

# OpenMV IDE 自带例程解析与练习

作者：付坤 版本：第一版 发布时间：2025 年 3 月 4 日

OpenMV IDE 自带许多例程，这些例程是基础的、易被理解的，具有非常好的参考价值，是读者零基础入门的首要学习材料。本文会对每一个例程仔细剖析，针对例程中的关键语法设计几个案例用来练习，读者在掌握每个例程的同时也具备举一反三的拓展能力。在开始学习本文的所有案例之前，您需要注意的是：

- 本文的目标读者为已经完成九年义务教育，并接触过任何一种计算机编程语言的朋友；
- 在个人电脑上已经安装好 OpenMV IDE；
- 装备好一款支持 OpenMV 生态的硬件，我们推荐自己 DIY 的产品，跟官方原版相比性能相同，价格实惠很多。

## 一、Hello word

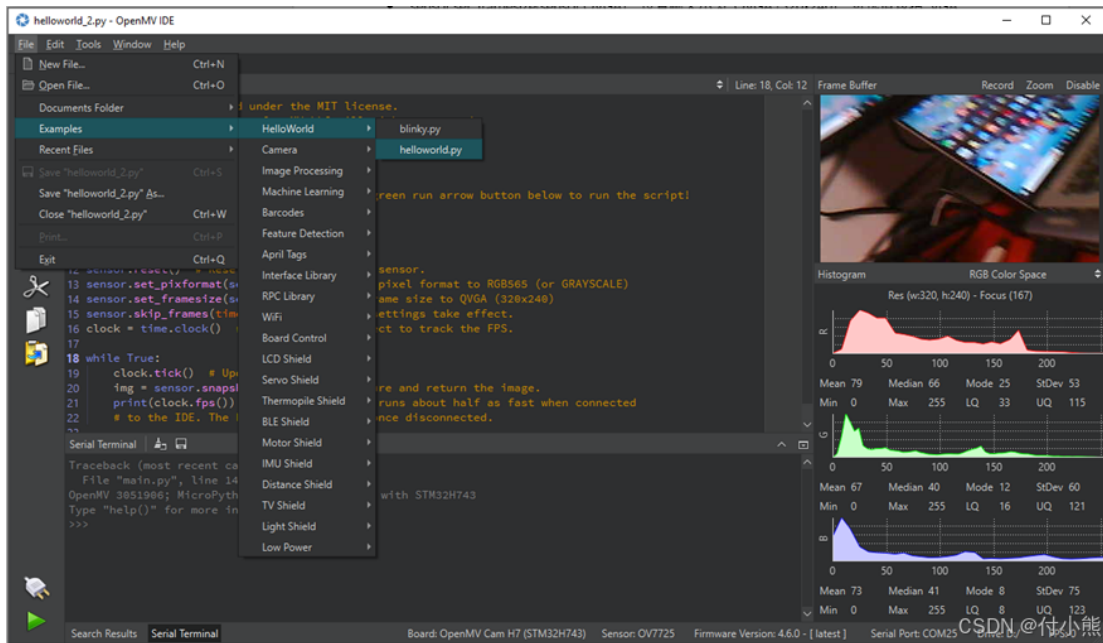
### 1.1 helloworld.py

#### 1.1.1 演示

本次演示使用的设备如下图所示



将 FKMV 相机通过附赠的 USB 数据线连接到电脑 USB 口，点击左下角的 USB 连接标志。成功连接后，找到路径为 File/Examples/Helloworld 程序路径，左键点击 helloworld.py 代码，IDE 会自动加载此代码到工作区。如未成功连接，以上程序路径是灰色不可选状态。在连接过程中会跳出让您购买许可证的提示，我们建议您根据自身情况，合理支持开源社区的工作。我们这里为了演示方便，就先点击“否”了。



### 1.1.2 代码

代码路径：File/Examples/Helloworld/ helloworld.py

```
# This work is licensed under the MIT license.
# Copyright (c) 2013-2023 OpenMV LLC. All rights reserved.
# https://github.com/openmv/openmv/blob/master/LICENSE
#
# Hello World Example
#
# Welcome to the OpenMV IDE! Click on the green run arrow button below to run the script!

import sensor
import time

sensor.reset() # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
sensor.skip_frames(time=2000) # Wait for settings take effect.
clock = time.clock() # Create a clock object to track the FPS.

while True:
    clock.tick() # Update the FPS clock.
    img = sensor.snapshot() # Take a picture and return the image.
    print(clock.fps()) # Note: OpenMV Cam runs about half as fast when connected
    # to the IDE. The FPS should increase once disconnected.
```

这段代码是一个简单的 OpenMV 示例脚本，用于展示如何使用 OpenMV 的图像传感器库进行基本操作。该程序初始化摄像头模块，并设置像素格式和帧大小。每次循环中拍摄一张图片，并打印当前帧率。是一个“Hello World”级别的 OpenMV 示例，非常适合入门学习。

一段工程源码，我会将其分成四部分，以这段工程源码为例，第一部分是前头带“#”号的、绿色的文字，这是“注释”给人看的部分，有的时候代码写的很长，第二天都不记得自己写的是啥，自己都看不懂了，类似备注。另花花绿绿的部分是真正的代码，是给设备看的，咱先不关心为啥设备能看懂，您先记下设备就是依靠这些代码执行一些动作的，就像  $1+1=2$  一样，先不管为什么。因为有的朋友一旦不理解某件事，就会抓狂，不自觉的拒绝新知识。您若非得较真，建议看一下我录制的《从零搭建一台计算机》系列视频，或许您会有收获。下一节会对“Hello World”代码做详细分析。

### 1.1.3 代码解析

代码中涉及到的一些函数，在这里有所提及 [MicroPython libraries](#)，建议有时间可以过一下，心里有个数，知道这个产品可以做一些什么事情。

#### 1.1.3.1 导入库

```
import sensor
import time
```

- **sensor**: OpenMV 提供的库，用于控制摄像头模块的行为，例如设置分辨率、拍照等。
- **time**: 标准 Python 库，用于管理时间与帧率。

`import` 和 `sensor` 及 `time` 的颜色不同，指示的是“import”是个有特殊功能的词，叫“关键词”，“import sensor”的功能是高速设备，我现在要用“sensor”这个库，在代码中用到的这个库里的一些方法（函数），你别不知道是啥，不知道去哪找。类似 c 语言中的“include”。

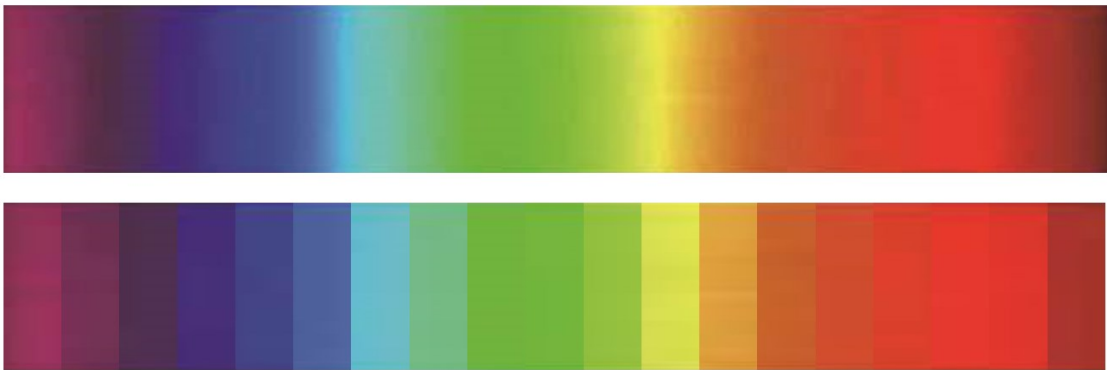
#### 1.1.3.2 初始化传感器

```
sensor.reset() # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
sensor.skip_frames(time=2000) # Wait for settings take effect.
clock = time.clock() # Create a clock object to track the FPS.
```

- `sensor.reset()`: 重置摄像头模块，初始化传感器。

- `sensor.set_pixformat(sensor.RGB565)`: 设置像素格式为 RGB565（即每个像素使用 16 位存储颜色）。可选值: `sensor.GRAYSCALE`（灰度图）或其他支持的格式。
- `sensor.set_framesize(sensor.QVGA)`: 设置帧大小为 QVGA (320x240)。可选值包括 VGA、QQVGA 等。
- `sensor.skip_frames(time=2000)`: 跳过若干帧，等待 2 秒以使设置生效。
- `time.clock()`: 创建一个时钟对象，用于跟踪帧率（FPS）。

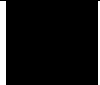

我们知道，光的三原色是红(RED)绿(GREEN)蓝(BLUE)，即首字母 RGB。我们这台相机感光元件（OV7725 CMOS），接收的是环境连续光，使用 ADC 采样后输出的是数字非连续光。下图仅作示意，顶端图片用来表征环境连续光，底部图片用来表示采样后的非连续光，采样的间隔越小，精度越高。











RGB 不同的组合决定了颜色，RGB565 是人为将红色亮度分成 25 共 32 个离散级别，将绿色分成 26 共 64 个离散级别，将蓝色分成 25 共 32 个离散级别，总的排列组合为  $25 \times 26 \times 25 = 65536$  种颜色。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0

上表中，一个 RGB565 颜色格式是由 16 个二进制数字决定，8 个二进制数为一个字节，一个 RGB565 由两个字节表示。最低位为 B0，即为蓝色 B 分量的最低位。下表是赋予 RGB565 不同的值，对应的不同颜色色块，您可以登陆网站 [RGB565 Color Picker](#)，自己动手尝试一下

RGB565	色块
00000 000000 00000	
00000 000000 11111	

00000 111111 00000	
00000 111111 11111	
11111 000000 00000	
11111 000000 11111	
11111 111111 00000	
11111 111111 11111	
10010 011100 10101	
01100 101011 01011	

这里除了 RGB565 还有一个灰度格式，这个格式输出白色到黑色的灰度图像 0~255，仅需一个字节。相比 RGB565 输出一个像素需要 2 个字节，灰度图像更快。

以 OV7725 为例，其 CMOS 参数为  $640 \times 480$ （VGA）像素阵列，约 30 万像素，VGA 全分辨率条件下能达到 60fps（每秒拍摄 60 张图像）。设置帧大小意味着选择  $640 \times 480$  这个像素矩阵中的某一部分输出，如设置  $320 \times 240$ （QVGA）输出，可达 120fps，适用于需要更高刷新率的场景。

### 1.1.3.3 主循环

```
while True:
    clock.tick() # Update the FPS clock.
    img = sensor.snapshot() # Take a picture and return the image.
    print(clock.fps()) # Note: OpenMV Cam runs about half as fast when connected
                       # to the IDE. The FPS should increase once disconnected.
```

- `clock.tick()`：记录每次循环的时间间隔，更新 FPS（每秒帧数）。
- `sensor.snapshot()`：拍摄一帧图像并返回图像对象（可以进行后续处理）。
- `print(clock.fps())`：打印实时的帧率。注意：当 OpenMV 摄像头连接到 IDE 时，性能会降低，断开连接后帧率会提高。

While True 后面紧跟的缩进代码，表示循环，这三段代码循环执行。在 `time` 库中我们能够找到跟 `clock` 有关的方法，`clock = time.clock()` 先创立了一个对象，将

time.clock()的特性赋予或者授权给 clock，clock.tick()是开始计时，clock.fps()用来获取计时时间，计算后返回帧率。

### 1.1.4 练习

#### 1.1.4.1 修改像素格式和分辨率输出

下面这段代码 [sensor.set\\_pixformat\(\)](#)和 [sensor.set\\_framesize\(\)](#)两个方法，我们找到位置，尝试几个不同的配置

Sensor.GRAYSCALE	8bits 灰度
Sensor.RGB565	16bits 彩色
Sensor.BAYER	8bits bayer 原像素彩色
Sensor. <a href="#">YUV422</a>	每像素 16 位（8 位 Y1，8 位 U，8 位 Y2，8 位 V 等）
Sensor.jpeg	支持 OV2640/OV5640

#### 1.1.4.2 修改稳定时间

在其官方描述中可以了解到 [sensor.skip\\_frames\(n=int, time=int\)](#)其实有两个参数，n 表示跳过多少帧，time 表示跳过多久。若 n 和 time 缺省时，执行跳过 300ms，即 sensor.skip\_frames()= sensor.skip\_frames(time=300)；若 n 和 time 仅有一个参数被赋值时，执行跳过帧数（n），或执行跳过一段指定的时间（time）；若 n 和 time 都被赋值时，执行跳过耗时最小的那个参数。

不管跳过多少帧，还是跳过多长时间，目的总是稳定相机后输出图像或图像处理的结果。在实际应用中，可能需要在满足相机稳定的最短时间内，迅速执行任务，这就需要您在真实的使用环境下，调试以获得合理的最小稳定时间。

#### 1.1.4.3 脱机运行

FKMV 设备支持脱机运行，Tools->save open script to OpenMV Cam (as main.py) 将 IDE 中的代码烧录到设备中，我们会在存储设备中找到这个代码，之后只要通电就能运行了。为了方便演示，我们使用 lcd 的 demo 例程（先不用管这些代码是什么意思），我们外接 TFTLCD 展示。