

Week 2(2020.09.14)

Lab 2 : Camera with RPi

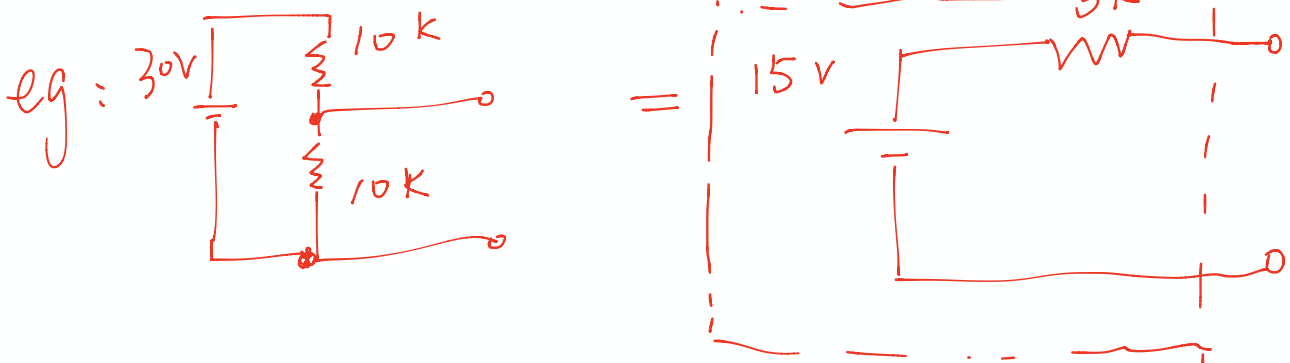
- Electronic components

- 线性器件：I、U成线性关系
- Passive components(被动器件)
 - 内部没有任何形式的电源
- 电压定律：一个环路内电压之和为0
 - 输电时，P不变，高电压 \rightarrow I减小，能量损耗减小
- Register color coding【色环电阻】：色环用于表示电阻值
 - 对并联的估算：两个电阻差不多大： $R(\text{total}) = R/2$ ；一个比另一个大很多： $R(\text{total}) = R(\text{smaller})$ ；
- Potentiometer 可变电阻：



- Sources 电源

- Thevenin Equivalent Circuit (等效电路)：含独立电源的线性电阻单口网络N，就端口特性而言，可以等效为一个电压源和电阻串联的单口网络。电压源的电压等于单口网络在负载开路时的电压；电阻 R_0 是单口网络内全部独立电源为零值时所得单口网络 N_0 的等效电阻。



▶ 最大功率传输: $P_{\max} = (V_{Th})^2 / 4R_{Th}$

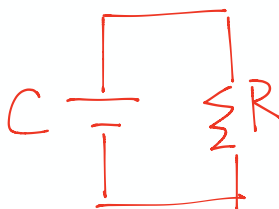
○ Capacitor:

▶ 有极性电容于无极性电容的区别在于,介质的不同:有极性电容大多(还有别材料,如极化材料)采用电解质做介质材料。另外,不同的电解质材料和工艺制造出的有极性电容同体积的容量也会不同。再有就是耐压和使用介质材料也有密切关系;

• Eg: 由阳极的铝箔和阴极的电解液分别形成两个电极,由阳极铝箔上产生的一层氧化铝膜做为电介质的电容。由于这种结构,使其具有极性,当电容正接的时候,氧化铝膜会由于电化反应而保持稳定,当反接的时候,氧化铝层会变薄,使电容容易被击穿损坏。所以电解电容在电路中必须注意极性,不能接反正负极。

▶ Capacitor Behavior:

• RC电路

 $C \frac{dv}{dt} = I = \frac{-V}{R} \Rightarrow V = V_0 e^{-t/RC}$

○ charging

○ Time-delay circuit; One minute of Power

○ Inductor



- Buck Convertor(降压变换器)

- Duty cycle



Bulk Convertor

- How to you convert one voltage to another?



$$V_L = L \frac{dI_L}{dt}$$

$$\Delta I_{L_{on}} = \int_0^{t_{on}} \frac{V_L}{L} dt = \frac{(V_i - V_o)}{L} t_{on}, t_{on} = DT$$

$$\Delta I_{L_{off}} = \int_{t_{on}}^{T=t_{on}+t_{off}} \frac{V_L}{L} dt = -\frac{V_o}{L} t_{off}, t_{off} = (1-D)T$$

Homework: derive the Boost Voltage as function of D (duty cycle)

Transformer is another way.
More at DC-DC power module



$$\frac{V_i - V_o}{L} t_{on} - \frac{V_o}{L} t_{off} = 0$$

$$D = \frac{V_o}{V_i}$$

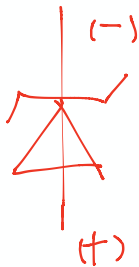
duty cycle, 指降压比

SHUST-DM442 Analog Circuit System Design

25

- Diode (二极管)

- 击穿之后还能恢复 (一般二极管击穿之后就损坏了)



- PCB Manufacturing and assembly
- Lab2 Identify Target -- code and comment

- import numpy as np
- import cv2
-
- cap = cv2.VideoCapture(0)
- cap.set(3,640)

- cap.set(4,480) 并打开摄像头
-
- while True:
- ret, img = cap.read()
- #cap.read()返回两个参数赋给两个值。第一个参数ret的值为True或False，代表有没有读到图片。第二个参数是frame，是当前截取一帧的图片。
- cv2.circle(img, (320,240), 2, (0, 0, 255), 3);
- #cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]]);
- #thickness：圆形轮廓的粗细（如果为正）。负厚度表示要绘制实心圆；lineType：圆边界的类型；shift：中心坐标和半径值中的小数位数
- gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #色彩空间的转化
- cv2.blur(gray,(3,3))
- #进行均值滤波,img表示输入的图像，(3,3)表示进行均值滤波的方框大小；(均值滤波：图像去噪)
- ret, binary = cv2.threshold(gray,50,255,cv2.THRESH_BINARY_INV)
- #threshold 进行阈值计算；cv2.threshold(src, thresh, maxval, type[, dst])→ src：表示的是图片源；thresh：表示的是阈值（起始值）；
- #maxval：表示的是最大值；type：表示的是这里划分的时候使用的是哪种类型的算法，常用值为0 (cv2.THRESH_BINARY)
- (_,contours,hierarchy) = cv2.findContours(binary,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
- #cv2.findContours()函数：查找检测物体的轮廓；
- #注意：cv2.findContours()函数接受的参数为二值图，即黑白的（不是灰度图），所以读取的图像要先转成灰度的，再转成二值图
- n=len(contours) #len()方法返回对象（字符串、列表、元组等）长度或项目个数
- contoursImg=[] #创建数组
- area = 0
- index = 0
- for i in range(n):
- if(cv2.contourArea(contours[i]) > area):
- area = cv2.contourArea(contours[i]) #contourArea函数：计算轮廓区域
- index = i
- for i in range(n):
- M= cv2.moments(contours[index]) #每个时刻的轮廓
- cx = 0
- Cy = 0 #坐标
- if(M['m00'] != 0):
- cx = int(M['m10']/M['m00']) #通过求重心坐标求x、y坐标
- cy = int(M['m01']/M['m00']) #<https://blog.csdn.net/u013700771/article/details/80687228>;
- text1 = "X:"+(str(cx-320));
- text2 = "Y:"+(str(cy-240)); #str() 函数将对象转化为适于人阅读的形式。语法：class str(object=)
- cv2.putText(img, text1, (200,280), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,255), 2, 8) #在图像上绘制文字的函数
- cv2.putText(img, text2, (200,320), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,255), 2, 8)
- cv2.line(img, (320,240), (cx,cy), (0, 255, 0), 1)
- cv2.line(img, (320,240), (320,cy), (0, 255, 0), 1)
- cv2.line(img, (cx,cy), (320,cy), (0, 255, 0), 1)
-
- img=cv2.circle(img,(cx,cy),2,(0,0,255),4)
- temp=np.zeros(img.shape,np.uint8)
- contoursImg.append(temp)
- contoursImg[i]=cv2.drawContours(img,contours,i,(255,0,0), 3)
-
- cv2.imshow('video',img) #显示绘制结果
-

- `k = cv2.waitKey(30) & 0xff`
- `if k == 27: # press 'ESC' to quit`
- `break`
-
- `cap.release()`
- `cv2.destroyAllWindows()`
-
-
-
-

Lab2上课感想：

首先是我遇到的困难。我在前几步都挺顺利的，也玩的非常开心，但是从人脸识别开始就遇到了困难。一开始我用阿里云镜像源下载opencv的时候非常地慢，在咨询学姐后她告诉我，**暂时无法解析域名的原因是没连上网**（我刚开始还真的没注意到我没连上网ToT），但是连网之后还是遇到了一样的问题。于是我又一次咨询了学姐，得知有两个可能的原因：**一个就是网络不好，另一个就是源有问题（应该是指链接源的过程出了问题）**，如果能通过VNC连接树莓派的话可能就是后一个原因。于是我尝试换源（毕竟有几个同学换源后下载速度变得很快），然而换源后依然下载速度“感人”ToT。在折腾了三个小时后我决定直接烧录队友已经下载好opencv的卡。在复制了队友的卡之后终于能够开始使用opencv了。

bonus部分我本来想试着做一下挑战自我，但是还是写不出来。最后我只是分析了一下队友写出来的代码，写了一些注释（以便我自己能看懂）也算是通过实例了解了一下python和opencv的一些函数和写代码的思路吧。（代码及注释在上面的笔记中）