V-function

功能介绍

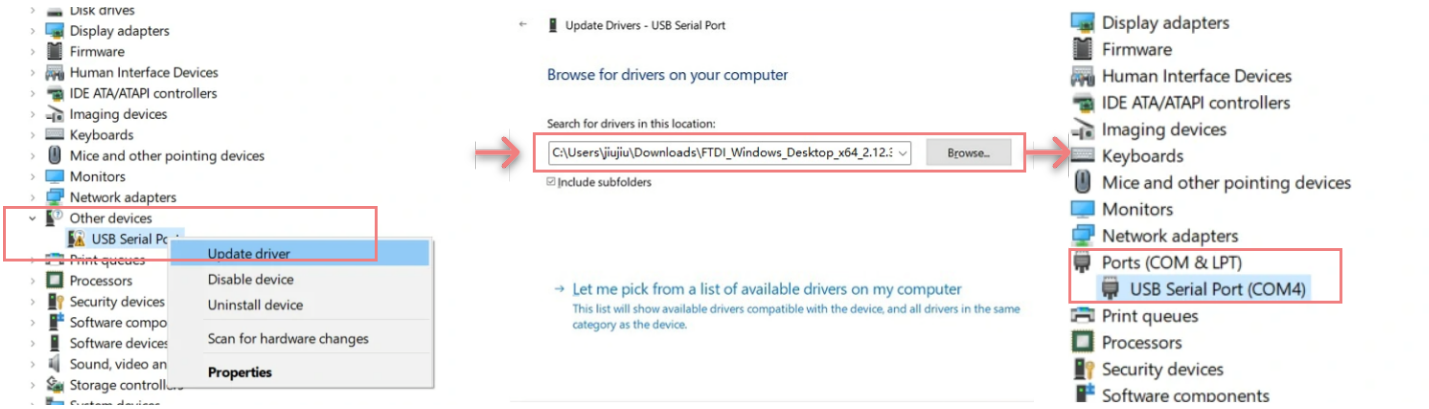
V-Function 是由 M5Stack 团队针对`M5StickV/UnitV`设备开发的多个`视觉识别`功能固件，摄像头在完成视觉识别后通过`串口输出(115200bps)识别结果`，基于不同的功能固件 (对象追踪，移动检测等), 用户能够快速的进行视觉识别的功能的搭建。本教程将向你介绍，如何烧录固件至你的设备中，并通过 UiFlow 图形化编程进行调用。

驱动安装

1. 在下方列表选择匹配操作系统的驱动文件进行下载，并解压。也可点击 FTDI Chip官方网站，获取更多版本的 FTDI 驱动。

驱动名称	下载链接
Windows_Desktop_x64	Download
Windows_Desktop_x86_32	Download
Windows_Desktop_ARM	Download
MacOS_Intel	Download
MacOS_ARM64	Download

2. 将设备连接至 PC，打开设备管理器为设备安装 FTDI 驱动（下图以 Windows 10 环境为例）。（注意：某些系统环境下，需要安装两次，驱动才会生效，未识别的设备名通常为`M5Stack`或`USB Serial`。）



MacOS 用户注意事项

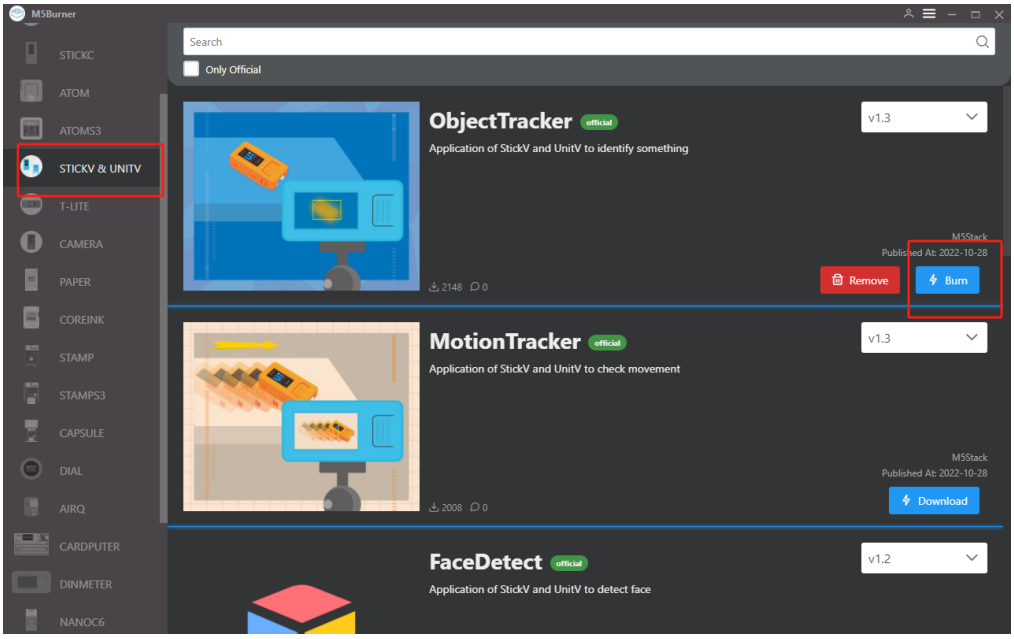
对于 MacOS 用户安装前请勾选`系统偏好设置` - > `安全性与隐私` - > `通用` - > `允许以下位置下载的App` - > `App Store和认可的开发者选项`。

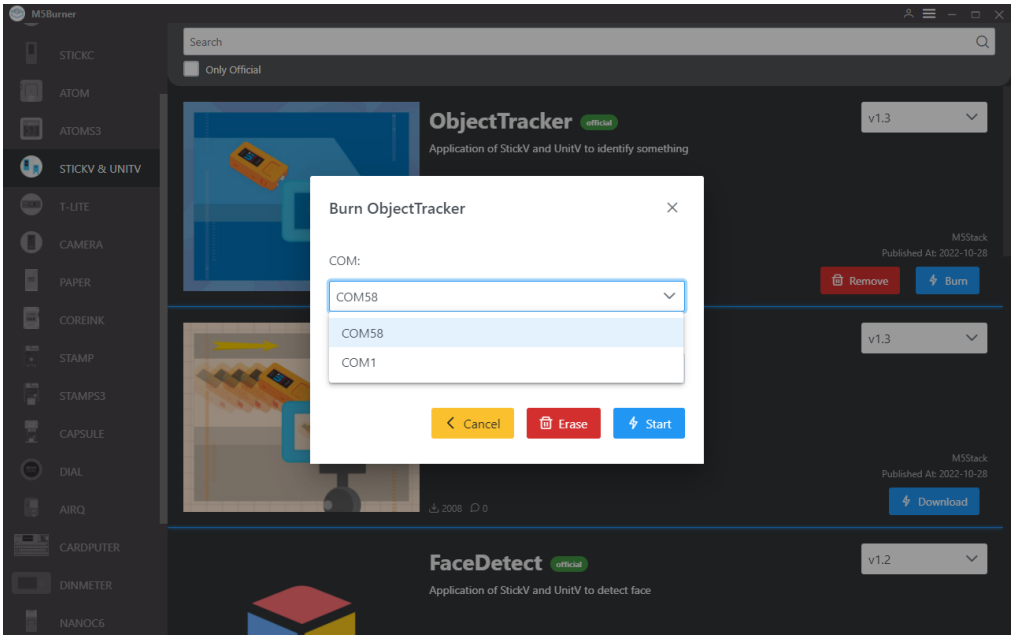
烧录固件

请根据您所使用的操作系统，点击下方按钮下载相应的 M5Burner 固件烧录工具。解压打开应用程序。

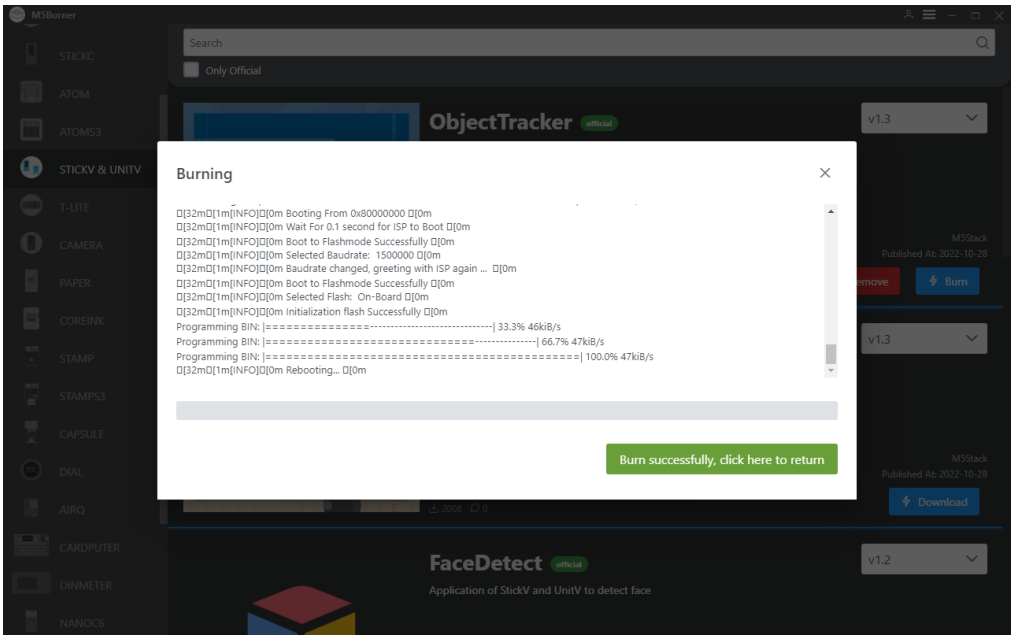
软件版本	下载链接
M5Burner_Windows	Download
M5Burner_MacOS	Download
M5Burner_Linux	Download

左侧设备栏选择设备为 M5StickV/UnitV, 根据使用需求选择对应的功能固件，进行下载。 将 M5StickV/UnitV 通过数据线连接至电脑，选择其对应的端口，点击 Burn 开始烧录。





当烧录日志提示`Burn Successfully`时，则表示固件已经烧录完成。



UiFlow 引用

烧录完功能固件的 M5StickV/UnitV 将作为 Unit 形式的从设备进行使用，因此用户需要使用其他的 M5 主机设备来与其交互。关于其他主控产品的 UiFlow 基本使用与操作，请访问其对应的产品文档页面进行获取。

访问https://flow.m5stack.com/ 进入 UiFlow。 点左侧功能面板中的 Unit 添加按钮，选中 UnitV 拓展进行添加。添加时，请根据实际使用的端口，进行配置。点击 ok 进行添加。

添加完成后，在功能块菜单中的 Unit 选项，即可找到包含的功能块。将其拖拽至右侧编程区域，即可进行使用。详细的 Block 功能说明，请点击下方 UiFlow Block 文档查看。

- UiFlow1 UnitV Blockly Docs

运动目标检测

检测当面画面的变化情况，判断检测区域内的物体是否存在运动。

运动目标检测 - 数据包格式

返回数据 JSON

```
{
    "FUNC": "MOTION DETECT V1.0",
    "DIFF TOTAL": 10000, //画面变动率
    "DIFF MAX": 75, // 最高变化率
    "TOTAL": 3, //边界框数量
    "0": {
        "x": 45,
        "y": 18,
        "w": 126,
        "h": 72,
        "area": 342 //该边界框内变化像素的数量
    },
    "1": {
        "x": 0,
        "y": 169,
        "w": 130,
```

```
        "h": 24,
        "area": 173
    },
    "2": {
        "x": 39,
        "y": 204,
        "w": 276,
        "h": 34,
        "area": 141
    }
}
```

设置 JSON

```
{
    "MOTION DETECT": 1.0, //功能标记,不可缺省
    "mode": "COMPUTE_MODE_STATIC", //可缺省 "COMPUTE_MODE_STATIC" 静态检测模式 or "COMPUTE_MODE_DYNAMIC" 动态检测
    "thr_w": 20, //可缺省 边界框宽阈值,[3,200]
    "thr_h": 20, //可缺省 边界框长阈值,[3,200]
    "stepx": 1, //可缺省 X扫描间隔,[0, 40],设置为0则关闭边界框检测
    "stepy": 2, //可缺省 Y扫描间隔,[0, 40],设置为0则关闭边界框检测
    "delta": 20, //可缺省 变化率阈值,[0, 99]
     "merge": 10 //可缺省 边界框合并阈值,[0, 40]
}
```

目标追踪

设置追踪目标，实时获取目标对象处于画面中位置信息。

目标追踪 - 数据包格式

返回数据 JSON

```
{
    "FUNC": "TARGET TRACKER V1.0",
    "x": 282,
    "y": 165,
    "w": 13,
    "h": 15
}
```

设置 JSON

```
{
    "TARGET TRACKER": " V1.0",
    "x": 282, //xywh均不可缺省
    "y": 165,
    "w": 13,
    "h": 15
}
```

颜色追踪

设置 LAB 颜色阈值，追踪画面中符合阈值目标，并实时获取目标对象处于画面中位置信息。

颜色追踪 - 数据包格式

返回数据 JSON

```
{
    "FUNC": "COLOR TRACKER V1.0",
    "TOTAL": 3, //边界框数量
    "0": {
        "x": 45,
        "y": 18,
        "w": 126,
        "h": 72,
        "area": 342 //该边界框内变化像素的数量
    },
    "1": {
        "x": 0,
        "y": 169,
        "w": 130,
        "h": 24,
        "area": 173
    },
    "2": {
        "x": 39,
        "y": 204,
        "w": 276,
        "h": 34,
        "area": 141
    }
}
```

设置 JSON

```
{
    "COLOR TRACKER": 1.0, //功能标记,不可缺省
    "thr_w": 20, //可缺省 边界框宽阈值,[3,200]
    "thr_h": 20, //可缺省 边界框长阈值,[3,200]
    "stepx": 1, //可缺省 X扫描间隔,[0, 40],设置为0则关闭边界框检测
    "stepy": 2, //可缺省 Y扫描间隔,[0, 40],设置为0则关闭边界框检测
    "merge": 10, //可缺省 边界框合并阈值,[0, 40]
    "Lmin": 0, //可缺省 L阈值下限 [0, 100]
    "Lmax": 0, //可缺省 L阈值上限 [0, 100]
    "Amin": 0, //可缺省 A阈值下限 [0, 255]
    "Amax": 0, //可缺省 A阈值上限 [0, 255]
    "Bmin": 0, //可缺省 B阈值下限 [0, 255]
    "Bmax": 0, //可缺省 B阈值上限 [0, 255]
}
```

人脸识别

识别画面中的人脸信息，并返回识别个数，对象坐标，置信率。

人脸识别 - 数据包格式

返回数据 JSON


```
{
   "FUNC": "FACE DETECT",  // 功能说明
   "count": 3,   // 识别到的人脸数量
   "2": {  // 人脸编号
      "x": 97,    // ROI
      "y": 26,
      "w": 64,
      "h": 86,
      "value": 0.859508,  // 置信率
       "classid": 0,
       "index": 2,
       "objnum": 3
        },
    "1": {
      "x": 70,
      "y": 157,
      "w": 38,
      "h": 63,
      "value": 0.712100,
      "classid": 0,
      "index": 1,
      "objnum": 3
      },
    "0": {
      "x": 199,
      "y": 145,
      "w": 31,
      "h": 40,
      "value": 0.859508,
      "classid": 0,
      "index": 0,
      "objnum": 3
      }
    }
```


二维码识别

识别画面中的二维码，并返回识别结果，以及版本。使用固件`Find code`

返回数据 JSON


```
{
   "count": 1,
   "FUNC": "FIND QRCODE",
   "0": {
      "x": 57,
      "y": 16,
```

```
      "w": 197,
      "h": 198,
      "payload": "m5stack",    //二维码数据
      "version": 1,    //二维码版本
      "ecc_level": 1,    //二维码ECC水平
      "mask": 2,    //二维码掩码
      "data_type": 4,    //二维码数据类型
      "eci": 0    //返回二维码的ECI。
   }
}
```

条形码识别

识别画面中的条形码，并返回识别结果，以及版本。使用固件`Find code`

返回数据 JSON

```
{
    "0": {
        "x": 62,
        "y": 90,
        "w": 100,
        "h": 45,
        "payload": "123", //数据
        "type": 15, //条码类别
        "rotation": 0.000000, //条码旋转角度
        "quality": 28 //条码在图像中被扫描的次数
    },
    "count": 1,
    "FUNC": "FIND BARCODE"
}
```

Datamatrix 码识别

识别画面中的 Datamatrix 码，并返回识别结果，以及码旋转角度，坐标数据。使用固件`Find code`

返回数据 JSON

```
{
    "0": {
        "x": 20,
        "y": 116,
        "w": 96,
        "h": 96,
        "payload": "m5stack",
        "rotation": 1.588250, //DM码旋转角度
        "rows": 16, //DM码行数
        "columns": 16, //DM码列数
        "capacity": 12, //DM码容量(字节)
        "padding": 1 //DM码剩余容量(字节)
    },
    "count": 1,
```

```
    "FUNC": "FIND DATAMATRIX"
}
```

Apriltag 码识别

识别画面中的 Apriltag 码 (仅支持 Tag36H11 类型), 并获取其位置的偏移。使用固件`Find code`

返回数据 JSON

```
{
    "0": {
        "x": 71,
        "y": 5,
        "w": 85,
        "h": 88,
        "id": 1,
        "family": 16,// AprilTag的类别
        "cx": 115,
        "cy": 49,
        "rotation": 6.219228,// 返回以弧度计的AprilTag的旋度(int)。
        "decision_margin": 0.451959,// AprilTag匹配的色饱和度(取值0.0 - 1.0),其中1.0为最佳。
        "hamming": 0,// AprilTag的可接受的数位误差数值
        "goodness": 0.000000, //AprilTag图像的色饱和度
        "x_translation": 0.868200, //旋转后将图像移动到左侧或右侧的单位数
        "y_translation": 0.245313,//旋转后将图像上移或下移的单位数
        "z_translation": -2.725188,//是通过图像缩放的量。默认情况下1.0
        "x_rotation": 3.093776,//x轴在帧缓冲器中旋转图像的度数
        "y_rotation": 0.065489,//y轴在帧缓冲器中旋转图像的度数
        "z_rotation": 6.219228 //z轴在帧缓冲器中旋转图像的度数
    },
    "count": 1,
    "FUNC": "FIND APRILTAG"
}
```

识别模式设置 JSON

以上多个识别码功能，均使用同一个固件`Find Code`实现，用户可以通过发送下方 JSON 数据，配置模式切换。

```
{
    "FIND CODE": 1.0,
    "mode":"DATAMATRIX" //识别模式,可选QRCODE,APRILTAG,DATAMATRIX,BARCODE
}
```

自定义标签识别

检测画面中的标签卡，并返回二进制序列。注：仅识别固定标签卡格式，请参考下方图片

自定义标签识别 - 数据包格式

返回数据 JSON

```
{
    "FUNC": "TAG READER V2.0",
    "TOTAL": 1,
    "0": {
        "x": 113,
        "y": 65,
        "w": 117,
        "h": 105,
        "p0x": 113, // p0x ~ p3y: TAG 4个顶点的坐标
        "p0y": 77,
        "p1x": 211,
        "p1y": 65,
        "p2x": 230,
        "p2y": 156,
        "p3x": 127,
        "p3y": 170,
        "rotation": 8, // TAG 的相对旋转角度
        "rows": 8, // TAG 的行数(本数值不含定位框)
        "columns": 8, // TAG 的列数(本数值不含定位框)
        "size": 64, // TAG 实际内容的数据长度,该值 = 内容的行数 * 内容的列数 = (rows) * (columns)
        "code": "0x003C42425A424200", // uint64_t类型的内容二进制代码,本键值最大编码64位(8 x 8)的TAG
        "binstr": "0000000000111100010000100100001001011010010000100100001000000000" //二进制数据的字符串形式,本
    }
}
```


巡线

检测画面中指定的颜色线条，并返回偏移角度。

巡线 - 数据包格式

返回数据 JSON


```
{
    "FUNC": "LINE TRACKER V1.0",
    "angle": 3.8593475818634033 //小车转弯的角度
}
```


设置 JSON


```
{
    "LINE  TRACKER": 1.0, //功能标记,不可缺省
    "thr_w": 20, //可缺省 边界框宽阈值,[3,200]
    "thr_h": 20, //可缺省 边界框长阈值,[3,200]
    "stepx": 1, //可缺省 X扫描间隔,[0, 40],设置为0则关闭边界框检测
    "stepy": 2, //可缺省 Y扫描间隔,[0, 40],设置为0则关闭边界框检测
    "merge": 10, //可缺省 边界框合并阈值,[0, 40]
    "Lmin": 0, //可缺省 L阈值下限 [0, 100]
    "Lmax": 0, //可缺省 L阈值上限 [0, 100]
    "Amin": 0, //可缺省 A阈值下限 [0, 255]
    "Amax": 0, //可缺省 A阈值上限 [0, 255]
    "Bmin": 0, //可缺省 B阈值下限 [0, 255]
```

```
    "Bmax": 0, //可缺省 B阈值上限 [0, 255]
    "weight_0": 0.1, // 可缺省 权重
    "weight_1": 0.3, // 可缺省 权重
    "weight_2": 0.7  // 可缺省 权重
}
```

相关内容

Github

常见问题

使用将从设备 (M5StickV/UnitV) 连接至主控后，主控端若存在数据获取不正常情况，请重启 M5StickV/UnitV。等待固件启动成功后重新尝试连接。