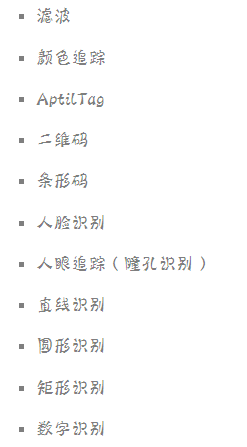
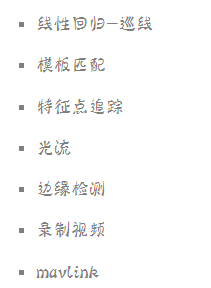
## Openmv基本介绍：

* 它拥有 10个IO，一个ADC/DAC，一个SPI, 一个I2C，两个UART，三个舵机接口。
* 他的主控芯片是性能非常强大的STM32F7，主频有216MHZ，2M flash。在色块追踪上，帧率可以到达85~90帧，速度非常快。
* 他的感光元件是OV7725，分辨率为640×480。
* 可以通过串口和Arduino，51，STM32等任何单片机通信。
* 也可以通过额外的ttl-rs232或者ttl-rs485模块和PLC通信。
* 同时，它的接口是可叠加的，可以用LCD来显示。
* 你可以安装WiFi来无线传输视频，OpenMV的wifi扩展板采用atwinc1500，传输速率高达48Mbps。直接在浏览器输入网址，即可实时查看摄像头图像。
* SD卡可以用来保存图片，录制视频。或者在调用模板匹配算法的时候，使用sd卡存储的图片就行匹配识别。
  + 图像直方图，可以实时查看颜色直方图，有rgb，gray，lab，yuv

Openmv内置算法：

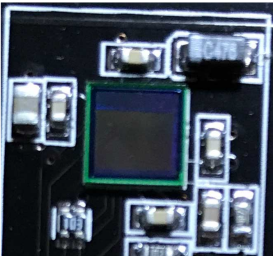
 

所有引脚均可承受5V电压，输出电压为3.3V。所有引脚都可以提供最高25mA的拉电流(source)或灌电流(sink)。在ADC或DAC模式下，P6不能承受5V电压。引脚总共可提供最高120mA的拉电流(source)或灌电流(sink)。VIN可以在3.6V和5V之间。不要从OpenMV Cam的3.3V引脚输出超过250mA的电流。

## 图像处理知识：

将光学信号转变成电信号的一个装置。在计算机视觉中，最简单的相机模型是小孔成像模型：小孔模型仍然是目前最广泛使用的相机模型。

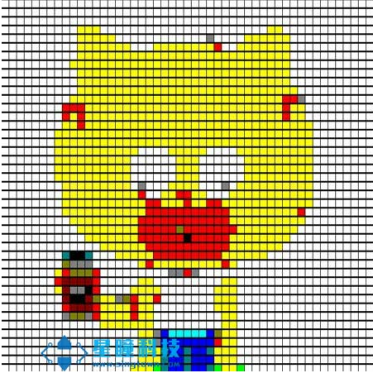
图像透过镜头，照在一个感光芯片上，感光芯片可以把光照的波长和强度等信息转成计算机（数字电路）可以识别的数字信号，感光元件是长这样的：

中间的方形元件就是感光元件。

像素和分辨率：

像素:

感光元件是有很多个感光点构成的，比如有640\*480个点，每个点就是一个像素，把每个点的像素收集整理起来，就是一副图片，那么这张图片的分辨率就是640\*480：

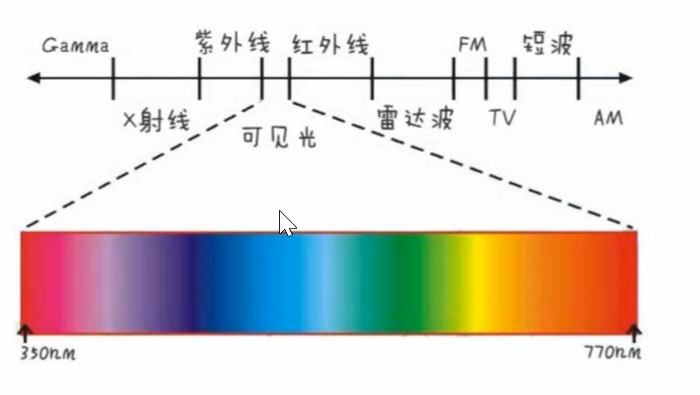


分辨率：

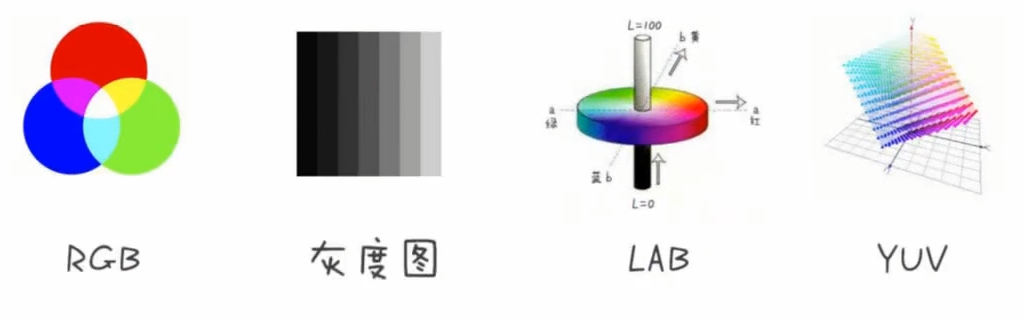
帧率（FPS）就是每秒钟处理的图片数量，如果超过20帧，人眼就基本分辨不出卡顿。当然，如果用在机器上，帧率是越高越好的，OpenMV的最大帧率对比：

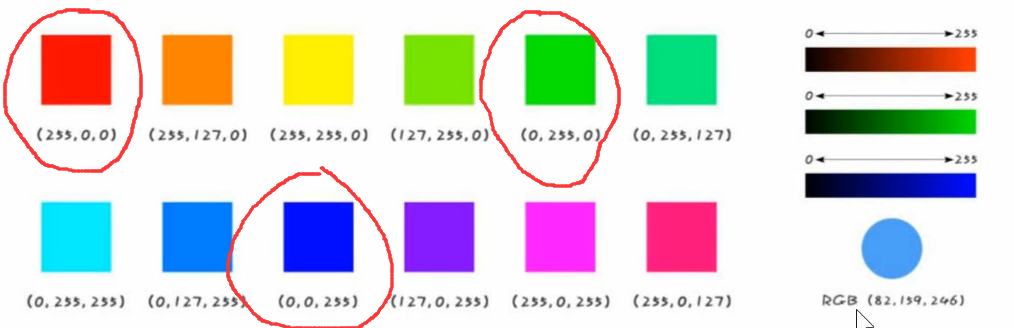
颜色：

波长不同的电磁波，



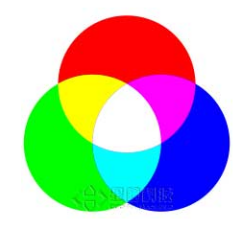
根据人眼的视觉效果，可以通过RGB，CMYK，HSB，LAB色域，来将可见光的颜色描述出来。





RGB三原色：

三原色的原理不是物理原因，而是由于人的生理原因造成的。人的眼睛内有几种辨别颜色的锥形感光细胞，分别对黄绿色、绿色和蓝紫色（或称紫罗兰色）的光最敏感（波长分别为564、534和420纳米）。所以RGB经常用于显示器上，用来显示图片。





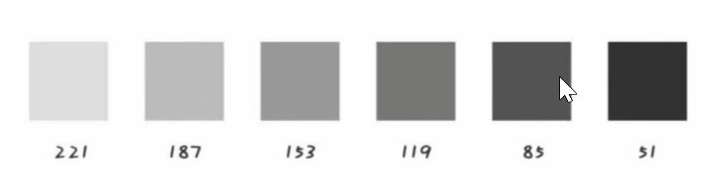
Lab亮度对比度：

Lab颜色空间中，L亮度；a的正数代表红色，负端代表绿色；b的正数代表黄色，负端代表兰色。不像RGB和CMYK色彩空间，Lab颜色被设计来接近人类视觉。

因此L分量可以调整亮度对，修改a和b分量的输出色阶来做精确的颜色平衡。

注意：在OpenMV的查找色块的算法中，运用的就是这个LAB模式！

灰度图：



镜头焦距：

是指镜头光学后主点到焦点的距离，是镜头的重要性能指标。镜头焦距的长短决定着拍摄的成像大小，视场角大小，景深大小和画面的透视强弱。焦距越长，视角越小，所成的像越大。

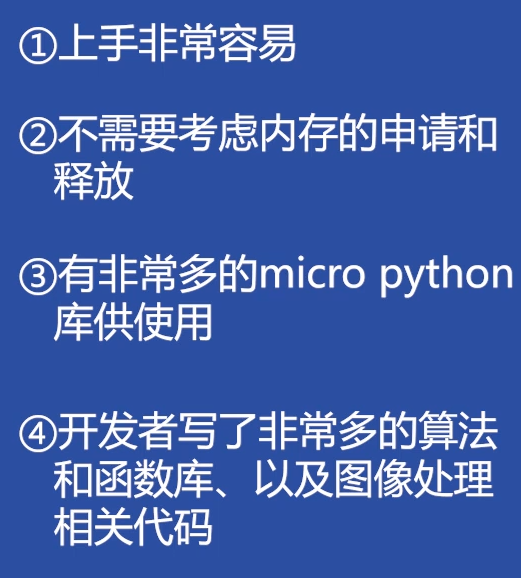
镜头滤片：



正常颜色应用中，是不需要红外光的，因为红外光也会使感光元件受到反应，就使得整个画面泛白。所以我们在镜头上放一个只能通过波长650nm以内的滤光片，就把红外光截止了。

## Python语法

Python是交互式的解释型语言，可以在openmv的串口助手中进行Python的编写。



1.列表list[a,b,c,d]

list=[1,23,52,69,85]



注意是list=[]

len(list)=5//获取长度

list[3]=69//获取某位元素

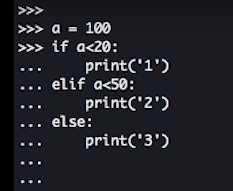
list.append(95), list=[1,23,52,69,85,95]//在列表最后添加元素

list.insert(1,55), list=[1,55,23,52,69,85,95]//在列表某位添加元素

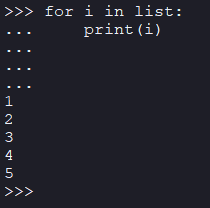
list.pop(),list=[1,55,23,52,69,85]//删除最后一位元素

list[4]=2000,对某位进行赋值

1. 元组tuple(a,b,c,d) 初始化后不能进行插入和删除元素
2. If else

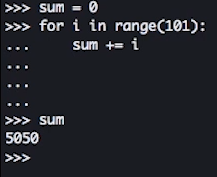


1. for



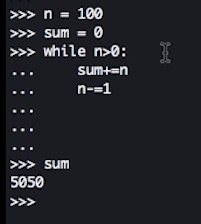
注意第一行最后的冒号

1. 计算和

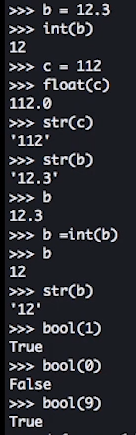


range(101),代表从0到100为101个数

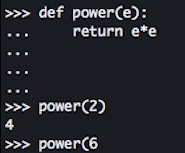
6.



7.进行精确度转换



1. 函数定义

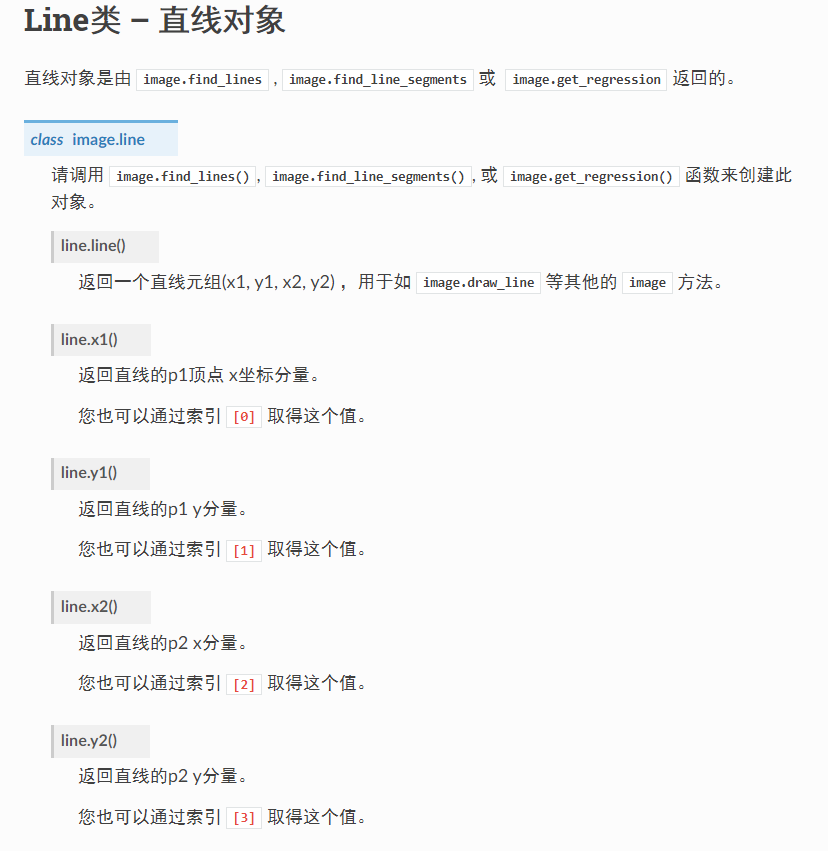


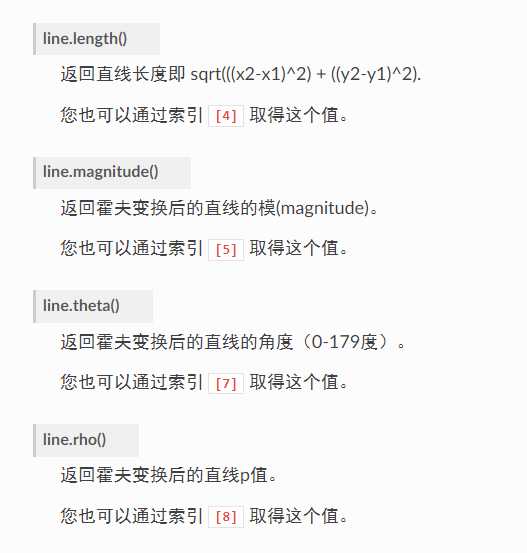
## 形状识别





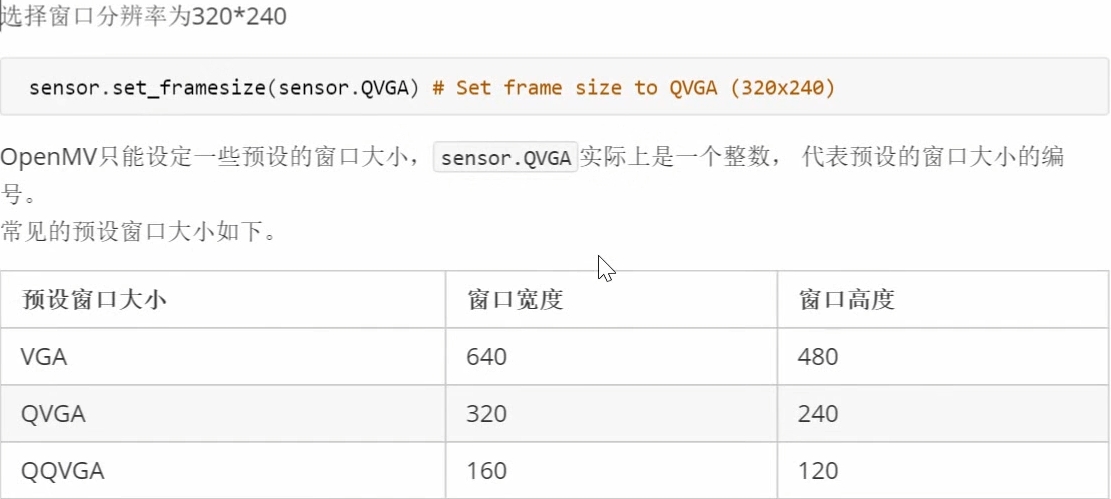






## 练1 hello world

Hello world

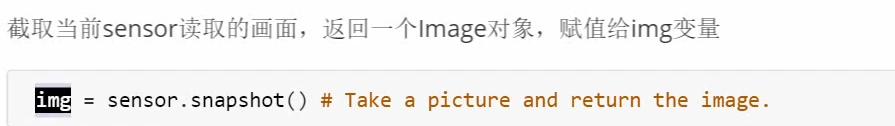


那为什么要预设这些不同窗口大小呢?原因是，有些机器视觉的算法比较耗内存，如果我们都用高清的图像质量的话，一方面帧率会很低，另一方面容易发生内存溢出，所以我们需要在图像质量跟性能之间做一些折中。

画质稳定后再进行图像的读取，保证读取数据准确性



返回图像



那大家有没有感到奇怪，好像我们的程序里面并没有发送图像到IDE的这个操作?那实时显示的视频是哪来的?

实际上，在OpenMV与OpenMV IDE usb相连的时候，每当OpenMV执行[snapshot 拍照的时候;OpenMV还会将当前图像压缩，通过特定的传输协议发送给OpenMV IDE。所以我们才能在FrameBuffer视频显示区域看到实时返回的图像。同时因为是串口发送，会占用很多资源，所以链接IDE时候的帧率大概是OpenMV脱机运行的一半。

计时模块 time

micropython的time模块与python标准库的time模块不是一个包，这里的time模块主要起计时的作用，用于追踪过去的时间。

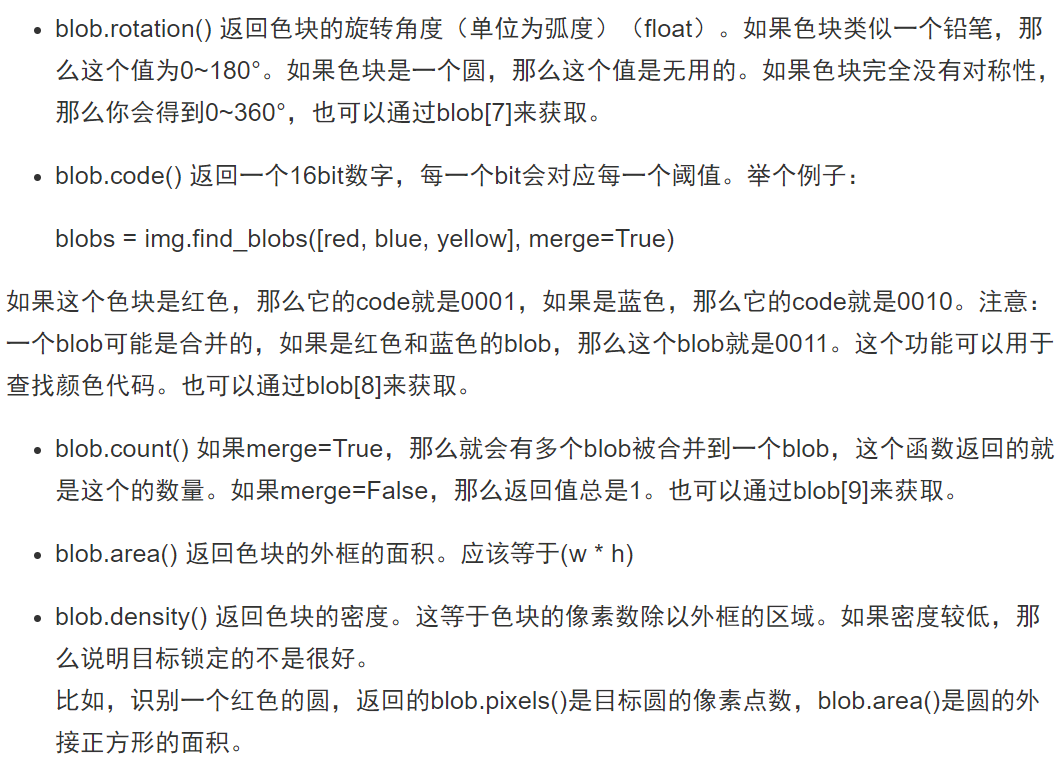


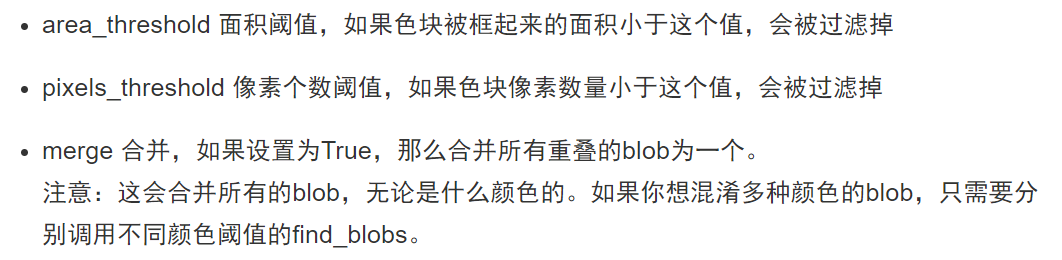


## 练2 色块寻找

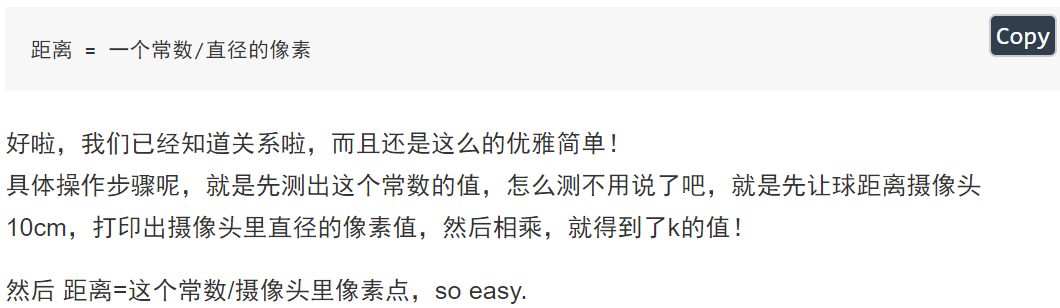
（单色块，多色块）

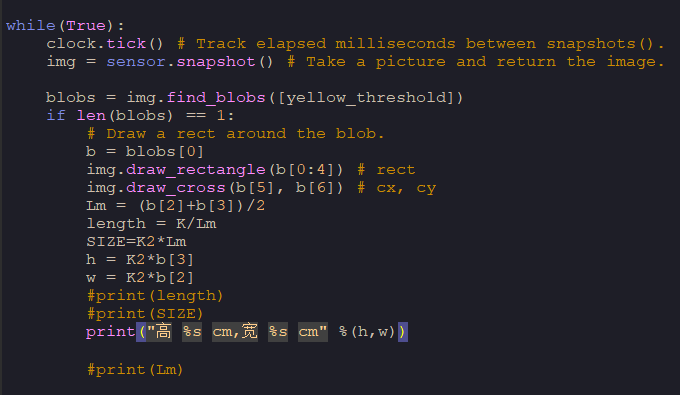




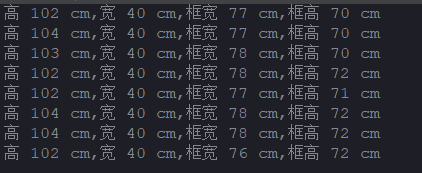


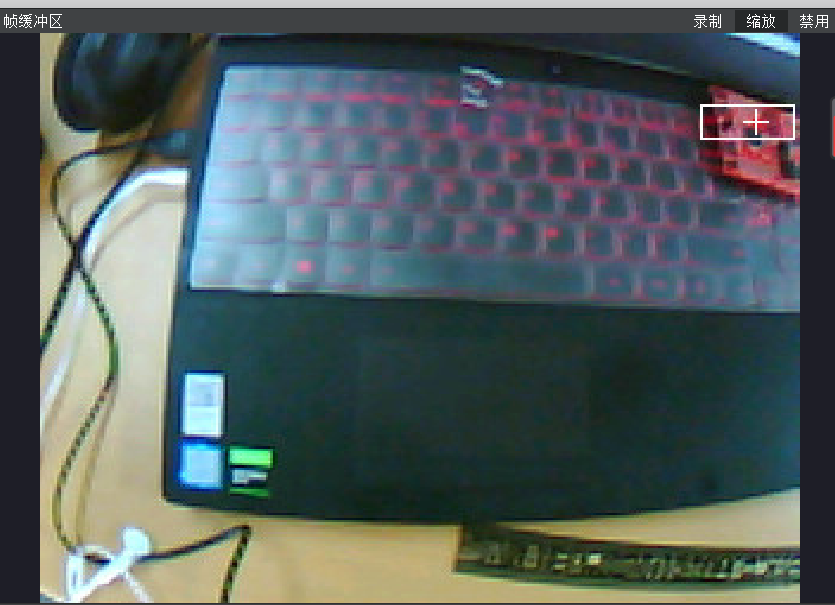
## 练3 物体大小识别

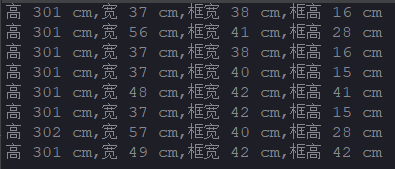














## 练4 openmv&32

Openmv

uart = UART(3,115200) #定义串口3变量

uart.init(115200, bits=8, parity=None, stop=1)#串口初始化设置

data = ustruct.pack(“<bbhhhhb”,

0x2C, #帧头1

0x12, #帧头2

int(cx1), # up sample by 4 #数据1

int(cx2),

int(cy), # up sample by 4 #数据2

int(cw), # up sample by 4 #数据1

int(ch), # up sample by 4 #数据2

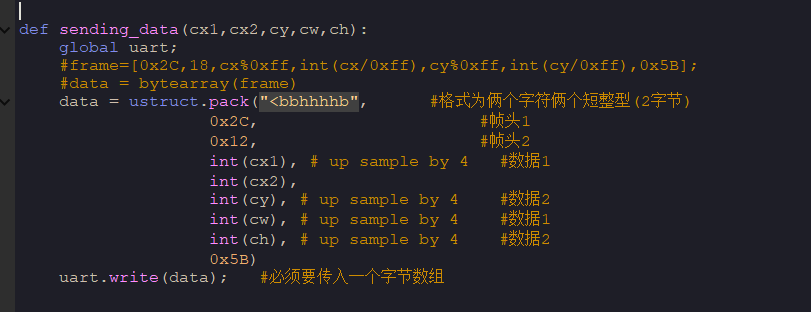
0x5B)

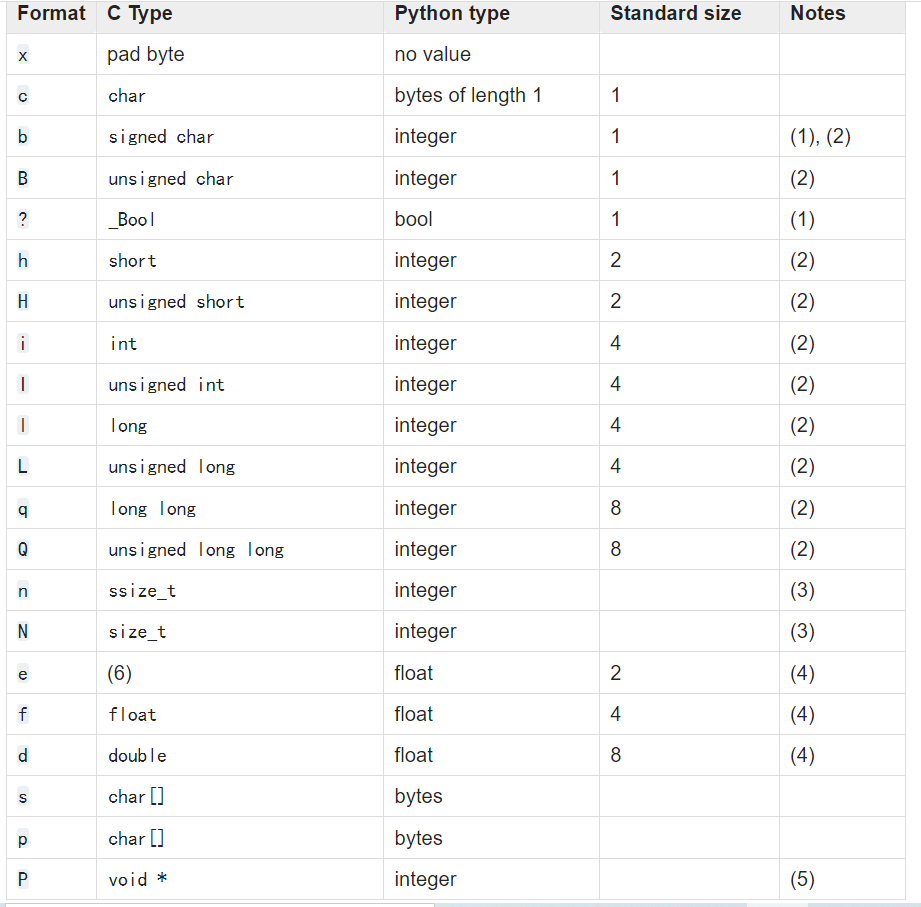
在串口等串行通信中，一次只能发一个字节，接收方也对应一次接收一个字节，有时一个字节并不能存放下我们采集到的数据，这时就需要多次传送单个字节。

<代表小端模式，指数据传输到另一方时数据存储的方式,小端模式指低位（字节/比特）放在低地址中，高位(字节/比特)放在高地址中，通过查阅资料得知ARM，intel,采用小端模式。

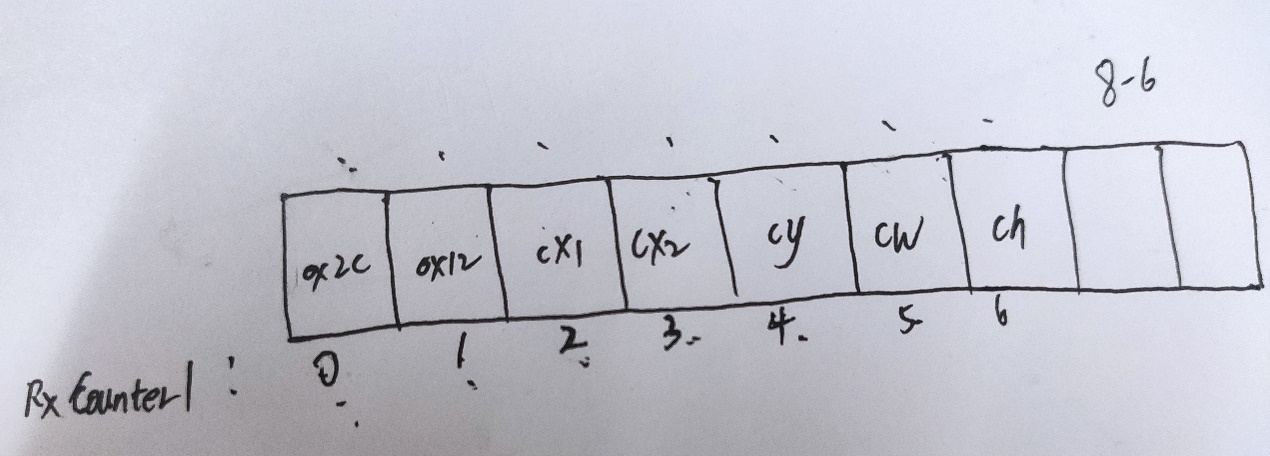
Openmv中帧率设置为QVGA 分辨率320\*240，Cx,Cy坐标设为短整型h,占两个字节，Cx坐标一个字节放不下，分两次传输低字节





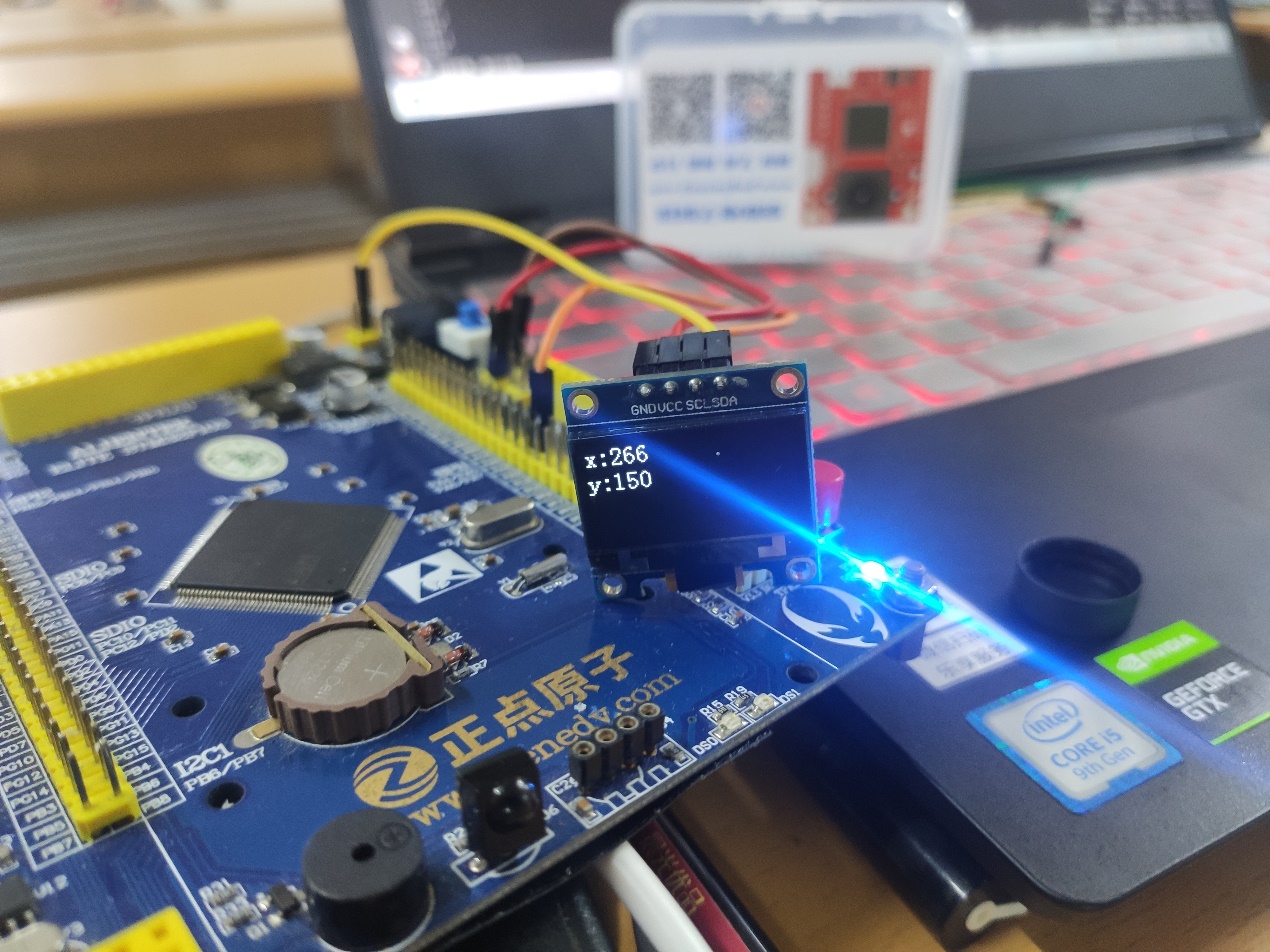


数据在stm32端储存格式如下：





Cx需要将传过来的2低字节数据进行组合，之后直接引用Cx.



最初代码来自CSDN大佬，最近的工作只是弄懂了大佬的代码，可以去下边的链接看大佬文章。

https://blog.csdn.net/lihaotian111/article/details/116384913?utm\_source=app&app\_version=4.8.0&code=app\_1562916241&uLinkId=usr1mkqgl919blen