1.7 MicroPython API 接口说明

1.外设接口

Hiwonder 外设模块

1.1.按键

Hiwonder . hw_key()			
函数描述	返回一个按键对象		
参数列表	key_num:（1 为 SW1 按键，2 为 SW2 按键）	返回值	1 为按下，0 为松开
示例	hw_key() . key_scan(key_num)		
注意事项	/		

1.2.补光灯

Hiwonder . fill_light()			
函数描述	返回一个补光灯控制对象		
参数列表	key_num:（1 为 SW1 按键，2 为 SW2 按键）	返回值	onoff: 1 为开，0 为关
示例	fill_light() . fill_onoff(onoff))		
注意事项	/		

1.3.LED 灯

Hiwonder . hw_led()			
函数描述	返回一个 LED 灯控制对象,控制 LED 灯亮灭		
参数列表	onoff：1 为开，0 为关	返回值	/
示例	led_onoff(onoff)		
注意事项	/		

1.4.IIC 接口

Hiwonder . hw_i2c()			
函数描述	扫描 IIC 总线上的从机,返回一个 IIC 外设控制对象		
参数列表	onoff：1 为开，0 为关	返回值	list 对象，包含扫描到的所有从机地址。
示例	hw_i2c . scan()		
注意事项	注意：LCD 触摸与外扩 IIC 接口使用同一条 IIC 总线，不接 IIC 外设，也可以扫描到一个 IIC 从机地址，即为触摸屏，在发送数据时也要选对 IIC 从机地址！		

hw_i2c . readfrom_into(addr , buf , stop=True)			
函数描述	读取数据并放到 buf 变量中		
参数列表	addr：从机地址 buf： bytearray 类型，定义了长度，读取到的数据存放在此。 stop： 是否产生停止信号，保留，目前只能使用默认值 Ture	返回值	list 对象，包含扫描到的所有从机地址。
示例	/		
注意事项	/		

hw_i2c . readfrom(addr, len, stop=True)			
函数描述	读取数据并放到 buf 变量中		
参数列表	addr：从机地址。 len： 类型， 定义了长度，读取到的数据存放在此。 stop： 是否产生停止信号，保留，目前只能使用默认值 Ture	返回值	读取到的数据，bytes 类型.
示例	/		
注意事项	/		

hw_i2c . readfrom_mem(addr , memaddr , nbytes , mem_size=8)			
函数描述	读取数据并放到 buf 变量中		
参数列表	addr：从机地址。 memaddr： 从机寄存器地址。 nbytes： 需要读取的长度。 mem_size： 寄存器宽度，默认为 8 位。	返回值	地址值
示例	/		
注意事项	/		

hw_i2c . readfrom_mem_into(addr, memaddr, buf, mem_size=8)			
函数描述	读取从机寄存器值到指定变量中		
参数列表	addr：从机地址。 memaddr： 从机寄存器地址。 buf： bytearray 类型，定义了长度，读取到的数据存放在此。 mem_size： 寄存器宽度，默认为 8 位。	返回值	返回从机寄存器值。
示例	/		
注意事项	/		

hw_i2c . writeto(addr, buf, stop=True)			
函数描述	写数据到目标地址		
参数列表	addr：从机地址。 buf： 需要发送的数据。 stop： 是否产生停止信号，保留，目前只能使用默认值 Ture。	返回值	成功发送的字节数。
示例	/		
注意事项	/		

hw_i2c . writeto_mem(addr, memaddr, buf, mem_size=8)			
函数描述	写数据到从机寄存器		
参数列表	addr：从机地址 memaddr： 从机寄存器地址 buf： 需要写的数据 mem_size：寄存器宽度， 默认为 8 位	返回值	/
示例	/		
注意事项	/		

1.5.UART 接口

Hiwonder . hw_uart()			
函数描述	读取 uart 信息		
参数列表	addr：从机地址。 memaddr： 从机寄存器地址。 buf： 需要写的数据。 mem_size： 寄存器宽度，默认为 8 位。	返回值	返回一个 UART 对象。
示例	/		
注意事项	/		

hw_uart() . send_bytearray(buf)			
函数描述	发送 buf 变量数据		
参数列表	buf：发送参数	返回值	bytearray 类型
示例	/		
注意事项	/		

hw_uart() . rec_bytes()			
函数描述	读取串口		
参数列表	/	返回值	bytearray 类型
示例	/		
注意事项	注意：最多一次性读取 200 字节，一般足够使用		

2.标准库

标准库集成了 cmath、gc、math、uos 等函数库。
更多标准库 API 可参考链接：[Python 标准库和 Micropython 标准微库 — CanMV 文档 (canaan-creative.com)]

2.1.cmath 模块

提供了一些处理复数的基本数学函数

cmath	
名称	作用
cmath.cos(z)	返回 z 的余弦。
cmath.exp(z)	返回 z 的指数
cmath.log(z)	返回 z 的自然对数。分支切割沿负实轴
cmath.log10(z)	返回 z 的以 10 为底的对数。分支切割沿负实
cmath.phase(z)	返回数字“z”的相位，范围（-pi，+ pi）
cmath.polar(z)	作为元组返回 z 的极性形式
cmath.rect(r, phi)	返回模数 r 和相位 phi 的复数
cmath.sin(z)	返回 z 的正弦值
cmath.sqrt(z)	返回 z 的平方根
cmath.e	自然对数的基础
cmath.pi	圆周长与直径的比值

2.2.gc 模块

控制垃圾收集模块

gc	
名称	作用
gc.enable()	启用自动垃圾回收
gc.disable()	禁用后仍然可以分配堆内存，仍然可以使用 gc.collect() 手动启动垃圾收集
gc.collect()	运行垃圾回收
gc.mem_alloc()	返回分配的堆 RAM 的字节数
gc.mem_free()	返回可用堆 RAM 的字节数，如果堆剩余数量未知，则返回-1

2.3.math 模块

提供了一些处理浮点数的基本数学函数。

math	
名称	作用
math.acos(x)	返回 x 的反余弦值
math.acosh(x)	返回 x 的反双曲余弦值
math.asin(x)	返回 x 的反正弦
math.asinh(x)	返回 x 的反双曲正弦值
math.atan(x)	返回 x 的反正切
math.atan2(y, x)	返回 y /x 的反正切的主值
math.atanh(x)	返回 x 的反双曲正切
math.cos(x)	返回 x 的余弦
math.cosh(x)	返回 x 的双曲余弦值
math.degrees(x)	返回弧度 x 转换为度数
math.exp(x)	返回 x 的指数
math.fabs(x)	返回 x 的绝对值
math.fmod(x, y)	返回 x /y 的余数
math.frexp(x)	将浮点数分解为尾数和指数。返回的值是元组（m，e），使得 x == m * 2 ** e 完全正确。如果 x == 0 则函数返回（0.0,0），否则关系 0.5 <= abs（m）<1 成立
math.isnan(x)	如果 x 不是数字，则返回 True
math.ldexp(x, exp)	返回 x *（2 ** exp）
math.log(x)	返回 x 的自然对数
math.log10(x)	返回 x 的以 10 为底的对数
math.log2(x)	返回 x 的 base-2 对数
math.modf(x)	返回两个浮点数的元组，是“x”的分数和整数部分。两个返回值都与 x 具有相同的符号
math.pow(x, y)	将 x 返回到'y`的幂
math.radians(x)	返回度数 x 转换为弧度
math.sin(x)	返回 x 的正弦值
math.sinh(x)	返回 x 的双曲正弦值
math.sqrt(x)	返回 x 的平方根
math.tan(x)	返回 x 的正切值
math.tanh(x)	返回 x 的双曲正切
math.trunc(x)	返回一个整数，“x”向 0 舍入

2.4.uos 模块

uos 模块包含用于文件系统访问和挂载，终端重定向和复制以及 uname 和 urandom 等函数。

所属函数：

math	
名称	作用
uos.urandom(n)	返回一个包含 n 个随机字节的字节对象。只要有可能，它就由硬件随机数生成器生成
uos.chdir(path)	更改当前目录
uos.getcwd()	获取当前目录
uos.listdir([dir])	如果没有参数，请列出当前目录。否则列出给定目录
uos.mkdir(path)	创建一个新目录
uos.rmdir(path)	删除目录
uos.remove(path)	删除文件
uos.rename(old_path, new_path)	重命名文件
uos.stat(path)	获取文件或目录的状态

3.Maix 模块

Maix 模块集成了 FPIOA、GPIO、KPU、freq 等函数库
更多 Maix 模块 API 可参考链接：
maix.FPIOA — CanMV 文档 (canaan-creative.com)

3.1.FPIOA（现场可编程 IO 阵列）

K210 支持每个外设随意映射到任意引脚， 使用 FPIOA 功能来实现。

FPIOA()			
函数描述	返回一个 FPIOA 对象		
参数列表	/	返回值	FPIOA 对象
示例	/		
注意事项	/		

FPIOA() . set_function(pin, func)			
函数描述	设置引脚对应的外设功能， 即引脚映射		
参数列表	pin： 引脚编号，取值 [0，47]， 具体的引脚连接请看电路图。 func：外设功能，传一个整型值，可以通过 fm.fpioa.help()或者查看下面的外设功能表. pin： 引脚编号，取值 [0，47]， 具体的引脚连接请看电路图。 func：外设功能，传一个整型值，可以通过 fm.fpioa.help()或者查看下面的外设功能表	返回值	FPIOA 对象
示例	/		
注意事项	/		

FPIOA() . get_Pin_num(func)			
函数描述	获取外设映射到哪个引脚上		
参数列表	func： 外设功能，传一个整型值，可以通过 fm.fpioa.help()或者查看下面的外设功能表	返回值	功能列表
示例	/		
注意事项	/		

外设功能表的部分参数如下：

外设功能（func）	简要描述
.....	.....
GPIOHS0	GPIO High speed 0
GPIOHS1	GPIO High speed 1
GPIOHS2	GPIO High speed 2
GPIOHS3	GPIO High speed 3
GPIOHS4	GPIO High speed 4
GPIOHS5	GPIO High speed 5
GPIOHS6	GPIO High speed 6
GPIOHS7	GPIO High speed 7
GPIOHS8	GPIO High speed 8
GPIOHS9	GPIO High speed 9
GPIOHS10	GPIO High speed 10
GPIOHS11	GPIO High speed 11
GPIOHS12	GPIO High speed 12
GPIOHS13	GPIO High speed 13
GPIOHS14	GPIO High speed 14
GPIOHS15	GPIO High speed 15

外设功能（func）	简要描述
GPIOHS16	GPIO High speed 16
GPIOHS17	GPIO High speed 17
GPIOHS18	GPIO High speed 18
GPIOHS19	GPIO High speed 19
GPIOHS20	GPIO High speed 20
GPIOHS21	GPIO High speed 21
GPIOHS22	GPIO High speed 22
GPIOHS23	GPIO High speed 23
GPIOHS24	GPIO High speed 24
GPIOHS25	GPIO High speed 25
GPIOHS26	GPIO High speed 26
GPIOHS27	GPIO High speed 27
GPIOHS28	GPIO High speed 28
GPIOHS29	GPIO High speed 29
GPIOHS30	GPIO High speed 30
GPIOHS31	GPIO High speed 31
GPIOHS25	GPIO High speed 25
GPIOHS26	GPIO High speed 26
GPIOHS27	GPIO High speed 27
GPIOHS28	GPIO High speed 28
GPIOHS29	GPIO High speed 29
GPIOHS30	GPIO High speed 30
GPIOHS31	GPIO High speed 31
GPIOHS28	GPIO High speed 28
GPIOHS29	GPIO High speed 29

外设功能（func）	简要描述
GPIOHS30	GPIO High speed 30
GPIOHS31	GPIO High speed 31
GPIO0	GPIO pin 0
GPIO1	GPIO pin 1
GPIO2	GPIO pin 2
GPIO3	GPIO pin 3
GPIO4	GPIO pin 4
GPIO5	GPIO pin 5
GPIO6	GPIO pin 6
GPIO7	GPIO pin 7
GPIO0	GPIO pin 0
GPIO1	GPIO pin 1
GPIO2	GPIO pin 2
GPIO3	GPIO pin 3
GPIO4	GPIO pin 4
GPIO5	GPIO pin 5
GPIO6	GPIO pin 6
GPIO7	GPIO pin 7
UART1_RX	UART1 Receiver
UART1_TX	UART1 Transmitter
UART2_RX	UART2 Receiver
UART2_TX	UART2 Transmitter
UART3_RX	UART3 Receiver
UART3_TX	UART3 Transmitter
UART1_RX	UART1 Receiver

外设功能（func）	简要描述
UART1_TX	UART1 Transmitter
UART2_RX	UART2 Receiver
UART2_TX	UART2 Transmitter
.....	.....

3.2.GPIO

总线扩展器，K210 上有高速 GPIO(GPIOHS) 和通用 GPIO。
该模块 API 接口详情可参考以下链接：
maix.GPIO – CanMV 文档 (canaan-creative.com)

GPIO(ID, MODE, PULL, VALUE)			
函数描述	返回一个 GPIO 控制对象		
参数列表	ID： 使用的 GPIO 引脚(一定要使用 GPIO 里带的常量来指定) MODE：GPIO 模式。 GPIO.IN 输入模式； GPIO.OUT 输出模式。 PULL：GPIO 上下拉模式 GPIO.PULL_UP 上拉 GPIO.PULL_DOWN 下拉 GPIO.PULL_NONE 即不上拉也不下拉	返回值	GPIO 控制对象
示例	/		
注意事项	/		

GPIO(ID, MODE, PULL, VALUE)			
函数描述	返回一个 GPIO 控制对象		
参数列表	ID： 使用的 GPIO 引脚(一定要使用 GPIO 里带的常量来指定) MODE：GPIO 模式。 GPIO.IN 输入模式； GPIO.OUT 输出模式。 PULL：GPIO 上下拉模式 GPIO.PULL_UP 上拉 GPIO.PULL_DOWN 下拉 GPIO.PULL_NONE 即不上拉也不下拉	返回值	GPIO 控制对象
示例	/		

注意事项	/

GPIO.value([value])			
函数描述	修改或读取 GPIO 引脚状态		
参数列表	[value]： 可选参数，1 高电平；0 低电平；如果此参数为空，则返回当前 GPIO 引脚状态。	返回值	如果[value] 参数为空，则返回当前 GPIO 引脚状态
示例	/		
注意事项	/		

GPIO.irq(CALLBACK_FUNC , TRIGGER_CONDITION, GPIO.WAKEUP_NOT_SUPPORT , PRORITY)			
函数描述	配置一个中断处理程序，当 pin 的触发源处于活动状态时调用它。如果管脚模式为 pin.in，则触发源是管脚上的外部值		
参数列表	CALLBACK_FUNC：中断回调函数，当中断触发的时候被调用 TRIGGER_CONDITION：GPIO 引脚的中断触发模式 • GPIO.IRQ_RISING 上升沿触发 • GPIO.IRQ_FALLING 下降沿触发 • GPIO.IRQ_BOTH 上升沿和下降沿都触发 GPIO.WAKEUP_NOT_SUPPORT：设置为中断模式，默认为 GPIO.WAKEUP_NOT_SUPPORT PRORITY：中断优先级	返回值	如果[value] 参数为空，则返回当前 GPIO 引脚状态
示例	/		
注意事项	/		

GPIO.disirq()			
函数描述	关闭中断		
参数列表	/	返回值	/
示例	/		
注意事项	/		

GPIO.mode(MODE)			
函数描述	设置 GPIO 输入输出模式		
参数列表	MODE：模式选择。 • GPIO.IN 输入模式 • GPIO.PULL_UP 上拉输入模式 • GPIO.PULL_DOWN 下拉输入模式 • GPIO.OUT 输出模式	返回值	/
示例	/		
注意事项	/		

3.3.KPU 模块

KPU 是通用的神经网络处理器，它可以在低功耗的情况下实现卷积神经网络计算，时时获取被检测目标的大小、坐标和种类，对人脸或者物体进行检测和分类。
该模块 API 接口详情可参考以下链接：
maix.KPU — CanMV 文档 (canaan-creative.com)
所属函数：

KPU.load(offset, file_path)			
函数描述	从 flash 或者文件系统中加载模型		
参数列表	offset 和 file_path 参数只能二选一，不需要关键词，直接传参即可。 offset: 模型在 flash 中的偏移大小，如 0xd00000 表示模型烧录在 13M 起始的地方，0x300000 表示在 Flash 3M 的地方。 file_path: 模型在文件系统中为文件名，如 “/sd/xxx.kmodel”。	返回值	如果正确加载，返回 kpu_net（kpu 网络对象），否则会抛出错误。
示例	/		
注意事项	/		

kpu.load_flash(model_addr, is_dual_buf, batch_size, spi_speed)			
函数描述	从 flash 或者文件系统中加载模型		
参数列表	model_addr: Flash addr 经过预处理的模型烧录到 flash 中的偏移地址。注意，这里需要预处理模型文件说明。 is_dual_buf: 0 为单一缓冲区加载，使用较少的 RAM 和较慢的速度动态加载该模型文件； 1 为开启双缓冲加载，需要更大的 RAM，运行速度相对较快。 batch_size: 当 is_dual_buf 设为 1 时，需设置 batch_size，建议值为 0x4000~0x10000，可以测试出模型的最佳值。如果 is_dual_buf 为 0 则设置为 0。 spi_speed: 使用 SPI flash 加载模型文件时，我们会暂时将 flash 设置为高速模式，	返回值	如果正确加载，返回 kpu_net（kpu 网络对象），否则会抛出错误。

	并设置所需的 spi 时钟频率。该值应 <= 80000000(实际频率，设值可能不等于实际频率。)		
示例	/		
注意事项	/		

KPU.init_yolo2(kpu_net, threshold, nms_value, anchor_num, anchor)			
函数描述	该函数为 yolo2 网络模型传入初始化参数，只有使用 yolo2 时使用		
参数列表	kpu_net: kpu 网络对象，即加载的模型对象，KPU.load()的返回值。 Threshold: 概率阈值， 只有是这个物体的概率大于这个值才会输出结果， 取值范围：[0, 1]。 nms_value: box_iou 门限，为了防止同一个物体被框出多个框，当在同一个物体上框出了两个框，这两个框的交叉区域占两个框总占用面积的比例如果小于这个值时，就取其中概率最大的一个框。 anchor_num: anchor 的锚点数， 这里固定为 len(anchors)//2。 anchor: 锚点参数与模型参数一致，同一个模型这个参数是固定的，和模型绑定的（训练模型时即确定了）， 不能改成其它值。	返回值	success: bool 类型，是否成功
示例	/		
注意事项	/		

KPU.deinit(kpu_net)			
函数描述	释放模型占用的内存，但是变量还在，可以使用 del kpu_net_object 的方式删除		
参数列表	kpu_net：kpu 网络对象	返回值	success：bool 类型，是否成功
示例	/		
注意事项	/		

KPU.run_with_output(img_obj)			
函数描述	使用 KPU 运行目标检测模型		
参数列表	img_obj：图像对象	返回值	/

示例	/
注意事项	/

KPU.regionlayer_yolo2()			
函数描述	获取检测结果		
参数列表	含以下 6 个数据： x，y，w，h：代表目标框的左上角 x,y 坐标，以及框的宽 w 高 h class：类别序号 prob ：概率值，范围：[0，1]	返回值	返回一个二维列表，每个子列表代表一个识别到的目标物体，目标物体信息列表包
示例	/		
注意事项	/		

4.image 图像处理模块

该模块移植于 openMV，与 openMV 功能相同。
该模块包含 Image、Line、Circle 等类，API 使用请参考以下链接：
[image - 图像处理 — CanMV 文档 (canaan-creative.com)]

5.sensor（摄像头）模块

sensor 模块(这里特指摄像头模块)用于控制开发板摄像头完成摄像任务。
该模块 API 接口详情可参考以下链接：
[sensor（摄像头） — CanMV 文档 (canaan-creative.com)]
所属函数：

sensor.reset([, freq=24000000[, set_regs=True[, dual_buff=False]]])			
函数描述	重置并初始化单目摄像头		
参数列表	freq: 设置摄像头时钟频率，频率越高帧率越高，但是画质可能更差。默认 24MHz， 如果摄像头有彩色斑点(ov7740)，可以适当调低比如 20MHz。 set_regs: 允许程序写摄像头寄存器，默认为 True。 如果需要自定义复位序列，可以设置为 False，然后使用 sensor.__write_reg(addr, value) 函数自定义写寄存器序列。	返回值	/

	dual_buff: 默认为 False。允许使用双缓冲，会增高帧率，但是内存占用也会增加(大约为 384KiB)。 choice: 指定需要搜索的摄像头类型，ov 类型(1)，gc 类型(2)，mt 类型(3)，不传入该参数则搜索全部类型摄像头		
示例	/		
注意事项	/		

sensor.set_framesize(framesize[, set_regs=True])			
函数描述	用于设置摄像头输出帧大小，开发板配置的屏幕是 320*240 分辨率，推荐设置为 QVGA 格式		
参数列表	framesize：帧大小。 set_regs: 允许程序写摄像头寄存器，默认为 True。 如果需要自定义设置帧大小的序列，可以设置为 False，然后使用 sensor.__write_reg(addr, value) 函数自定义写寄存器序列	返回值	success： bool 类型，是否成功
示例	/		
注意事项	/		

sensor.run(enable)			
函数描述	图像捕捉功能控制		
参数列表	enable：1 为开始抓取图像；0 为停止抓取图像	返回值	success： bool 类型，是否成功
示例	/		
注意事项	/		

sensor.snapshot()			
函数描述	使用摄像头拍摄一张照片		
参数列表	/	返回值	img： 返回的图像对象
示例	/		
注意事项	/		

sensor.shutdown(enable)			
函数描述	关闭摄像头		
参数列表	enable: True 表示开启摄像头 False 表示关闭摄像头	返回值	/
示例	/		
注意事项	/		

sensor.skip_frames(n, [, time])			
函数描述	跳过指定帧数或者跳过指定时间内的图像，让相机图像在改变相机设置后稳定下来		
参数列表	n：跳过 n 帧图像 time: 跳过指定时间，单位为 ms 若 n 和 time 皆未指定，该方法跳过 300 毫秒的帧；若二者皆指定，该方法会跳过 n 数量的帧，但将在 time 毫秒后返回	返回值	/
示例	/		
注意事项	/		

6.LCD 模块

用于 LCD 屏幕显示控制。
该模块 API 接口详情可参考以下链接：
[lcd（屏幕显示） — CanMV 文档 (canaan-creative.com)]
所属函数：

lcd.init(type=1, freq=15000000, color=lcd.BLACK, invert = 0, lcd_type = 0)			
函数描述	初始化 LCD 屏幕显示		
参数列表	type: 设备的类型（保留给未来使用），用户默认即可。 freq: LCD （实际上指 SPI 的通讯速率）的频率。 color: LCD 初始化的颜色， 可以是 16 位的 RGB565 颜色值，比如 0xFFFF； 或者 RGB888 元组，比如(236, 36, 36)，默认 lcd.BLACK。	返回值	/

	invert：LCD 反色显示。 lcd_type：lcd 类型，用户默认即可		
示例	/		
注意事项	/		

lcd.display(image, roi=Auto, oft=(x, y))			
函数描述	在液晶屏上显示一张 image（GRAYSCALE 或 RGB565）		
参数列表	Image：图像对象。 roi： 是一个感兴趣区域的矩形元组(x, y, w, h)。若未指定，即为图像矩形。 oft： 设置偏移坐标，设置了这个坐标就不会自动填充周围了。	返回值	/
示例	/		
注意事项	/		

lcd.clear(color)			
函数描述	将液晶屏清空为黑色或者指定的颜色		
参数列表	color： LCD 初始化的颜色， 可以是 16 位的 RGB565 颜色值，比如 0xFFFF； 或者 RGB888 元组， 比如(236, 36, 36)	返回值	/
示例	/		
注意事项	/		

lcd.rotation(dir)			
函数描述	设置 LCD 屏幕方向		
参数列表	dir：取值范围 [0,3]，从 0 到 3 依次顺时针旋转	返回值	当前方向，取值 [0,3]
示例	/		
注意事项	/		

lcd.mirror(invert)			
函数描述	设置 LCD 是否镜面显示		
参数列表	invert：是否镜面显示，True 或者 False	返回值	当前设置，是否镜面显示，返回 True 或者 False
示例	/		
注意事项	/		

lcd.fill_rectangle(x, y, w, h, color)			
函数描述	填充 LCD 指定区域		
参数列表	X：起始坐标 x x：起始坐标 y w：填充宽度 h：填充高度 color：填充颜色， 可以是元组，比如(255，255，255)，或者 RGB565``uint16 值， 比如红色 0x00F8	返回值	当前设置，是否镜面显示，返回 True 或者 False
示例	/		
注意事项	/		

7.souchscreen（触摸）模块

touchscreen 模块包含了基本的读取触摸屏幕操作。
该模块 API 接口详情可参考以下链接：
touchscreen（触摸屏幕） — CanMV 文档（canaan-creative.com）
所属函数：

Souchscreen . init(i2c=None, cal=None)			
函数描述	初始化触摸屏		
参数列表	i2c: 目前支持的是 I2C 通信的触摸屏，传入 I2C 实例对象， 后期这个参数可能会被重命名或者取消。 cal: 校准数据，是一个 7 个整型值的元组， 可以通过 touchscreen.calibrate() 函数得到	返回值	/
示例	/		
注意事项	/		

Souchscreen . read()			
函数描述	读取当前屏幕的状态以及按下的点的坐标值		
参数列表	/	返回值	一个由 3 个整型值组成的元组（status，x，y），注意这个值会一直保持上一个状态 status：状态，取值有： **touchscreen.STATUS_PRESS**：屏幕被按下 **touchscreen.STATUS_MOVE**：屏幕被按住并移动 **touchscreen.STATUS_RELEASE**：屏幕不再被按住，即没有点击 x： x 轴坐标 y： y 轴坐标
示例	/		
注意事项	/		