



视觉传感器

MicroPython 编程指南



V 0.8.1

2019.06.29

目 录

免责声明和版权公告.....	2
1 配置 Mu Editor.....	3
2 连接 MU 视觉传感器与 Micro:bit.....	4
3 API 使用说明.....	5
3.1 初始化传感器.....	5
3.2 开启算法.....	5
3.3 设置算法性能.....	6
3.4 开启摄像头高帧率模式.....	7
3.5 设置摄像头白平衡.....	8
3.6 设置摄像头数码变焦.....	9
3.7 恢复传感器默认设置.....	9
3.8 重启传感器.....	9
3.9 LED 灯光设置.....	9
3.10 获取算法识别结果.....	10
技术支持.....	12
修订历史.....	13

免责声明和版权公告

- 本手册中的信息适用于摩图科技公司所生产的小 MU 视觉传感器 III 代，敬请留意摩图科技官网 <http://www.morpx.com> 以便获取传感器固件与库函数的最新版本，版本更新不另行通告。
- 请仔细阅读和理解本手册中的信息，不正确的使用可能导致产品无法正常工作，检测效果变差，甚至产品损坏。
- 私自维修或改装传感器上的电子元件造成损坏，摩图科技将不予以保修。
- 本手册中所提及的技术方案、视觉算法、通讯协议均为摩图科技自主研发，具有知识产权，任何组织或个人不得拷贝、抄袭、剽窃摩图科技的技术成果，对于任何侵权行为，摩图科技将采取法律手段予以维权。
- MORPX 是杭州摩图科技有限公司的注册商标，MU 是小 MU 视觉传感器的注册商标。文本或图片中涉及到的所有商标（名称与图案）归属于其持有者，特此声明。

1. 配置 Mu Editor

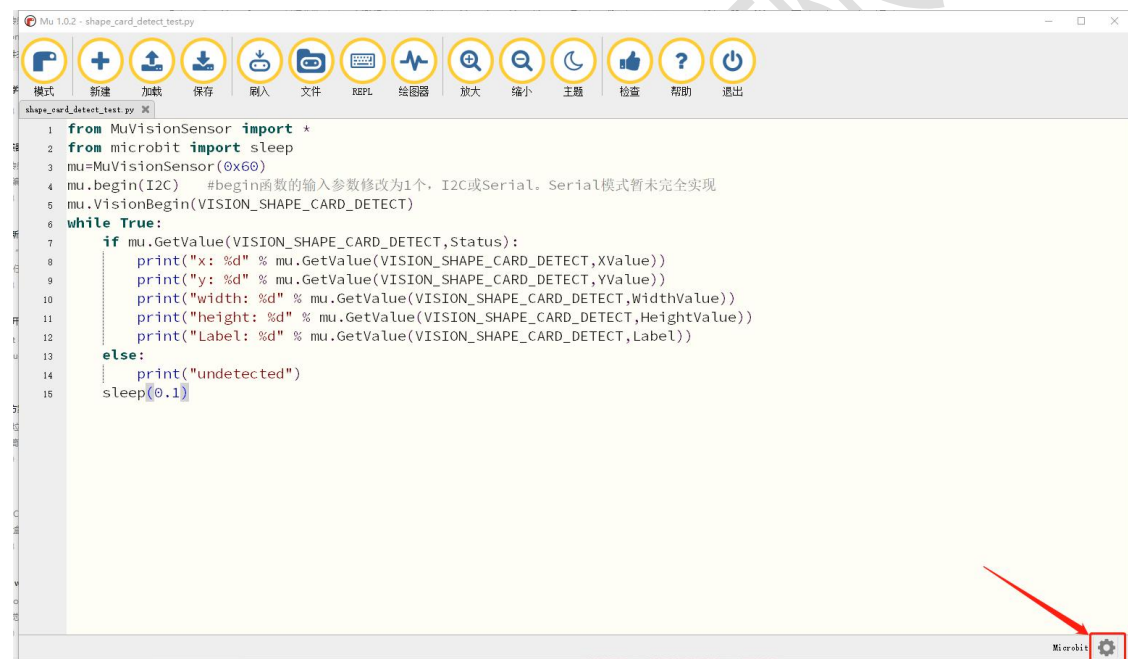
Mu Editor 是一款具有友好 GUI 界面的 MicroPython 集成开发工具，包含了代码编辑、烧录、REPL 终端、串口绘图器等功能。通过 Micro:bit 主板控制小 MU 视觉传感器，需要使用包含了 MuVisionSensor 传感器的 MicroPython 固件，请按以下步骤进行设置：

(1) 下载固件：

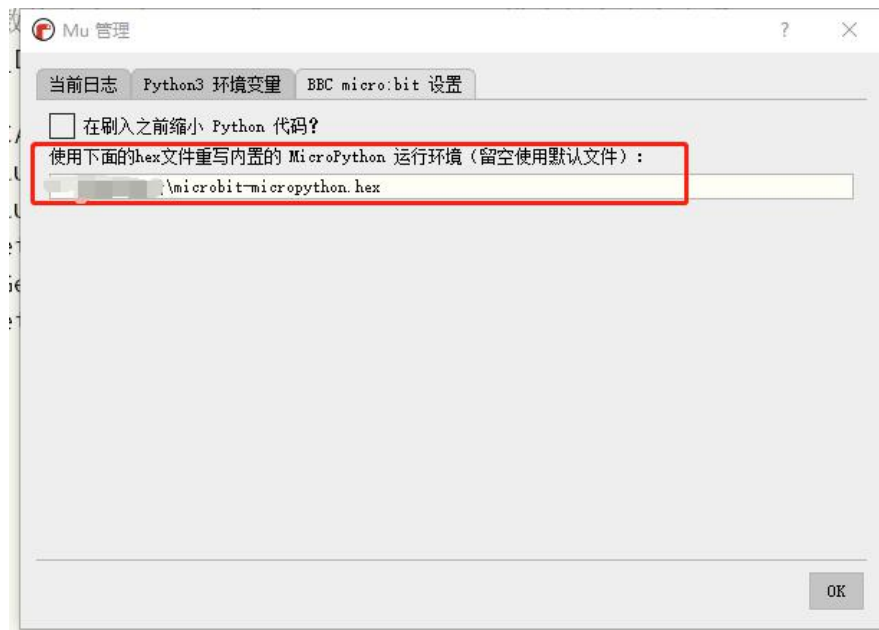
GitHub: <https://github.com/mu-opensource/MuVisionSensorIII-MicroPython>

官网: <http://mai.morpx.com/page.php?a=sensor-support>

(2) 下载安装 Mu Editor，打开后点击界面右下角的设置图标



(3) 在 Micro:bit 设置中，添加“microbit-micropython.hex”文件的路径



(4) 导入传感器

```
>>>from MuVisionSensor import *
```

导入传感器后即可使用 MuVisonSensor 类中的所有公开 API

*MuVisionSensor 传感器中关键字的自动补全仅在 REPL 模式下可用

```
BBC micro:bit REPL

MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; micro:bit v1.0.1 with nRF51822
Type "help()" for more information.
>>>
>>> from MuVisionSensor import *
```

2. 连接 MU 视觉传感器与 Micro:bit



*目前仅支持 I2C 模式

将传感器左侧输出模式拨码开关 1 拨至下方，2 拨至上方；

将传感器输出接口 SDA (1) 口和 SCL (2) 口接至 Micro:bit 对应的 SDA 口 (P20) 与 SCL 口 (P19) ；

修改传感器的 I2C 地址，将传感器的地址选择拨码开关拨至对应位置（默认地址 0x60 则 1、2 都在下方；不推荐修改此设置）；

3. API 使用说明

3.1 初始化传感器

第一步调用 `MuVisionSensor(adress)` 创建一个对象，并指定传感器地址，指定的地址要与地址选择拨码开关的设置保持一致

第二步是调用 `begin()` 函数，启动传感器

```
1. #example:
2. from MuVisionSensor import *
3. mu = MuVisonSensor(0x60) #创建对象，地址可指定为 0x60~0x63
4. mu.begin() #完成初始化
```

3.2 开启算法

API: `MuVisionSensor.VisionBegin(vision_type)`

目前支持的 `vision_type` 有：

<code>VISION_COLOR_DETECT</code>	颜色检测
<code>VISION_COLOR_RECOGNITION</code>	颜色识别

VISION_BALL_DETECT	球体检测
VISION_BODY_DETECT	人体检测
VISION_SHAPE_CARD_DETECT	形状卡片检测
VISION_TRAFFIC_CARD_DETECT	交通卡片检测
VISION_NUM_CARD_DETECT	数字卡片检测
VISION_ALL	开启所有算法

```
1. #example:
2. from MuVisionSensor import *
3. ... #省略初始化过程
4. mu.VisionBegin(VISION_COLOR_DETECT)
5. #开启颜色检测算法
6. mu.VisionBegin(VISION_SHAPE_CARD_DETECT | VISION_BALL_DETECT)
7. #同时开启形状卡片检测和球体检测算法
```

3.3 设置算法性能

API: `MuVisionSensor.VisionSetLevel(vision_type, level)`

可选的 vision_type 同上

可选的 level 有：

LevelDefault	默认
LevelSpeed	速度优先
LevelBalance	平衡
LevelAccuracy	准确性优先

1. #example:
2. from MuVisionSensor import *
3. ... #省略初始化过程
4. mu.VisionSetLevel(VISION_BALL_DETECT, LevelSpeed)
5. #将球体检测算法的性能设置为速度优先

获取算法性能

API: MuVisionSensor.VisionSetLevel(vision_type)

返回值 0~3 代表四种算法性能

1. #example:
2. from MuVisionSensor import *
3. ... #省略初始化过程
4. if mu.VisionSetLevel(VISION_BALL_DETECT) is LevelSpeed:
5. print("当前球体检测算法性能为速度优先模式")

3.4 开启摄像头高帧率模式 (*识别速度增加, 同时功耗增加)

API: MuVisionSensor.CameraSetFPS(mode)

可选的 mode 有:

FPSNormal

正常模式

FPSHigh

高帧率模式

1. #example:
2. from MuVisionSensor import *

3. ... #省略初始化过程

4. mu.CameraSetFPS(FPSHigh)

5. #开启高帧率模式

获取摄像头帧率模式

API:MuVisionSensor.CameraGetFPS()

返回值为 0(FPSNormal)或 1(FPSHigh)

3.5 设置摄像头白平衡 (*调节因为外界光源变化而引起的图像偏色)

API:MuVisionSensor.CameraSetAwb(mode)

可选的 mode 有:

AutoWhiteBalance 自动白平衡

LockWhiteBalance 锁定白平衡

WhiteLight 白光模式

YellowLight 黄光模式

获取摄像头白平衡模式

API:MuVisionSensor.CameraGetAwb()

返回值为 0~3, 对应 4 种白平衡模式

1. #example:

2. from MuVisionSensor import *

3. ... #省略初始化过程

4. if mu.CameraGetAwb() is AutoWhiteBalance:

5. print("当前白平衡模式为自动")

3.6 设置摄像头数码变焦

API:MuVisionSensor.CameraSetZoom(mode)

可选的 mode 有：

ZoomDefault	默认
Zoom1	变焦模式 1
Zoom2	变焦模式 2
Zoom3	变焦模式 3
Zoom4	变焦模式 4
Zoom5	变焦模式 5

获取摄像头变焦模式

API:MuVisionSensor.CameraGetZoom()

返回值为 0~5，对应 6 种白平衡模式

3.7 恢复传感器默认设置（关闭所有算法，重置所有硬件设置）

API:MuVisionSensor.SensorSetDefault()

3.8 重启传感器

API:MuVisionSensor.SensorSetRestart()

3.9 LED 灯光设置

API:MuVisionSensor.LedSetColor(led, detected_color, undetected_color, level)

参数说明：

LED：要配置的 LED 灯，可选值为

LED1

LED2

detected_color: 检测到结果时的颜色, 可选值为

LedClose	LED 关
LedRed	红色
LedGreen	绿色
LedYellow	黄色
LedBlue	蓝色
LedPurple	紫色
LedCyan	青色
LedWhite	白色

undetected_color: 未检测到结果时的颜色, 可选值同上

level: 亮度值, 可输入 0~15 的数字, 数值越大越亮

3.10 获取算法识别结果

API: MuVisionSensor.GetValue(vision_type, object_inf)

vision_type 的可选值同上

object_inf 的可选值为:

Status	检测状态, 0 代表没检测到, 1 代表检测到
XValue	目标的横向坐标
YValue	目标的纵向坐标
WidthValue	目标的宽度
HeightValue	目标的高度
Label	目标的标签
RValue	红色通道值 (颜色识别模式)

GValue 绿色通道值（颜色识别模式）

BValue 蓝色通道值（颜色识别模式）

```
1. #example:
2. from MuVisionSensor import *
3. ... #省略初始化过程
4. #形状卡片检测
5. while True:
6.     if mu.GetValue(VISION_SHAPE_CARD_DETECT,Status):
7.         print("x: %d" % mu.GetValue(VISION_SHAPE_CARD_DETECT,XValue))
8.         print("y: %d" % mu.GetValue(VISION_SHAPE_CARD_DETECT,YValue))
9.         print("height: %d" % mu.GetValue(VISION_SHAPE_CARD_DETECT,HeightValue))
10.        print("width: %d" % mu.GetValue(VISION_SHAPE_CARD_DETECT,WidthValue))
11.        print("Label: %d" % mu.GetValue(VISION_SHAPE_CARD_DETECT,Label))
12.    else:
13.        print("undetected")
14. # 颜色检测
15. while True:
16.     if mu.GetValue(VISION_COLOR_RECOGNITION,Status):
17.         print("R: %d" % mu.GetValue(VISION_COLOR_RECOGNITION,RValue) )
18.         print("G: %d" % mu.GetValue(VISION_COLOR_RECOGNITION,GValue) )
19.         print("B: %d" % mu.GetValue(VISION_COLOR_RECOGNITION,BValue) )
```

技术支持

感谢您购买使用小 MU 视觉传感器，我司会持续更新本产品的固件以及配套的库函数

与例程，您可以从如下网址获取最新的技术资料：

GitHub: <https://github.com/mu-open-source/>

官网: <http://mai.morpx.com/page.php?a=sensor-support>

您在使用过程中有遇到技术方面的问题，可以通过电话 0571-81958588 或者 email:

support@morpx.com 与我司取得联系。

官方技术支持微信号



官方技术支持 QQ 号



修订历史

日期	版本	发布说明
2019-3-28	V0.0.1-beta	起草
2019-06-25	V0.8.0	<ol style="list-style-type: none"> 1. 文档格式标准化; 2. 更新了固件下载链接; 3. 更新了传感器的图片, 硬件版本为 V311; 4. 增加技术支持的联系方式; 5. 增加了文档的修订历史; 6. 其他错误修正。
2019-06-29	V0.8.1	<ol style="list-style-type: none"> 1. 调整了书签; 2. 示例程序格式调整。