LAMPIRAN 1

Program *Python*:

```
import os, sys, math, time
from PIL import Image, ImageDraw, ImageFile, ImageFont
import picamera
from collections import OrderedDict
from fractions import Fraction
#scan a column to determine top and bottom of area of
lightness
def
getSpectrumYBound(pix,x,middleY,spectrum_threshold,spectrum_th
reshold_duration):
      c=0
      spectrum_top=middleY
      for y in range(middleY,0,-1):
            r, g, b = pix[x,y]
            brightness=r+g+b
            if brightness<spectrum_threshold:
                  c=c+1
                  if c>spectrum_threshold_duration:
                        break;
            else:
                  spectrum_top=y
                  c=0
      c=0
      spectrum_bottom=middleY
      for y in range(middleY,middleY*2,1):
            r, g, b = pix[x,y]
            brightness=r+g+b
            if brightness<spectrum_threshold:
                  c=c+1
                  if c>spectrum_threshold_duration:
                        break;
            else:
                  spectrum_bottom=y
                  c=0
      return spectrum_top,spectrum_bottom
#find aperture on right hand side of image along middle line
def findAperture(pix,middleX,middleY):
      aperture_brightest=0
      aperture_x=0
      for x in range(middleX,im.size[0],1):
```

```
			r, g, b = pix[x,middleY]
			brightness=r+g+b
			if brightness>aperture_brightest:
				aperture_brightest=brightness
				aperture_x=x
		aperture_threshold=aperture_brightest*0.9
		aperture_x1=aperture_x
		for x in range(aperture_x,middleX,-1):
			r, g, b = pix[x,middleY]
			brightness=r+g+b
			if brightness<aperture_threshold:
				aperture_x1=x
				break
		aperture_x2=aperture_x
		for x in range(aperture_x,im.size[0],1):
			r, g, b = pix[x,middleY]
			brightness=r+g+b
			if brightness<aperture_threshold:
				aperture_x2=x
				break
		aperture_x=(aperture_x1+aperture_x2)/2
		spectrum_threshold_duration=64
		apertureYBounds=getSpectrumYBound(pix,aperture_x,middleY,
aperture_threshold,spectrum_threshold_duration);
		aperture_y=(apertureYBounds[0]+apertureYBounds[1])/2
		aperture_height=(apertureYBounds[1]-
apertureYBounds[0])*0.9
		return { 'x':aperture_x, 'y':aperture_y,
'h':aperture_height, 'b': aperture_brightest }
# draw aperture onto image
def drawAperture(aperture,draw):
		draw.line((aperture['x'],aperture['y']-
aperture['h']/2,aperture['x'],aperture['y']+aperture['h']/2),f
ill="#000")
#draw scan line
def drawScanLine(aperture,spectrumAngle,draw):
		xd=aperture['x']
		h=aperture['h']/2
		y0=math.tan(spectrumAngle)*xd+aperture['y']
		draw.line((0,y0-h,aperture['x'],aperture['y']-
h),fill="#888")
		draw.line((0,y0+h,aperture['x'],aperture['y']+h),fill="#8
88")
```

```
#return an RGB visual representation of wavelength for chart
def wavelengthToColor(lambda2):
    # Based on:
http://www.efg2.com/Lab/ScienceAndEngineering/Spectra.htm
    # The foregoing is based on:
http://www.midnightkite.com/color.html
    factor = 0.0;
    color=[0,0,0]
    #thresholds = [ 380, 440, 490, 510, 580, 645, 780 ];
    #                   vio  blu  cyn  gre  yel  end
    thresholds =  [ 380, 400, 450, 465, 520, 565, 780 ];
    for i in range(0,len(thresholds)-1,1):
        t1 = thresholds[i]
        t2 = thresholds[i+1]
        if (lambda2 < t1 or lambda2 >= t2):
           continue
        if (i%2!=0):
           tmp=t1
           t1=t2
           t2=tmp
        if i<5:
           color[ i % 3] = (lambda2 - t2) / (t1-t2)
        color[ 2-i/2] = 1.0;
        factor = 1.0;
        break
    #Let the intensity fall off near the vision limits
    if (lambda2 >= 380 and lambda2 < 420):
        factor = 0.2 + 0.8*(lambda2-380) / (420 - 380);
    elif (lambda2 >= 600 and lambda2 < 780):
        factor = 0.2 + 0.8*(780 - lambda2) / (780 - 600);
    return (
int(255*color[0]*factor),int(255*color[1]*factor),int(255*colo
r[2]*factor) )
##name=sys.argv[0]
##shutter=long(sys.argv[1])
os.system('raspistill -w 640 -h 580 -ISO 6000000 -br 80 -co
100 -o _rawsample.jpg')
rawFilename="_rawsample.jpg"
##time.sleep(3)
##camera.capture(rawFilename,resize=(1296,972))
im = Image.open(rawFilename)
middleY=im.size[1]/2
middleX=im.size[0]/2
```

```
pix = im.load()
#print im.bits, im.size, im.format
draw=ImageDraw.Draw(im)
spectrumAngle=0.03
aperture=findAperture(pix,middleX,middleY)
#print aperture
drawAperture(aperture,draw);
drawScanLine(aperture,spectrumAngle,draw)
##wavelengthFactor=0.892 # 1000/mm
wavelengthFactor=0.892*2.0*600/650 # 500/mm
xd=aperture['x']
h=aperture['h']/2
step=1
last_graphY=0
maxResult=0
results=OrderedDict()
for x in range(0,xd*7/8,step):
      wavelength=(xd-x)*wavelengthFactor
      if (wavelength<380):
            continue
      if (wavelength>1000):
            continue
      #general efficiency curve of 1000/mm grating
      eff=(800-(wavelength-250))/800
      if (eff<0.3):
            eff=0.3
      #notch near yellow maybe caused by camera sensitivity
      mid=575
      width=10
      if (wavelength>(mid-width) and wavelength<(mid+width)):
            d=(width-abs(wavelength-mid))/width
            eff=eff*(1-d*0.1);
      #up notch near 590
      mid=588
      width=10
      if (wavelength>(mid-width) and wavelength<(mid+width)):
            d=(width-abs(wavelength-mid))/width
            eff=eff*(1+d*0.1);
      y0=math.tan(spectrumAngle)*(xd-x)+aperture['y']
      amplitude=0
      ac=0.0
      for y in range(int(y0-h),int(y0+h),1):
            r, g, b = pix[x,y]
            q=math.sqrt(r*r+b*b+g*g*1.5);
##          q=r+b+g*2
            if y<(y0-h+2) or y>(y0+h-3):
                  q=q*0.5
            amplitude=amplitude+q
            ac=ac+1.0
      amplitude=amplitude/(ac)/(eff)
      #amplitude=1/eff
      results[str(wavelength)]=amplitude
      if amplitude>maxResult:
            maxResult=amplitude
      graphY=amplitude/50*h
      draw.line((x-step,y0+h-last_graphY, x,y0+h-
graphY),fill="#fff")
      last_graphY=graphY
for wl in range(400,1001,50):
      x=xd-(wl/wavelengthFactor)
      y0=math.tan(spectrumAngle)*(xd-x)+aperture['y']
      draw.line((x,y0+h+5, x,y0+h-5))
      draw.text((x,y0+h+15),str(wl))
exposure=maxResult/(255+255+255)
print "ideal exposure between 0.15 and 0.30"
print "exposure=",exposure
if (exposure<0.15):
      print "consider increasing shutter time"
elif (exposure>0.3):
      print "consider reducing shutter time"
#save image with markup
outputFilename="_outsample.jpg"
ImageFile.MAXBLOCK = 2**20
im.save(outputFilename, "JPEG", quality=80, optimize=True,
progressive=True)
#normalise results
for wavelength in results:
      results[wavelength]=results[wavelength]/maxResult
#save csv of results
csvFilename="sample.csv"
csv = open(csvFilename, 'w')
csv.write("wavelength,amplitude\n")
for wavelength in results:
      csv.write(wavelength)
      csv.write(",")
      csv.write("{:0.3f}".format(results[wavelength]))
      csv.write("\n");
csv.close()
#generate spectrum diagram
antialias=4
w=600*antialias
h2=300*antialias
h=h2-20*antialias
sd = Image.new('RGB', (w, h2), (255,255,255))
draw = ImageDraw.Draw(sd)
w1= 380.0
w2= 780.0
f1 = 1.0/w1;
f2 = 1.0/w2;
for x in range(0,w,1):
      # Iterate across frequencies, not wavelengths
      lambda2 = 1.0/(f1-(float(x)/float(w)*(f1-f2)))
      c=wavelengthToColor(lambda2)
      draw.line((x, 0, x, h),fill=c)
pl=[]
pl.append( (w,0) )
pl.append( (w,h) )
for wavelength in results:
      wl=float(wavelength)
      x=int( (wl-w1)/(w2-w1) * w )
      #print wavelength,x
      pl.append( ( int(x), int((1-results[wavelength])*h) ) )
pl.append( (0,h) )
pl.append( (0,0) )
draw.polygon(pl,fill="#FFF")
draw.polygon(pl)
font = ImageFont.truetype("Lato-Regular.ttf", 12*antialias)
draw.line((0,h,w,h),fill="#000",width=antialias)
for wl in range(400,1001,10):
      x=int( (float(wl)-w1)/(w2-w1) * w )
      draw.line((x,h,
x,h+3*antialias),fill="#000",width=antialias)
for wl in range(400,1001,50):
      x=int( (float(wl)-w1)/(w2-w1) * w )
      draw.line((x,h,
x,h+5*antialias),fill="#000",width=antialias)
      wls=str(wl)
      tx=draw.textsize(wls,font=font)
      draw.text((x-tx[0]/2,h+5*antialias),wls,font=font)
```

```
#save chart
sd = sd.resize((w/antialias,h/antialias), Image.ANTIALIAS)
outputFilename="_chartsample.png"
sd.save(outputFilename, "PNG", quality=95, optimize=True,
progressive=True)
```