash-based device. So the decision was made to break the microcontroller in two: 1) An eight-pin one-time programmable PIC chip with four 8-bit analog input channels and some EEPROM, and 2) an 18-pin Ubicom SX chip with onboard flash reprogrammability. The 8-pin PIC would handle all the conversion and white-balancing tasks and communicate its results to the SX, which would find the color in its color lookup table and then format and transmit its output to the Betabrite sign. That way, it would be possible to point to the humble 8-pin PIC and claim (quite accurately) that it was doing all the sensing work, with the rest of the circuitry just being there to run the demo. And the 8-bit ADC, though not breathtakingly precise, has proven adequate for the task.

Because of the SX's onboard reprogrammability, it could be a surface-mount component to save room and, thus, visual complexity – as could the external serial EEPROM for holding the color lookup table and the RS232 driver. Passive components were implemented as through-hole parts because there was room to do so and because surface-mount passives are much harder to position hand solder than surface-mount ICs.

Firmware

The 8-pin PIC, a Microchip Technologies PIC12CE674, includes four 8-bit analog-to-digital input channels and sixteen bytes of program-accessible EEPROM. Three of the analog inputs are used to convert the red, green and blue sensor voltage outputs to digital data. A fourth, digital input is used as a "white balance" pushbutton sensor. Two other digital lines are used to communicate with the second microcontroller. When the white balance pushbutton is pressed, the current RGB input readings are saved in EEPROM as the white reference. Upon power up, these readings are read from EEPROM and are subsequently used to correct the raw RGB data until a new button push establishes new values. The 12CE674 operates from its own 4MHz internal RC clock at a 1MIPS execution rate.

The PIC chip communicates asynchronously (NRZ) with the SX chip at 2400 baud. Because of the PIC chip's imprecise RC clock, the SX sends it a continuous stream of data, filling in with zero bytes (nulls) where necessary. This is to provide the PIC with a continuous baud rate reference with which it can keep its own software clock synchronized. The PIC chip accepts two commands from the SX: Raw and Balanced. Upon receiving the Raw command, it sends a continuous stream of data to the SX in the form:

%rrggbbss<cr>,

where *rr*, *gg*, and *bb* are each two hex digits, representing the output from their respective sensors. *ss* is a two-digit hex value representing the status of the data returned. The upper four bits are flags for raw data enabled and red, green, and blue saturation. The saturation flags are set for each color component whose raw value is 255. The second digit of the status is always "0".

Upon receiving the Balanced command, the PIC switches to balanced mode, in which each color component is subjected to the following correction formula:

Balanced value = Raw value * 256 / Reference value

The "Reference value" for each color is obtained whenever the white balance pushbutton is pressed and also at power-up, when it is read from the PIC's onboard serial EEPROM. If the above formula should return a value greater than 255, it saturates to 255 instead. White balancing is necessary because the sensors do not give equal voltage outputs for white light – green being the more predominant. Also, the color temperature of the lamp can upset the balance. The Carley #880 lamp, for example, with a color temperature of 2750 Kelvin is a little heavy in the red part of the spectrum. White balancing with a reference white color sample cancels out all these anomalies.

Once each balanced value is computed, it is sent to the SX in the same format as above for raw data, except that the MSB of the status byte is cleared. Saturation bits still pertain to raw data, however, not balanced values.

The SX chip operates on a sample-and-lookup cycle, wherein it reads the next available data from the PIC, then compares the color with each defined color in an off-chip EEPROM lookup table, trying to find the closest one. What is meant by "closest" bears some discussion. When a human says that two colors are close, he means that they "look somewhat alike", that is, their distance from each other in some multidimensional color space is short. Much effort has been spent in the last 100 years to define the human perceptual color space. The goal is a mathematically precise coordinate system in which the Cartesian distance between two colors approximates the psychologically-perceived distance between them. Two things are certain: 1) The space is *at least* three-dimensional, and 2) the RGB space is nowhere close to being a candidate. Perhaps the best candidate so far is the three-dimensional CIElab color space. There are well-defined formulae for converting from RGB to CIElab and back again, but the formulae are highly non-linear and difficult to implement in an 8-bit microcontroller.

But re-examining the problem leads to some simplifications. If the problem had been, "Take the presented color, which maybe you've never seen before, and return the closest stored color to it", there would be some justification for using CIElab. After all, we'd want the device to return a color that at least *looks like* the color it's viewing. But that's not the job at hand. It is, rather, "You've seen this color before; which one is it?" This is a much easier problem, because we know there's one color in the list that matches almost exactly. And zero distance in one color space is zero in any other, as is "really, really close". We've got data in the RGB space, so why not just do the comparisons there?

Now for some further simplification. Mention was made above of the Cartesian distance. This would be expressed in RGB space as:

$$D_{ij} = SQRT[(Red_i - Red_j)^2 + (Green_i - Green_j)^2 + (Blue_i - Blue_j)^2]$$

Just to compare two distances, we can omit the square-root, but this is still a lot of multiplication.

The color we're looking for is *really close* to the one we're looking at, remember. And hopefully the other colors will be *really far away* by comparison. So why not just look for the coordinate along which the colors are farthest apart, be it red, green or blue, and call the distance along that one coordinate the distance between the colors? That's an easy number to calculate: it's just the absolute value of a numeric difference. So this, in fact is what we do. It could be called a *minimax* strategy, since it endeavors to find the color which minimizes the maximum distance along any RGB coordinate axis. Algorithmically, it works like this:

```
get Color;
MinDiff = 255;
for (i = 0; i <= 99; i++) {
    if SavedColor[i].defined {
        MaxDiff = max(abs(SavedColor[i].red - Color.red),
                      abs(SavedColor[i].green - Color.green),
                      abs(SavedColor[i].blue - Color.blue));
        if (MaxDiff < MinDiff) {
            MinDiff = MaxDiff;
            ClosestColor = i
        }
    }
}
return ClosestColor
```

The saved table, the *SavedColor* array in the above code, is implemented in the serial EEPROM external to the SX. Each of the 100 saved colors is stored as a four-byte entry, one byte each for the red, green, and blue components, plus a byte defining whether that color is defined. In each of the SX's sample-and-lookup cycles, the sampled color is compared with each color in the table using the above algorithm. The index of the closest color is then the color number that's sent to the sign.

In order for the sign to actually display a color number, that number must come up multiple times in a row, the multiple being a constant in the program. This is to prevent displaying false color numbers when the sensor is between two colors as the wheel is spinning. When the same color persists for a really long time, say 15 seconds, again set by a constant in the program, the SX commands the sign to revert to a previously stored advertising message. When colors are rapidly changing, such that no two consecutive colors are the same, the SX causes the "Color:" message to display, but with a "spinning" character animation in place of a number.

Schematic

A schematic diagram of the color sensor demo unit appears below.

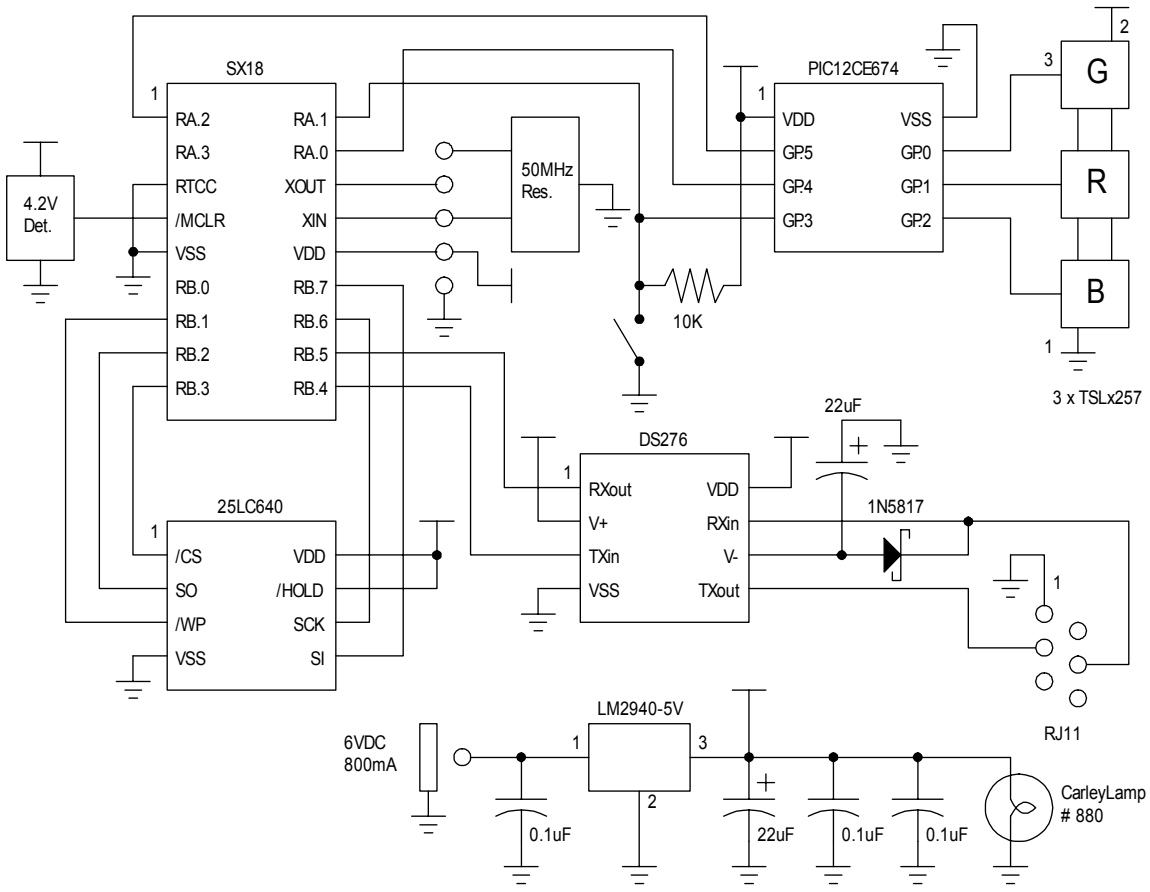

Figure 2. Color Sensor Demo Schematic