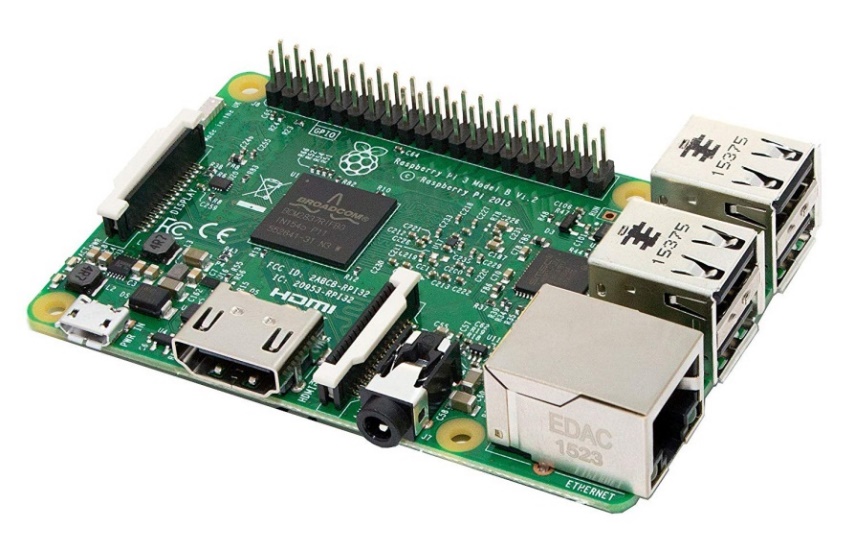
Отчёт по теме «СИСТЕМА ИЗМЕРЕНИЯ ПИТАТЕЛЬНЫХ ВЕЩЕСТВ В СИСТЕМЕ КОНТРОЛЯ И УПРАВЛЕНИЯ ЗАМКНУТОЙ ВОДНОЙ СРЕДОЙ»

**Одноплатный ПК Raspberry Pi 3 Model B**

Raspberry Pi 3 Model B является прямым наследником Raspberry Pi 2 Model B (Рисунок 1). Плата полностью совместима с предшественником, но наделена большей производительностью и новыми средствами коммуникации:

* 64-х битным четырёхъядерным процессором ARM Cortex-A53 с тактовой частотой 1,2 ГГц на однокристальном чипе Broadcom BCM2837;
* встроенными Wi-Fi 802.11n и Bluetooth 4.1.



**Рисунок 1. Одноплатный компьютер Raspberry Pi 3 Model B**

**Технические характеристики**

На Raspberry Pi 3 Model В установлен 64-х битный четырёхъядерный процессор ARM Cortex-A53 с тактовой частотой 1,2 ГГц на ядро в составе однокристальной платформы Broadcom BCM2837.

Данный чип обеспечивает прирост производительности на 50–60% в сравнении с Raspberry Pi 2 Model В и почти десятикратное преимущество перед первым Raspberry Pi. Благодаря этому одноплатный компьютер открывает ещё больше возможностей для «интернета вещей» и встраиваемых проектов.

Raspberry Pi 3 Model B имеет 1 ГБ оперативной памяти, но эта память делится с графической подсистемой.

Графический двухъядерный процессор VideoCore IV поддерживает стандарты OpenGL ES 2.0, OpenVG, MPEG-2, VC-1 и способен кодировать, декодировать и выводить Full HD-видео (1080p, 60 FPS, H.264 High-Profile).

**Технические характеристики**

* Процессор: 64-битный 4-ядерный ARM Cortex-A53 1,2 ГГц.
* Оперативная память: 1ГБ LPDDR2 SDRAM.
* Цифровой видеовыход: HDMI.
* Композитный выход: 3,5 мм (4 pin).
* USB порты: USB 2.0×4.
* Сеть: WiFi 802.11n, 10/100 Мб RJ45 Ethernet.
* Bluetooth: Bluetooth 4.1, Bluetooth Low Energy.
* Разъем дисплея: Display Serial Interface (DSI).
* Разъем видеокамеры: MIPI Camera Serial Interface (CSI-2).
* Карта памяти: MicroSD.
* Порты ввода-вывода: 40 шт.
* Габариты: 85x56x17 мм.

**Программное обеспечение OpenCV**

OpenCV – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков.

Назначение – утверждение общего стандартного интерфейса компьютерного зрения для приложений в этой области, способствование росту числа таких приложений и создания новых моделей использования PC.

Провозглашается также цель сделать платформы Intel привлекательными для разработчиков таких приложений за счёт дополнительного ускорения OpenCV с помощью Intel Performance Libraries (Сейчас включают IPP (низкоуровневые библиотеки для обработки сигналов, изображений, а также медиа-кодеки) и MKL (специальная версия LAPACK и FFTPack)). OpenCV способна автоматически обнаруживать присутствие IPP и MKL и использовать их для ускорения обработки.

Цветовая модель**—** термин, обозначающий абстрактную модель описания представления цветов в виде кортежей чисел, обычно из трёх или четырёх значений, называемых цветовыми компонентами или цветовыми координатами. Вместе с методом интерпретации этих данных множество цветов цветовой модели определяет цветовое пространство.

**Цветовая модель RGB**

В основе одной из наиболее распространенных цветовых моделей, называемой RGB моделью, лежит воспроизведение любого цвета путем сложения трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). Каждый канал - R, G или B имеется свой отдельный параметр, указывающий на количество соответствующей компоненты в конечном цвете. Например: (255, 64, 23) – цвет, содержащий сильный красный компонент, немного зелёного и совсем немного синего. Естественно, что этот режим наиболее подходит для передачи богатства красок окружающей природы. Но он требует и больших расходов, так как глубина цвета тут наибольшая – 3 канала по 8 бит на каждый, что дает в общей сложности 24 бита.

Поскольку в RGB модели происходит сложение цветов, то она называется аддитивной(additive). Цветовым пространством RGB модели является единичный куб.

**Цветовая модель HSV**

HSV (англ. Hue, Saturation, Value — тон, насыщенность, значение) цветовая модель, в которой координатами цвета являются:

* Hue — цветовой тон, (например, красный, зелёный или сине-голубой). Варьируется в пределах 0—360°, однако иногда приводится к диапазону 0—100 или 0—1.
* Saturation — насыщенность. Варьируется в пределах 0—100 или 0—1. Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют чистотой цвета. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.
* Value (значение цвета) или Brightness — яркость. Также задаётся в пределах 0—100 или 0—1.

В цветовом пространстве модели HSV (Hue, Saturation, Value), иногда называемой HSB (Hue, Saturation, Brightness), используется цилиндрическая система координат, а множество допустимых цветов представляет собой шестигранный конус, поставленный на вершину.

Основание конуса представляет яркие цвета и соответствует V = 1. Однако цвета основания V = 1 не имеют одинаковой воспринимаемой интенсивности. Тон (H) измеряется углом, отсчитываемым вокруг вертикальной оси OV. При этом красному цвету соответствует угол 0°, зелёному – угол 120° и т. д. Цвета, взаимно дополняющие друг друга до белого, находятся напротив один другого, т. е. их тона отличаются на 180°. Величина S изменяется от 0 на оси OV до 1 на гранях конуса.

Конус имеет единичную высоту (V = 1) и основание, расположенное в начале координат. В основании конуса величины H и S смысла не имеют. Белому цвету соответствует пара S = 1, V = 1. Ось OV (S = 0) соответствует ахроматическим цветам (серым тонам).

Процесс добавления белого цвета к заданному можно представить, как уменьшение насыщенности S, а процесс добавления чёрного цвета – как уменьшение яркости V.

**Цветовая модель** CMYK

CMYK - Cyan, Magenta, Yellow, Key color - субтрактивная (subtract, англ. - вычитать) схема формирования цвета, используемая в полиграфии для стандартной триадной печати. Обладает меньшим, в сравнении с RGB, цветовым охватом.

CMYK называют субстрактивной моделью потому, что бумага и прочие печатные материалы являются поверхностями, отражающими свет. Удобнее считать, какое количество света отразилось от той или иной поверхности, нежели сколько поглотилось. Таким образом, если вычесть из белого три первичных цвета - RGB, мы получим тройку дополнительных цветов CMY. «Субтрактивный» означает «вычитаемый» — из белого вычитаются первичные цвета.

Key Color (черный) используется в этой цветовой модели в качестве замены смешению в равных пропорциях красок триады CMY. Дело в том, что только в идеальном варианте при смешении красок триады получается чистый черный цвет. На практике же он получится, скорее, грязно-коричневым - в результате внешних условий, условий впитываемости краски материалом и не идеальности красителей. К тому же, возрастает риск неприводки в элементах, напечатанных черным цветом, а также переувлажнения материала (бумаги).

**Цветовая модель Lab**

В цветовом пространстве Lab значение светлоты отделено от значения хроматической составляющей цвета (тон, насыщенность). Светлота задана координатой L (изменяется от 0 до 100, то есть от самого темного до самого светлого), хроматическая составляющая — двумя декартовыми координатами a и b. Первая обозначает положение цвета в диапазоне от зеленого до пурпурного, вторая — от синего до желтого.

В отличие от цветовых пространств RGB или CMYK, которые являются, по сути, набором аппаратных данных для воспроизведения цвета на бумаге или на экране монитора (цвет может зависеть от типа печатной машины, марки красок, влажности воздуха на производстве или производителя монитора и его настроек), Lab однозначно определяет цвет. Поэтому Lab нашел широкое применение в программном обеспечении для обработки изображений в качестве промежуточного цветового пространства, через которое происходит конвертирование данных между другими цветовыми пространствами (например, из RGB сканера в CMYK печатного процесса). При этом особые свойства Lab сделали редактирование в этом пространстве мощным инструментом цветокоррекции.

Благодаря характеру определения цвета в Lab появляется возможность отдельно воздействовать на яркость, контраст изображения и на его цвет. Во многих случаях это позволяет ускорить обработку изображений, например, при допечатной подготовке. Lab предоставляет возможность избирательного воздействия на отдельные цвета в изображении, усиления цветового контраста, незаменимыми являются и возможности, которые это цветовое пространство предоставляет для борьбы с шумом на цифровых фотографиях.

**Вывод**

Так как входными данными для системы компьютерного зрения OpenCV являются изображения в формате RGB, то эту цветовую модель мы используем по умолчанию.

Однако с помощью цветовой модели RGB довольно сложно однозначно определить цвет, поэтому необходимо перевести данные из цветовой модели RBG в другую.

CMYK обладает меньшим, в сравнении с RGB, цветовым охватом, поэтому он нам не подходит.

LAB используется в качестве промежуточного цветового пространства, через которое происходит конвертирование данных между другими цветовыми пространствами.

Таким образом, для реализации системы цветовая модели HSV подходит наилучшим образом, так как оттенок цвета (цветовой тон) задаётся только одной координатой - hue, что позволяет легче отсеивать только нужные цвета с картинки. Кроме того, с помощью координат saturation и value можно минимизировать или полностью исключить влияние слабой освещённости, затемнения или теней в рабочей области камеры.

**Практическая часть**

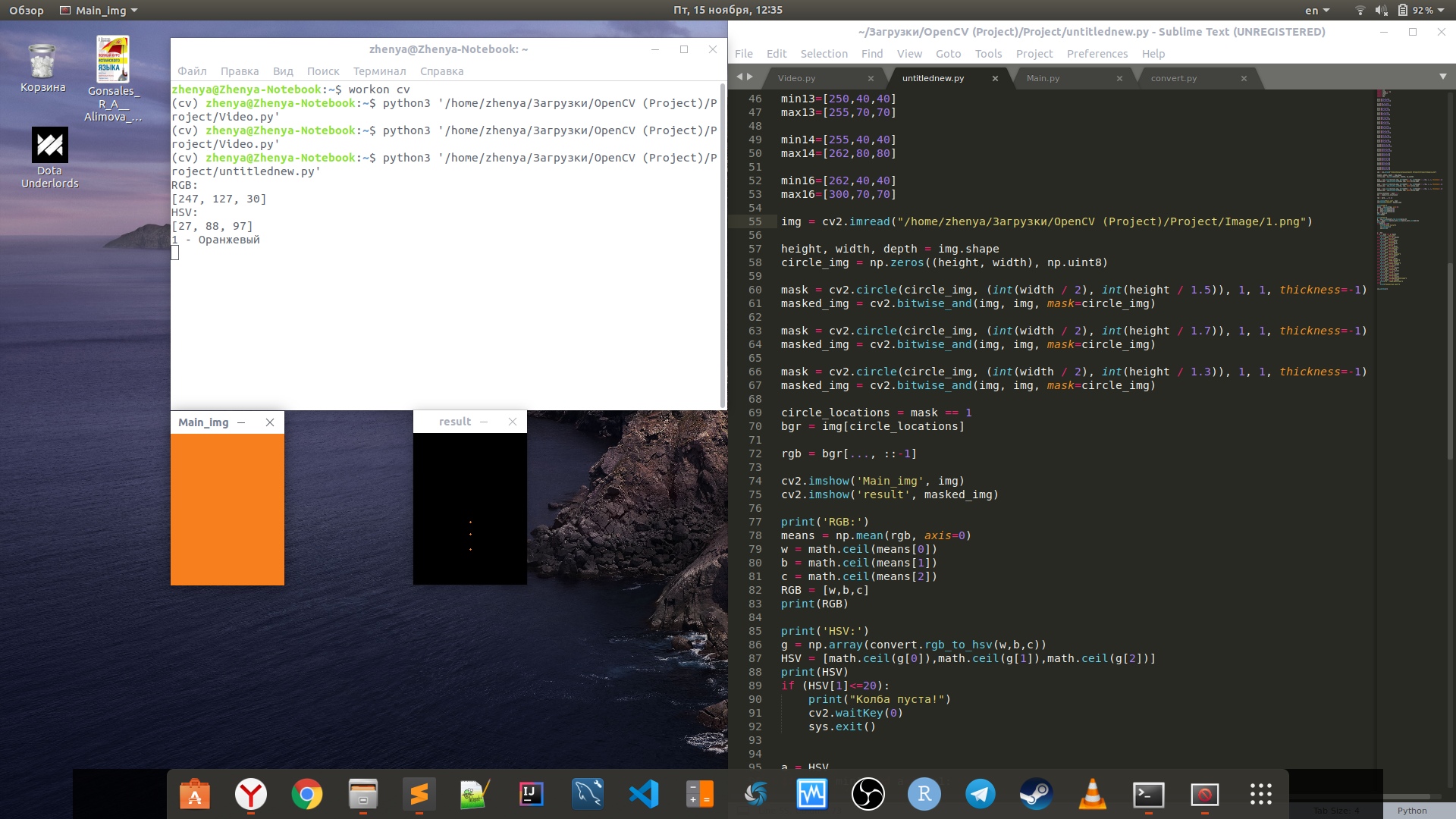
После изучения теоретической части были проведены практические тесты. В качестве реактивов для окрашивания воды были использованы краски акварель. Примеры изображений, полученных при помощи модуля Video.py представлены на рисунке 2.



**Рисунок 2**

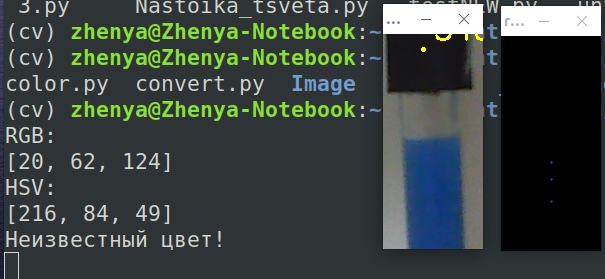
Затем, данные изображения подаются в модуль Main.py в качестве входных данных. Происходит Перевод значения из цветовой модели RGB в цветовую модель HSV. Зачет выходной массив данных в формате HSV сравнивается с эталонным (заранее вычисленным) значение и в консоли выводиться результат.

Пример работы программы на идеальном (чистом) цвете представлен на рисунке 3.



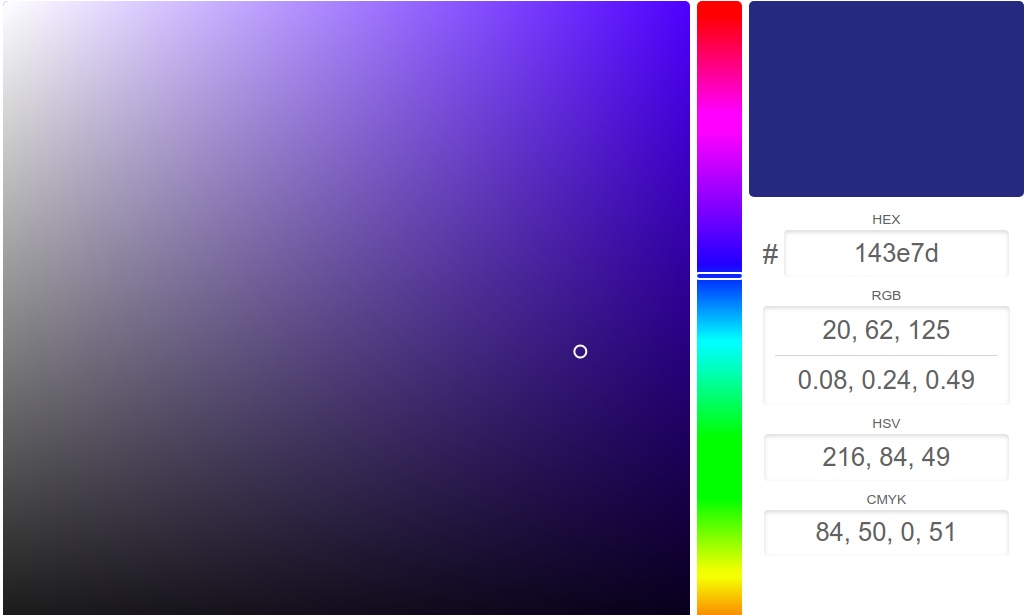
**Рисунок 3**

Примеры работы программы в реальных условиях представлены на рисунке 4.



**Рисунок 4**

К сожалению алгоритм не выдал однозначного ответа по тому, какого цвета жидкость в пробирке, однако если вспомнить специфику представления цвета в формате HSV, то можно сказать что Hue (цветовой тон) программа определила довольно точно. Проблема в насыщенности и яркости цвета, поскольку в алгоритм занесены идеальные показатели цвета, в конкретной ситуации она не смогла дать однозначный ответ.



**Рисунок 5**

Вывод: таким образом для корректной работы программы необходимо создать идеальные условия освещения и видео съемки колбы с реагентом. Достичь этого можно путем создания фото-будки, замкнутой системы со свои освещением. Также необходимо подкорректировать цветовые значения, которые используются для распознавания цвета. Коррекцию необходимо производить в тех условиях, в которых будет производиться съемка в бедующем.

Реализация программного модуля Сonvert.py на языке программирования Python3, который используется для перевода значений из цветовой модели RGB в цветовую модель HSV:

def rgb\_to\_hsv(r, g, b):

# R, G, B values are divided by 255

# to change the range from 0..255 to 0..1:

r, g, b = r / 255.0, g / 255.0, b / 255.0

# h, s, v = hue, saturation, value

cmax = max(r, g, b) # maximum of r, g, b

cmin = min(r, g, b) # minimum of r, g, b

diff = cmax-cmin # diff of cmax and cmin.

# if cmax and cmax are equal then h = 0

if cmax == cmin:

h = 0

# if cmax equal r then compute h

elif cmax == r:

h = (60 \* ((g - b) / diff) + 360) % 360

# if cmax equal g then compute h

elif cmax == g:

h = (60 \* ((b - r) / diff) + 120) % 360

# if cmax equal b then compute h

elif cmax == b:

h = (60 \* ((r - g) / diff) + 240) % 360

# if cmax equal zero

if cmax == 0:

s = 0

else:

s = (diff / cmax) \* 100

# compute v

v = cmax \* 100

return h, s, v

Код программного модуля Video.py:

import cv2  
import numpy as np  
  
cv2.namedWindow( "result" )  
  
cap = cv2.VideoCapture(0)  
cap.set(cv2.CAP\_PROP\_FPS, 30)  
cap.set(cv2.CAP\_PROP\_FRAME\_WIDTH, 600)   
cap.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, 800)  
hsv\_min = np.array((0, 0,0), np.uint8)  
hsv\_max = np.array((360, 360, 15), np.uint8)  
  
color\_yellow = (0,255,255)  
i=0  
  
while True:  
flag, img = [cap.read](https://vk.com/away.php?to=http%3A%2F%2Fcap.read&cc_key=" \t "_blank)()  
  
hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV )  
thresh = cv2.inRange(hsv, hsv\_min, hsv\_max)  
  
moments = cv2.moments(thresh, 1)  
dM01 = moments['m01']  
dM10 = moments['m10']  
dArea = moments['m00']  
  
if dArea > 100:  
x = int(dM10 / dArea)  
y = int(dM01 / dArea)  
[cv2.circle](https://vk.com/away.php?to=http%3A%2F%2Fcv2.circle&cc_key=)(img, (x, y), 1, color\_yellow, 2)  
cv2.putText(img, "%d-%d" % (x,y), (x+10,y-10),   
cv2.FONT\_HERSHEY\_SIMPLEX, 1, color\_yellow, 2)  
  
cv2.imshow('result', img)  
  
ret, frame = [cap.read](https://vk.com/away.php?to=http%3A%2F%2Fcap.read&cc_key=" \t "_blank)()  
key=cv2.waitKey(30)  
if key==ord('q'):  
break  
if key==ord('c'):  
i+=1  
cropped = img[y-15:y+200, x-40:x+60]  
cv2.imshow('resultnew', cropped)  
cv2.imwrite('/home/zhenya/Загрузки/OpenCV (Project)/Project/Image/image'+str(i)+'.png', cropped)  
  
cap.release()  
cv2.destroyAllWindows()

Код программного модуля Main.py

import cv2  
import numpy as np   
import convert  
import math  
import sys  
import color  
  
img = cv2.imread("/home/zhenya/Загрузки/OpenCV (Project)/Project/Image/image6.png")  
  
height, width, depth = img.shape  
circle\_img = np.zeros((height, width), np.uint8)  
  
mask = [cv2.circle](https://vk.com/away.php?to=http%3A%2F%2Fcv2.circle&cc_key=)(circle\_img, (int(width / 2), int(height / 1.5)), 1, 1, thickness=-1)  
masked\_img = cv2.bitwise\_and(img, img, mask=circle\_img)  
  
mask = [cv2.circle](https://vk.com/away.php?to=http%3A%2F%2Fcv2.circle&cc_key=)(circle\_img, (int(width / 2), int(height / 1.7)), 1, 1, thickness=-1)  
masked\_img = cv2.bitwise\_and(img, img, mask=circle\_img)  
  
mask = [cv2.circle](https://vk.com/away.php?to=http%3A%2F%2Fcv2.circle&cc_key=)(circle\_img, (int(width / 2), int(height / 1.3)), 1, 1, thickness=-1)  
masked\_img = cv2.bitwise\_and(img, img, mask=circle\_img)  
  
circle\_locations = mask == 1  
bgr = img[circle\_locations]  
  
rgb = bgr[..., ::-1]  
  
cv2.imshow('Main\_img', img)  
cv2.imshow('result', masked\_img)  
  
print('RGB:')  
means = np.mean(rgb, axis=0)  
r = math.ceil(means[0])  
g = math.ceil(means[1])  
b = math.ceil(means[2])  
RGB = [r,g,b]  
print(RGB)  
  
print('HSV:')  
g = np.array(convert.rgb\_to\_hsv(r,g,b))  
h = math.ceil(g[0])  
s = math.ceil(g[1])  
v = math.ceil(g[2])  
HSV = [h,s,v]  
print(HSV)  
if (HSV[1]<=20):  
print("Колба пуста!")  
cv2.waitKey(0)  
sys.exit()  
  
color.color(h,s,v)