led_balancing

July 1, 2022

This design has several different color discrete LEDs as well as seven segment displays. It is desired that the luminous intensity of the seven segment displays be balanced, the luminous intensity of the discrete LEDs be balanced, and the ratio of luminous intensity between the seven segment displays and descrete LEDs be controlled.

Components modelled in the following calculations: 1. Green 0805 LED, 150080GS750000 2. Red 0805 LED, 150080RS75000 3. Blue 0805 LED, 150080BS75000 4. Green Seven Segment Display, LTC-4727G 5. Red Seven Segment Display, LTC-4727JR 6. Yellow Seven Segment Display, LTC-4727JS

[1]:
```
%reload_ext lab_black
from scipy.interpolate import interp1d
from scipy.optimize import root_scalar
from numpy import genfromtxt
from eseries import find_nearest, E12
from worstcase import unit
```

[2]:
```
def vf2if(pn):
    # Given a list of forward voltages and the corresponding forward currents,
    # return a function which interpolates the given data.
    filename = pn + "_vf2if.txt"
    data = genfromtxt(filename, delimiter=",")
    return interp1d(data[:, 0], data[:, 1], fill_value="extrapolate")
```

[3]:
```
def if2li(pn):
    # Given a list of forward currents and the corresponding luminous␣
 ↪intensities,
    # return a fuction which interpolates the given data.
    filename = pn + "_if2li.txt"
    data = genfromtxt(filename, delimiter=",")
    return interp1d(data[:, 0], data[:, 1], fill_value="extrapolate")
```

[4]:
```
def li2res(pn, vcc, li):
    # Given the desired luminous intensity and LED drive voltage,
    # return the value of the series resistor.
    pn_if2li = if2li(pn)
    if_rootfun = lambda x: pn_if2li(x) - li
    if_sol = root_scalar(if_rootfun, x0=10, bracket=[0, 100])
    assert if_sol.converged

    pn_vf2if = vf2if(pn)
    vf_rootfun = lambda x: pn_vf2if(x) - if_sol.root
    vf_sol = root_scalar(vf_rootfun, x0=3, bracket=[0, 10])
    assert vf_sol.converged

    res = 1000 * (vcc - vf_sol.root) / if_sol.root
    return find_nearest(E12, res) * unit.ohm
```

[5]:
```
def balanced_discrete_leds(vcc, li):
    grn = li2res("150080GS75000", vcc, li)  # Green 0805 LED
    red = li2res("150080RS75000", vcc, li)  # Red 0805 LED
    blu = li2res("150080BS75000", vcc, li)  # Blue 0805 LED
    return grn, red, blu
```

[6]:
```
def balanced_seven_segment_displays(vcc, li):
    grn = li2res("LTC-4727G", vcc, li * 1000)  # Green Seven Segment
    red = li2res("LTC-4727JR", vcc, li * 1000)  # Red Seven Segment
    yel = li2res("LTC-4727JS", vcc, li * 1000)  # Yellow Seven Segment
    return grn, red, yel
```

[7]:
```
grn, red, blu = balanced_discrete_leds(vcc=3.3, li=20)
print("green:", grn)
print("red:", red)
print("blue:", blu)
```

```
green: 1500.0 ohm
red: 680.0 ohm
blue: 330.0 ohm
```

[8]:
```
grn, red, yel = balanced_seven_segment_displays(vcc=3.3, li=0.8)
print("green:", grn)
print("red:", red)
print("yellow:", yel)
```

```
green: 220.0 ohm
red: 220.0 ohm
yellow: 120.0 ohm
```