



小 MU 视觉传感器 - 测试指导

MU Vision Sensor - Testing



杭州摩图科技有限公司

版本 V0.1

2019.06.22

目 录

免责声明和版权公告.....	3
1 概述.....	4
1.1 产品介绍.....	4
2 硬件与结构.....	4
2.1 产品外观.....	4
2.2 尺寸标注.....	4
2.3 信号输出接口.....	5
2.4 输出模式选择拨码开关.....	5
2.5 地址选择拨码开关.....	5
2.6 电气特性.....	5
3 视觉算法.....	6
3.1 图像识别的特性.....	6
3.2 图像坐标系与检测数据.....	6
3.3 算法列表.....	7
4 测试.....	7
4.1 算法基础测试.....	7
4.2 摄像头功能测试.....	8
4.2.1 白平衡测试.....	8
4.2.2 采样速率测试.....	8
4.2.3 镜头缩放测试.....	9
4.2.4 镜头旋转测试.....	9
4.3 LED 功能测试.....	9
4.3.1 自动模式测试.....	9
4.3.2 手动模式测试.....	10
4.4 通讯功能测试.....	10
4.5 颜色检测算法测试.....	11
4.6 颜色识别算法测试.....	11
4.7 多类型卡片算法测试.....	12
修订历史.....	13

免责声明和版权公告

•本手册中的信息仅适用于摩图科技公司所生产的小 MU 视觉传感器 MUVS-AB2 型号 V311 版本（下称产品）的出厂固件，固件的升级能够提升性能或者引入新功能，敬请留意摩图科技官网 <http://www.morpx.com> 以便获取最新版本，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。

•未经摩图科技确认及授权，不可私自维修或改装产品上的电子元件，造成损坏的将不予以保修。

•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 概述

1.1 产品介绍

小 MU 视觉传感器是一款用于图像识别的传感器，其内置的深度学习引擎可以识别多种目标物体，例如颜色检测，球体检测，人体检测，卡片识别等。检测结果可以通过 UART 或 I2C 进行输出，体积小，功耗低，所有算法本地处理，无须联网，可广泛应用于智能玩具、人工智能教具、创客等产品或领域。

该手册提供了小 MU 功能测试方案及步骤，需要配合 arduino 开发环境以及 MU Vision Sensor 库文档。

2 硬件与结构

2.1 产品外观

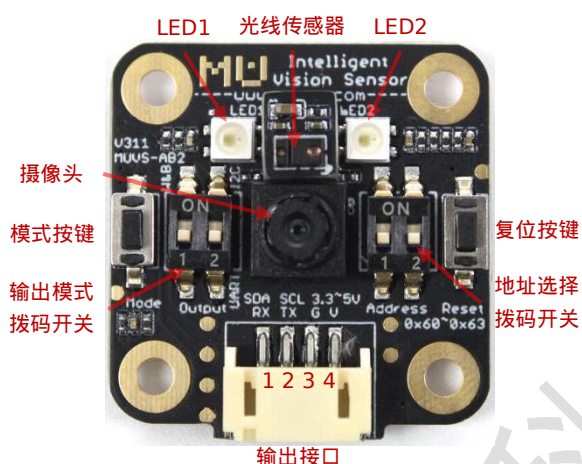


图 1. 正面视图



图 2. 反面视图

2.2 尺寸标注

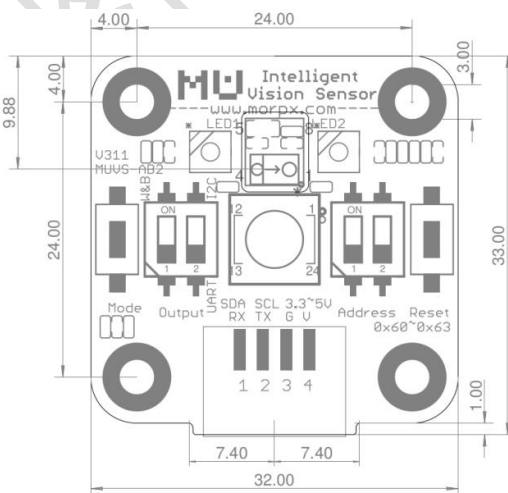


图 3. 正面尺寸标注视图

表 1. pcb 尺寸信息

项目	参数
长	33mm
宽	32mm
高	11.5mm
重量	6.8g

2.3 信号输出接口

用于与主控设备的信号传输使用，采用 PH2.0-4P 接口，UART 模式和 I2C 模式引脚定义见下表：

表 2. 信号输出接口定义

管脚	UART 模式	I2C 模式
1	RX	SDA
2	TX	SCL
3	GND	GND
4	VDD	VDD

2.4 输出模式选择拨码开关

采用 2 位拨码开关来选择信号输出类型，定义见下表：

表 3. 输出模式拨码开关定义

拨码 1	Wifi 开关	拨码 2	信号输出类型
0	禁止使用 Wifi 功能*	0	UART 通讯方式
1	允许使用 Wifi 功能*	1	I2C 通讯方式

注 1：*所示功能，当前版本尚不具备

注 2：向上拨动为 1，向下拨动为 0

2.5 地址选择拨码开关

采用 2 位拨码开关来设定小 MU 设备地址，定义见下表：

表 4. 地址选择拨码开关定义

拨码 1	拨码 2	设备地址
0	0	0x60
0	1	0x61
1	0	0x62
1	1	0x63

注 1：UART 模式下可使用 0x00 作为广播地址

注 2：向上拨动为 1，向下拨动为 0

2.6 电气特性

表 5. 电气特性参数

项目	单位	最小值	典型值	最大值
输入电压	V	3.3	5	5.5
供电电流 ⁽¹⁾	mA	500	-	-
工作电流(UART 模式) ⁽²⁾	mA	60	80	90
工作电流(I2C 模式) ⁽²⁾	mA	60	85	100
UART 波特率	bps	9600	115200	921600
I2C 工作频率	Kbps	-	100	400
工作温度	°C	-20	25	70
Wifi 输入频率	MHz	2412	-	2484

注 1：在使用 Wifi 时，天线初始化时需要一个至少 500mA 的电流，初始化之后会降低为正常功耗水平；

注 2：此工作电流在未开启 Wifi 下测得，其与 LED 亮度，摄像头采样频率，算法开启状态等因素相关。

3 视觉算法

小 MU 视觉传感器集成有多种计算机视觉算法，内置摩图科技自主研发的嵌入式深度学习引擎，可以支持物体的检测和分类。MU 支持同时开启多种算法，但算法开启越多，速度将会变慢。每种算法可以单独设置其输出模式，部分算法支持性能 Level，参数输入，多物体检测，详见各算法介绍。

3.1 图像识别的特性

图像识别是对光源、色彩、背景、物体移动速度等因素十分敏感的技术，使用环境的差异会对图像检测结果将产生不同的影响，为了获取较好的识别结果：

- (1) 避免在过暗、过亮、强逆光的环境下使用，比如昏暗的房间，具有强烈光亮差别的窗边等；
- (2) 避免让灯光或强烈阳光直射目标物体，避免造成物体反光；
- (3) 避免在彩色灯光或可变换的灯光下使用，稳定均匀的白色是最好的光源；
- (4) 避免正对光源使用；
- (5) 颜色敏感型算法不要在有近似颜色的背景下使用，比如绿色网球不要在绿色地毯上使用；
- (6) 避免在有类似目标物体的环境下使用，避免误报，比如球体检测旁边有一个橙子；

3.2 图像坐标系与检测数据

小 MU 视觉传感器采用图像坐标系，左上角为坐标原点 (0, 0)，图像长宽比例为 4: 3，图像检测结果量化到了 0 ~ 100 的区间内，如下图所示：

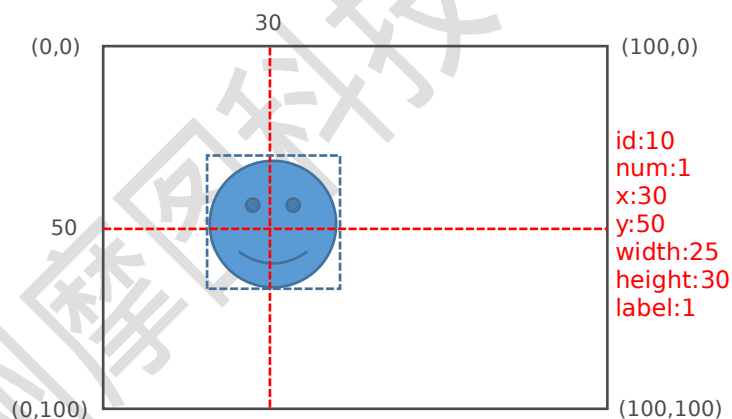


图 4. 图像坐标示意图

检测数据：

- (1) id：算法类型编号
- (2) num：目标物体数量
- (2) x：水平中心坐标
- (3) y：垂直中心坐标
- (4) width：物体边缘宽度
- (5) height：物体边缘高度
- (6) label：分类标签编号，部分算法适用，详见各算法介绍

3.3 算法列表

表 7. 算法列表

算法 ID	算法类型	识别对象	返回结果					特殊要求
			data1	data2	data3	data4	data5	
1	颜色检测	指定的颜色	x	y	width	height	label	良好的白色光源
2	颜色识别	指定区域颜色	R	G	B	/	label	良好的白色光源
3	球体检测	橙色乒乓球 绿色网球	x	y	width	height	label	背景不宜过于复杂, 尽量排除与目标球体近似的物体
4	预留	/	/	/	/	/	/	/
5	人体检测	人体上半身	x	y	width	height	/	需要包含头部特征, 正对摄像头效果最好
6	图形类卡片识别	对号, 叉号, 圆形, 方形, 三角形 5 种卡片	x	y	width	height	label	卡片与镜头尽量垂直, 卡片旋转<30°, 倾斜角度<30°
7	交通标志类卡片识别	前进, 左转, 右转, 掉头, 停止 5 种卡片	x	y	width	height	label	卡片与镜头尽量垂直, 卡片旋转<30°, 倾斜角度<30°
8	数字类卡片识别	0~9 数字卡片	x	y	width	height	label	卡片与镜头尽量垂直, 卡片旋转<30°, 倾斜角度<30°

4 测试

本手册测试包括算法基础测试, 摄像头功能测试, LED 功能测试, 通讯功能测试, 颜色检测算法测试, 颜色识别算法测试, 多类型卡片算法测试

4.1 算法基础测试

测试目的: 测试并统计各算法的运行基本信息, 查看不同性能下算法检测数据, 识别率, 运行速度等内容。

测试步骤:

- (1) 打开 test/01_VisionBasic.ino 项目
- (2) 宏 #define VISION_TYPE XXXXX 选则一项视觉算法;
- (3) 宏 #define VISION_LEVEL XXXXX 选择一项算法性能参数;
- (4) 选择通讯方式 UART 或 I2C, 与小 MU 上的 Output 拨码开关保持一致;
注:默认 UART 为软串口方式,硬件串口用于输出调试信息,如果通讯方式为 UART, 还需设置软串口 Rx 和 Tx 引脚。
- (4) 小 MU 默认地址为 0x60, 请确认与 Address 拨码开关保持一致;
- (5) 编译并下载程序至 Arduino 中;
- (6) 打开串口监视器查看输出数据;
- (7) 前后左右上下移动目标物体, 查看坐标和宽高变化趋势;
- (8) 更换同算法(如交通卡片识别)中的不同 label 物体(如前进卡片换为左转卡片), 查看 label 标签编号是否一致;
- (9) 更换与该算法不同类型的物体, 观察是否有误报现象;
- (10) 更换使用场景, 观察不同场景对算法识别的影响。

测试数据与预期结果:

Frame: 帧号, 连续输出 1~100 的帧号, 没有跳帧

Rate: 识别率, 识别到物体次数/总运行次数, 不同算法性能下会有所差异

Time: 当前帧算法处理时间

Fps: 算法运行平均帧率

Detected (检测到) /undetected (未检测到): 是否检测到目标物体

Label: 分类标签编号, 应与手册中描述一致

X: 目标物体中心 x 坐标 (0 ~ 100%), 物体左右移动应符合变化趋势

Y: 目标物体中心 y 坐标 (0 ~ 100%), 物体上下移动应符合变化趋势

Width: 目标物体宽度 (0 ~ 100%), 物体前后移动应符合变化趋势

Height: 目标物体高度 (0 ~ 100%), 物体前后移动应符合变化趋势

4.2 摄像头功能测试

测试摄像头各配置项功能, 包括白平衡测试, 采样速率测试, 镜头缩放测试, 镜头旋转测试, 查看各配置对视觉识别效果的影响。

4.2.1 白平衡测试

测试目的: 灯光环境会对物体颜色产生一定的影响, 通过该测试体会白平衡功能对于颜色矫正的作用, 学会使用锁定白平衡功能。

(1) 打开 test/02_CameraConfig.ino 项目

(2) 打开 AWB_TEST 宏定义;

(3) 判断当前房间内灯光环境, 并在下方宏定义配置中, 选择一种 AWB_MODE 模式: kAutoWhiteBalance (自动白平衡), kLockWhiteBalance (锁定白平衡), kWhiteLight (白光模式), kYellowLight (黄光模式);

注: 如果选择锁定白平衡模式, 开机时应将摄像头距离白纸 20cm 处进行测光 2 秒。

(4) 该测试默认采用颜色识别算法;

(5) 选择通讯方式和小 MU 地址;

(6) 编译并下载程序至 Arduino 中;

(7) 打开串口监视器查看输出数据;

(8) 将被测物体正对摄像头中心位置, 并保持一定的距离, 如 20cm, 查看颜色 label 是否正确;

(9) 将被测物体移近或远离摄像头, 查看颜色 label 是否正确;

(10) 更改光源环境再进行测试。

测试数据与预期结果:

Detected (检测到) /undetected (未检测到): 是否检测到目标物体

Label: 分类标签编号, 应与被识别物体颜色一致

4.2.2 采样速率测试

测试目的: 高采样速率有助于减少图像拖影现象, 缩短运行时间, 适合于需要快速检测的场合, 但功耗会有所增加

(1) 打开 test/02_CameraConfig.ino 项目

(2) 打开 FPS_TEST 宏定义;

(3) 选择一个 FPS_VALUE, kFPSHigh (高速率), kFPSNormal (普通速率)

(4) 该测试默认采用球体检测算法;

(5) 选择通讯方式和地址;

(6) 编译并下载程序至 Arduino 中;

(7) 打开串口监视器查看输出数据;

(8) 按不同速率移动球体, 查看检测情况。

测试数据与预期结果:

Detected (检测到) /undetected (未检测到): 是否检测到目标物体

Label: 分类标签编号, 应与手册中描述一致

X: 目标物体中心 x 坐标 (0 ~ 100%), 物体左右移动应符合变化趋势

Y: 目标物体中心 y 坐标 (0 ~ 100%), 物体上下移动应符合变化趋势

Width: 目标物体宽度 (0 ~ 100%), 物体前后移动应符合变化趋势

Height: 目标物体高度 (0 ~ 100%), 物体前后移动应符合变化趋势

4.2.3 镜头缩放测试

测试目的: 提高镜头缩放数值可以识别到远处的物体, 但视野范围会变窄, 反之识别物体近, 但视野范围广。

(1) 打开 test/02_CameraConfig.ino 项目

(2) 打开 ZOOM_TEST 宏定义;

(3) 选择一个 ZOOM_VALUE, kZoom1 (距离近, 视野广) ~ kZoom5 (距离远, 视野窄);

(4) 该测试默认采用形状卡片识别;

(5) 选择通讯方式和地址;

(6) 编译并下载程序至 Arduino 中;

(7) 打开串口监视器查看输出数据;

(8) 前后左右上下移动被测物体, 观察坐标和宽高变化趋势。

测试数据与预期结果:

同上。

4.2.4 镜头旋转测试

测试目的: 镜头旋转后, 被识别物体需要反向放置, 适用于有特殊需求的场合。

(1) 打开 test/02_CameraConfig.ino 项目

(2) 打开 ROTATE_TEST 宏定义;

(3) 设置 ROTATE_FLAG, 0 (正向角度), 1 (图像旋转 180 度);

(4) 该测试默认采用形状卡片识别;

(5) 选择通讯方式和地址;

(6) 编译并下载程序至 Arduino 中;

(7) 打开串口监视器查看输出数据;

(8) 分别正放和反放卡片, 观察镜头旋转后的检测效果。

测试数据与预期结果:

同上。

4.3 LED 功能测试

测试 LED 配置功能, 包括自动模式, 手动模式, 颜色更改, 亮度更改。

4.3.1 自动模式测试

测试目的: LED 灯光颜色会随物体检测结果而自动变化, 起到检测过程的指示性作用, 灯光每闪烁一次即执行了一帧图像处理。

- (1) 打开 test/03_LedConfig.ino 项目
- (2) 宏 LED1_MANUAL_MODE 与 LED2_MANUAL_MODE 设为 0, 即为自动模式;
- (3) 选择 LED1_HOLD 和 LED2_HOLD 是否开启, 关闭状态下 (设为 0), 每帧图像处理完后会自动关闭 LED, 即形成闪烁效果; 如果为开启状态 (设为 1), 每帧图像处理完后 LED 会持续发光;
- (4) 配置 LED1_DETECTED_COLOR (识别到物体时 LED1 显示的颜色), LED1_UNDETECTED_COLOR (未识别到物体时 LED1 显示的颜色), LED2_DETECTED_COLOR (识别到物体时 LED2 显示的颜色), LED2_UNDETECTED_COLOR (未识别到物体时 LED2 显示的颜色);
- (5) 设置 LED1_LEVEL (LED1 发光亮度), LED2_LEVEL (LED2 发光亮度), 范围 0 ~ 15 (最亮);
- (6) 该测试默认采用人体检测算法;
- (7) 选择通讯方式和小 MU 地址;
- (8) 编译并下载程序至 Arduino 中;
- (9) 打开串口监视器查看输出数据;
- (10) 识别被测物体, 观察灯光显示状态与设定参数是否一致。

测试数据与预期结果:

Detected (检测到) /undetected (未检测到): 是否检测到目标物体

4.3.2 手动模式测试

测试目的: 手动模式下, LED 灯光会持续显示固定的颜色, 当设为白色时, 可以作为补光灯使用。

- (1) 打开 test/03_LedConfig.ino 项目
- (2) 宏 LED1_MANUAL_MODE 与 LED2_MANUAL_MODE 设为 1, 即为手动模式;
- (3) 配置 LED1_DETECTED_COLOR (LED1 显示的颜色), LED2_DETECTED_COLOR (LED2 显示的颜色);
- (4) 设置 LED1_LEVEL (LED1 发光亮度), LED2_LEVEL (LED2 发光亮度), 范围 0 ~ 15 (最亮);
- (5) 该测试默认采用人体检测算法;
- (6) 选择通讯方式和小 MU 地址;
- (7) 编译并下载程序至 Arduino 中;
- (8) 打开串口监视器查看输出数据;
- (9) 观察灯光显示状态与设定参数是否一致。

测试数据与预期结果:

Detected (检测到) /undetected (未检测到): 是否检测到目标物体

4.4 通讯功能测试

测试目的: 测试软串口, 硬串口 (仅 Mega1280/Mega2560), I2C 通讯方式下, 不同配置下的通信速率, 丢包率问题。该测试不断向寄存器写入和读取循环变化的数据, 其过程包含小 MU 内部数据处理的时间, 协议解析时间等。

- (1) 打开 test/04_Communication.ino 项目
- (2) 选择小 MU 地址 MU_DEVICE_ADDRESS, Address 拨码开关 00 对应地址 0x60, 01 对应地址 0x61, 10 对应地址 0x62, 11 对应地址 0x63;

(3) 选择通讯方式, OUTPUT_MODE_SOFT_SERIAL (软件虚拟串口) 适用于所有 arduino 板子, OUTPUT_MODE_HARD_SERIAL (硬件串口) 适用于具有 Serial1 串口的板子, 例如 Mega1280, Mega2560, OUTPUT_MODE_I2C (I2C) 适用于所有板子;

注: 如果选择软串口方式, 还需要配置 SOFT_SERIAL_RX_PIN, SOFT_SERIAL_TX_PIN 端口

(4) 选择测试波特率 BAUDRATE 或是 I2C 时钟频率;

(5) 该测试默认采用人体检测算法;

(6) 打开串口监视器查看输出数据;

(7) 测试不同长度的传输线路, 如 20cm, 50cm, 100cm 等, 查看通讯情况;

(8) 测试不同类型的传输线路, 如排线, 双绞线, 电缆线等, 查看通讯情况。

测试数据与预期结果:

Total send: 总测试次数

Successed: 读/写正确的数量

Success rate: 读/写成功率

R/W average time: 读/写平均时间

4.5 颜色检测算法测试

测试目的: 颜色检测是指定一种颜色, 返回其在空间的坐标和大小, 首先需要指定待检测的颜色和最小检测框的大小, 最小检测框可以在一定程度上排除背景中杂色的误报。

(1) 打开 test/05_ColorDetect.ino 项目

(2) 选择一项待检测的颜色 COLOR;

(3) 设置最小检测框的 WIDTH (宽) 和 HEIGHT (高);

(4) 设置算法性能 VISION_LEVEL;

(5) 设置颜色检测算法;

(6) 该测试默认采用锁定白平衡方式, 因此开机时需要面向白纸校准白平衡;

(7) 选择通讯方式和小 MU 地址;

(8) 打开串口监视器查看输出数据;

(9) 前后左右上下移动被测物体, 查看数据输出情况;

(10) 测试最小检测框对算法的影响作用;

(11) 测试强光, 逆光, 暗光, 反光, 黄光等环境对颜色检测的影响。

测试数据与预期结果:

Frame: 帧号, 连续输出 1 ~ 100 的帧号, 没有跳帧

Rate: 识别率, 识别到物体次数/总运行次数, 不同算法性能下会有所差异

Time: 当前帧算法处理时间

Fps: 算法运行平均帧率

Detected (检测到) /undetected (未检测到): 是否检测到目标物体

Label: 颜色分类标签编号, 应与被测颜色一致

X: 目标物体中心 x 坐标 (0 ~ 100%), 物体左右移动应符合变化趋势

Y: 目标物体中心 y 坐标 (0 ~ 100%), 物体上下移动应符合变化趋势

Width: 目标物体宽度 (0 ~ 100%), 物体前后移动应符合变化趋势

Height: 目标物体高度 (0 ~ 100%), 物体前后移动应符合变化趋势

4.6 颜色识别算法测试

测试目的：颜色识别需先指定一个识别区域，包括坐标和宽高，然后返回该区域的平均颜色信息。

- (1) 打开 test/06_ColorRecognition.ino 项目
- (2) 设置识别区域的 CENTER_X (中心 X 坐标), CENTER_Y (中心 Y 坐标), WIDTH (宽), HEIGHT (高), 其范围为 0 ~ 100, 左上角为坐标原点 (0, 0);
- (3) 设置算法性能 VISION_LEVEL;
- (4) 设置颜色识别算法;
- (5) 该测试默认采用锁定白平衡方式, 因此开机时需要面向白纸校准白平衡;
- (6) 选择通讯方式和小 MU 地址;
- (7) 打开串口监视器查看输出数据;
- (8) 测试不同的颜色物体, 查看数据输出情况;
- (9) 前后左右上下移动摄像头, 查看数据输出情况;
- (10) 测试识别框对算法的影响作用;
- (11) 测试强光, 逆光, 暗光, 反光, 黄光等环境对颜色识别的影响。

测试数据与预期结果:

Frame: 帧号, 连续输出 1 ~ 100 的帧号, 没有跳帧
Rate: 识别率, 识别到物体次数/总运行次数, 不同算法性能下会有所差异
Time: 当前帧算法处理时间
Fps: 算法运行平均帧率
Label: 颜色分类标签编号, 应与被测颜色一致
R: 目标物体红色值 (0 ~ 255)
G: 目标物体绿色值 (0 ~ 255)
B: 目标物体蓝色值 (0 ~ 255)

4.7 多类型卡片算法测试

测试目的：测试三种卡片算法同时工作的情况。

- (1) 打开 test/07_MultiCard.ino 项目
- (2) 设置 VISION_TYPE, 同时开启形状卡片识别, 交通卡片识别, 数字卡片识别;
- (3) 该测试默认算法性能为准确率优先模式;
- (4) 选择通讯方式和小 MU 地址;
- (5) 打开串口监视器查看输出数据;
- (6) 测试单类卡片, 查看数据输出情况;
- (7) 测试多类卡片, 查看数据输出情况;

测试数据与预期结果:

Frame: 帧号, 连续输出 1 ~ 100 的帧号, 没有跳帧
Shape card: 形状卡片检测结果, 与被测卡片一致
Traffic card: 交通卡片检测结果, 与被测卡片一致
Number card: 数字卡片检测结果, 与被测卡片一致

修订历史

日期	版本	发布说明
2019.06.22	V0.1	起草
/	/	/

杭州摩图科技有限公司