Morpx Documentation

Morpx

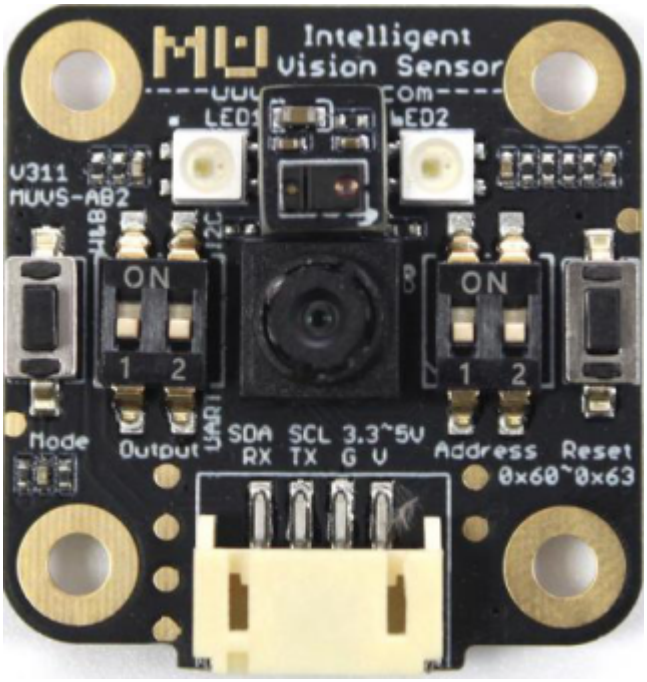
2020 年 06 月 09 日

MU Vision Sensor 3 开发教程

CHAPTER 1

MU Vision Sensor 3 简介

MU 视觉传感器 3（MU Vision Sensor 3）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，体积小巧，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。



1.1 硬件设置

1.1.1 1. 设置通讯模式

MU 支持 4 种通讯模式: `UART,I2C,WIFI,` 图传模式。根据所需要的通讯方式, 拨动 MU 左侧的 `Output` 拨码开关。

选择通讯方式后, 程序中的通讯方式应与拨码开关保持一致。编程时应首先配置通讯方式, 然后才可以进行其他的参数配置, 使用过程中不可更改, 每次切换通讯方式, 需要重启小 MU。

输出模式	拨码开关	编号	LED 指示
UART		00	闪烁红色
I2C		01	闪烁绿色
WiFi		10	闪烁黄色
图传		11	闪烁紫色

1.1.2 2. 设置地址

MU 支持 4 个地址:0x60(默认),0x61,0x62,0x63。当 MU 与其他传感器地址冲突时需要进行更改。I2C 模式下支持多个不同地址的 MU 协同工作, 可以给 MU 分配不同的地址。

注解: 一般情况下使用默认地址 `0x60` 即可。

设备地址	拨码开关	编号	设备地址	拨码开关	编号
0x60		00	0x61		01
0x62		10	0x63		11

1.1.3 3. 线路连接

UART/WiFi/图传模式

MU	RX	TX	G	V
主控	TX	RX	GND	5V

I2C 模式

MU	SCL	SDA	G	V
主控	SCL	SDA	GND	5V

1.2 软件设置

详见本目录下各平台对应教程。

1.3 特殊模式介绍

1.3.1 WiFi/图传模式配网

WiFi/图传模式可通过向 MU 发送 AT 指令的方式进行配网，串口默认波特率为 9600。

可通过输入以下指令获取所有 AT 指令：

```
AT+HELP
```

注意： 所有指令必须以 "\r\n" 或 ' ' 结尾

MU 可支持 AP 模式联网和 STA 模式联网，两种联网区别如下：

AP 模式 AP 模式为 MU 默认的 WiFi 模式，该模式下 MU 会生成一个 WiFi 热点，用户使用手机或电脑去连接此热点即可。WiFi 成功连接后，MU 的 LED 就会熄灭。

默认的热点名称为 MORPX-MU-AB 。

注解： WiFi 名称中 A 为 MU 左侧 LED 颜色的首字母，B 为右侧 LED 颜色的首字母。

（如：左侧 LED 为红色 **R** ed，右侧为黄色 **Y** ellow，则默认 WiFi 名称为 MORPX-MU-RY）

若需要自定义 WiFi 名称，可通过串口发送以下指令进行配置：

```
AT+WIFISET=<yourSSID>,<yourPassword>,AP
AT+WIFICON=1
```

若设置成功，则返回：

```
OK
wifi ap mode starting...
OK
```

STA 模式 STA 模式需要 MU 和用户的设备去连接一个公共的 WiFi，以实现二者的互联。可通过串口发送以下指令进行配置：

```
AT+WIFISET=<yourSSID>,<yourPassword>,STA
AT+WIFICON=1
```

> **注意：** <yourSSID> 和 <yourPassword> 必须是一个已存在的 WiFi（区分大小写），否则会连接失败。

若设置成功，则返回：

```
OK
wifi sta mode connecting...
OK
```

1.3.2 图传模式图像查看

将 MU 设置成图传模式及完成配网 后，可通过打开网址 192.168.4.1 查看图像。

1.3.3 无线透传

WiFi/图传模式皆可进行无线透传，完成配网 后，可通过以下方式来进行无线透传：

注解： 因为手机、电脑不同平台，不同操作系统下 TCP/UDP 调试软件各不相同，软件的设置方式大同小异，这里定义以下几个名词：

- 本地 IP：即 MU 的 IP 地址
- 目标 IP：即 MU 需要发送消息的目标设备的 IP 地址

---

1. 打开 TCP/UDP 调试软件，选择 UDP，将模式设置为 Unicast

2. 查询本地 IP，通过串口向 MU 发送指令：

```
AT+WIFISIP
```

返回 MU 的本地 IP。

3. 将 TCP/IP IP 栏设置为 MU 的本地 IP，端口设置为 3333

注解： STA 模式下路由会为 MU 和目标设备随机分配一个 IP 地址，需要通过以下方式配置：

1. 查询目标 IP （大多 TCP/IP 软件会显示当前设备的 IP 地址）

2. 通过串口向 MU 发送指令：

```
AT+WIFIUDP=<targetIP>,3333
```

返回：

```
OK
```

至此，WiFi 配置完毕，TCP/UDP 调试软件发送的所有数据会通过 MU 的串口转发出来，通过串口向 MU 发送的所有数据也会在 TCP/UDP 调试软件的监视器上显示出来。

CHAPTER 2

MU 3 Mixly 教程

本文介绍 MU Vision Sensor 3 配合 Arduino 开发板在米思奇（Mixly）开发环境下的开发教程。

Mixly 是一款图形化编程软件，用户可以通过拼接积木块的方式来编写程序。

Mixly 基础教程请参考官方帮助文档 Mixly 帮助文档

2.1 Mixly 开发环境搭建

2.1.1 完整安装包下载

对于未使用过 Mixly 的用户，可以下载完整的定制 Mixly 安装包，已包含 MU Vision Sensor 3 和 MoonBot Kit 的库。

Windows/Linux/Mac 完整版 MoonBot Mixly 安装包下载地址：https://pan.baidu.com/s/1h8Cuj8UYm99Mh3O1ppmMfg，提取码：`ksme`

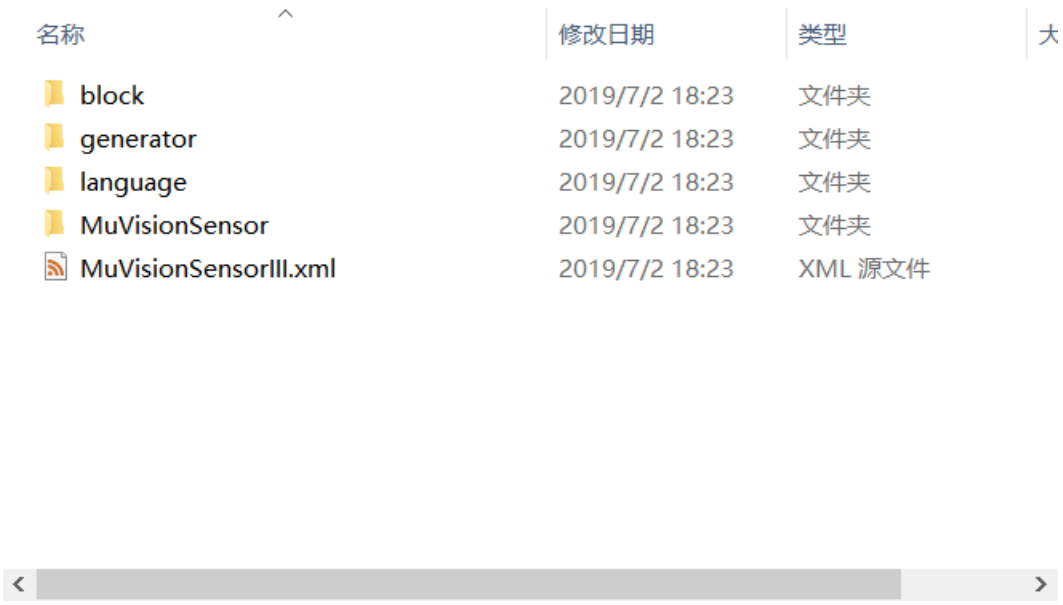
2.1.2 Mixly 库导入

对于已安装 Mixly 的用户，可通过导入 MU 的库来支持编程。此方法同样适用于其他第三方库的导入。

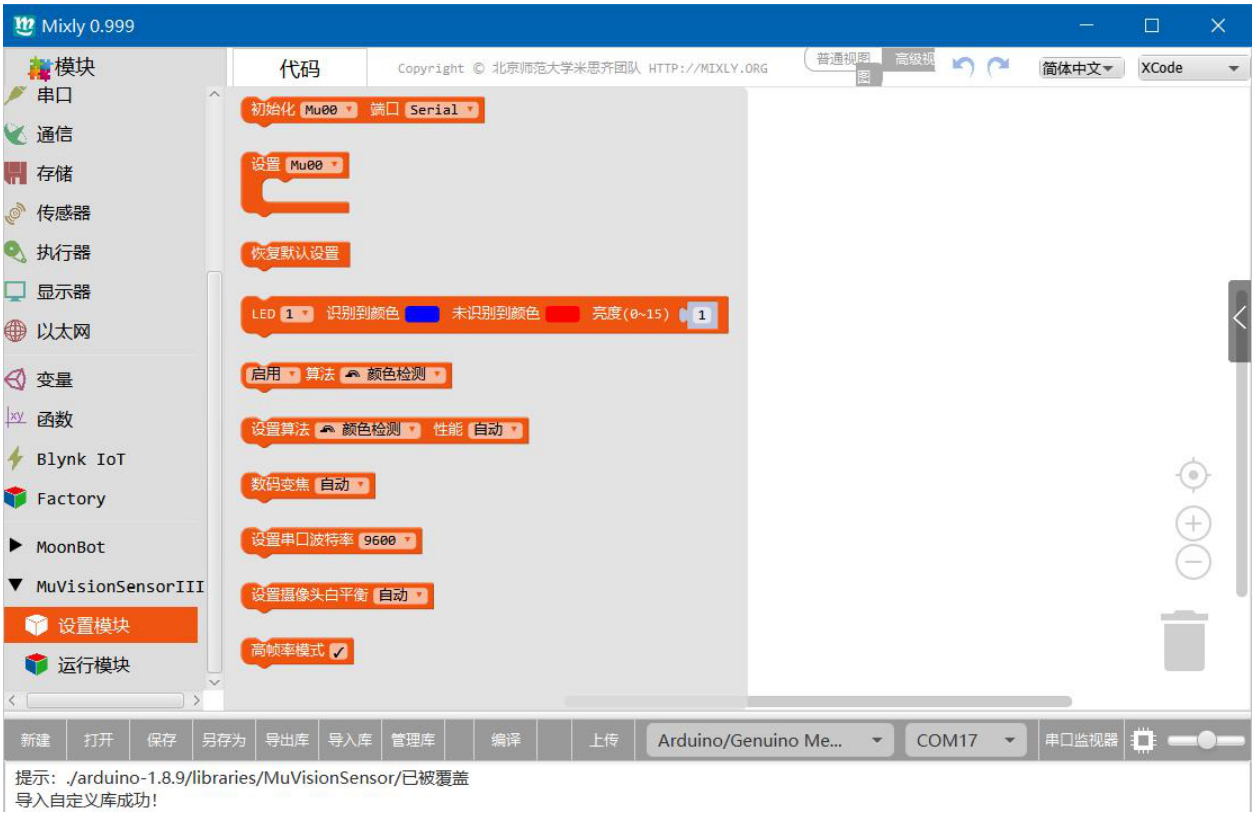
打开 Mixly 软件，选择主控设备，常用 Arduino Uno。如果使用 MoonBot 主控，则选择 Arduino Mega（atmega 1280），选择设备的 COM 口。

点击导入库。

选中库中名为 MUVisionSensor3.xml 文件，点击打开。

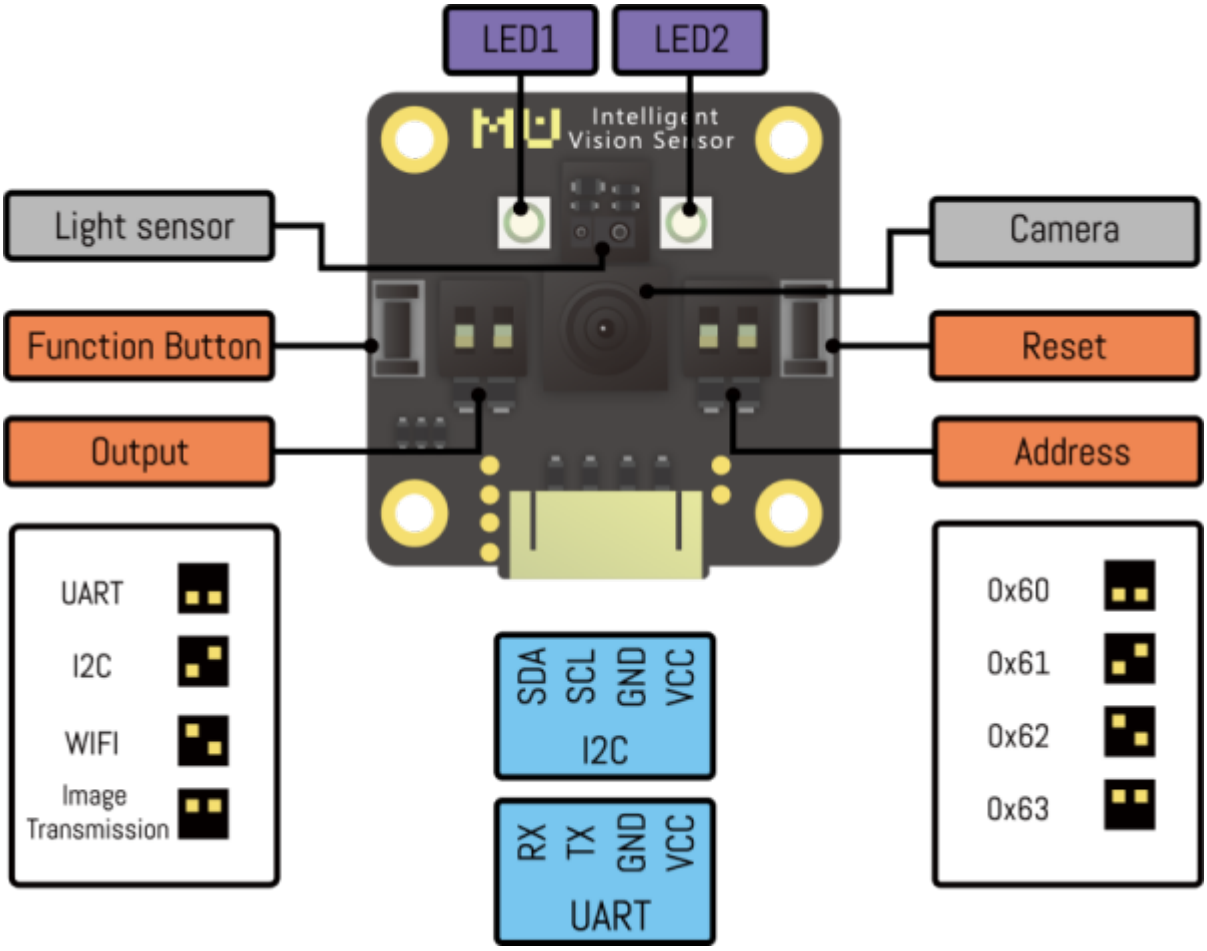


若在 Mixly 导航栏出现 MUVisionSensor3 一栏，且下方出现导入自定义库成功，则导入成功。



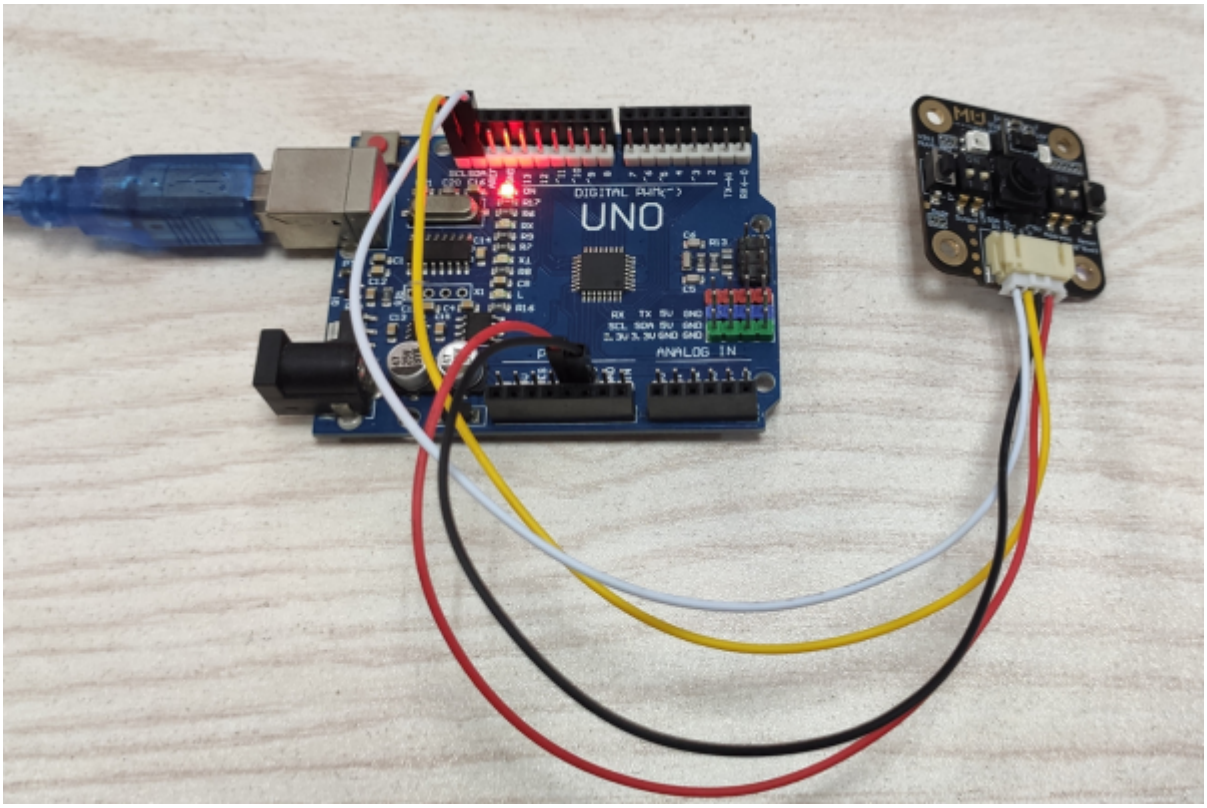
2.2 Arduino 硬件连接

MU Vision Sensor 3 的外设和接口如图所示:



2.2.1 I2C 模式

1. 将模块左侧输出模式拨码开关 1 拨至下方，2 拨至上方，切换至 I2C 模式；
2. (不推荐修改此设置) 将模块右侧的地址选择拨码开关拨至对应位 (默认地址 0x60，1、2 都在下方)；
3. 将模块输出接口 SDA 口接至 Arduino 对应的 SDA 口，SCL 口接至 Arduino 对应的 SCL 口。



2.2.2 串口模式

1. 将模块左侧输出模式拨码开关 1、2 都拨至下方，切换至串口模式；
2. (不推荐修改此设置) 将模块的地址选择拨码开关拨至对应位 (默认地址 1、2 都在下方)；
3. 将模块输出接口 RX 口接至 Arduino 对应的 TX 口，TX 口接至 Arduino 对应的 RX 口。

2.2.3 AT 指令模式（适用于 V1.1.5 及以上版本的固件）

1. 将模块左侧输出模式拨码开关 1 拨至下方，2 都拨至上方，切换至 AT 指令模式；
2. 将 MU 输出接口 RX 口接至 Arduino 对应的 TX 口，TX 口接至 Arduino 对应的 RX 口。

2.2.4 图传模式（适用于 V1.1.5 及以上版本的固件）

1. 将模块左侧输出模式拨码开关 1、2 都拨至上方，切换至图传模式；
2. 将 MU 输出接口 RX 口接至 Arduino 对应的 TX 口，TX 口接至 Arduino 对应的 RX 口。

2.3 模块说明

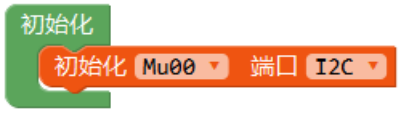
2.3.1 设置模块

初始化模块

1. 硬件串口：视觉传感器使用串口模式，连接主控的硬件串口时主控初始化，该串口为主控和电脑端的串口通信，用于视觉会占用，电脑端打印字符会错乱或通信异常；

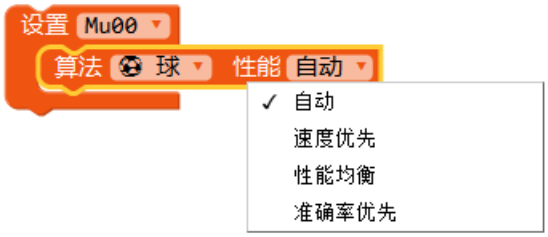
1. 软件串口：视觉传感器使用串口模式，连接主控的软件串口时主控初始化，主控可自定义 RX 和 TX 引脚，实际环境软串口速度太快可能不稳定，波特率不建议超过 9600；

1. 硬件 I2C：视觉传感器使用 I2C 模式，连接主控 I2C 引脚时主控初始化。



开启算法

设置算法性能

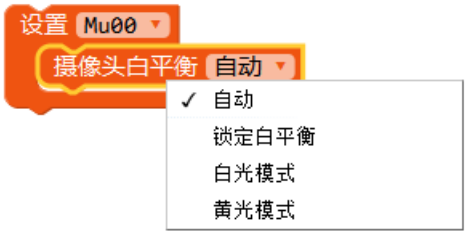


开启摄像头高帧率模式

识别速度增加，同时功耗、发热量增加。

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。



板载 LED 灯光设置

恢复模块默认设置

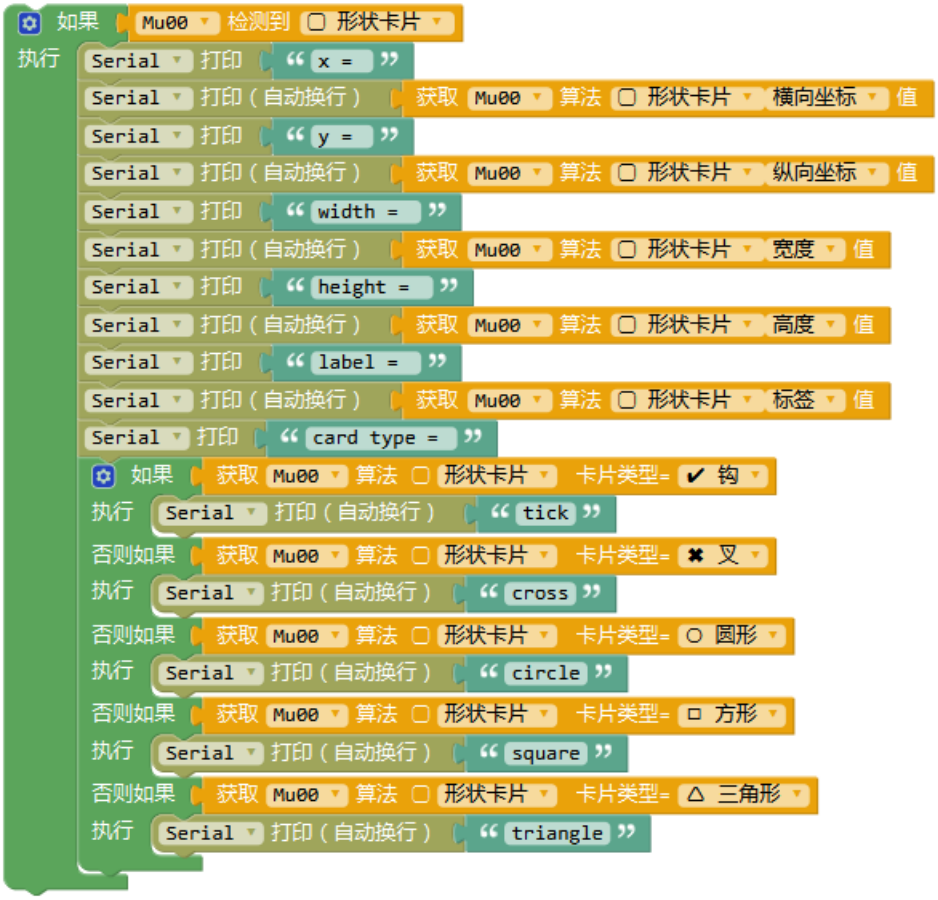
关闭所有算法，重置所有硬件设置。

2.3.2 运行模块

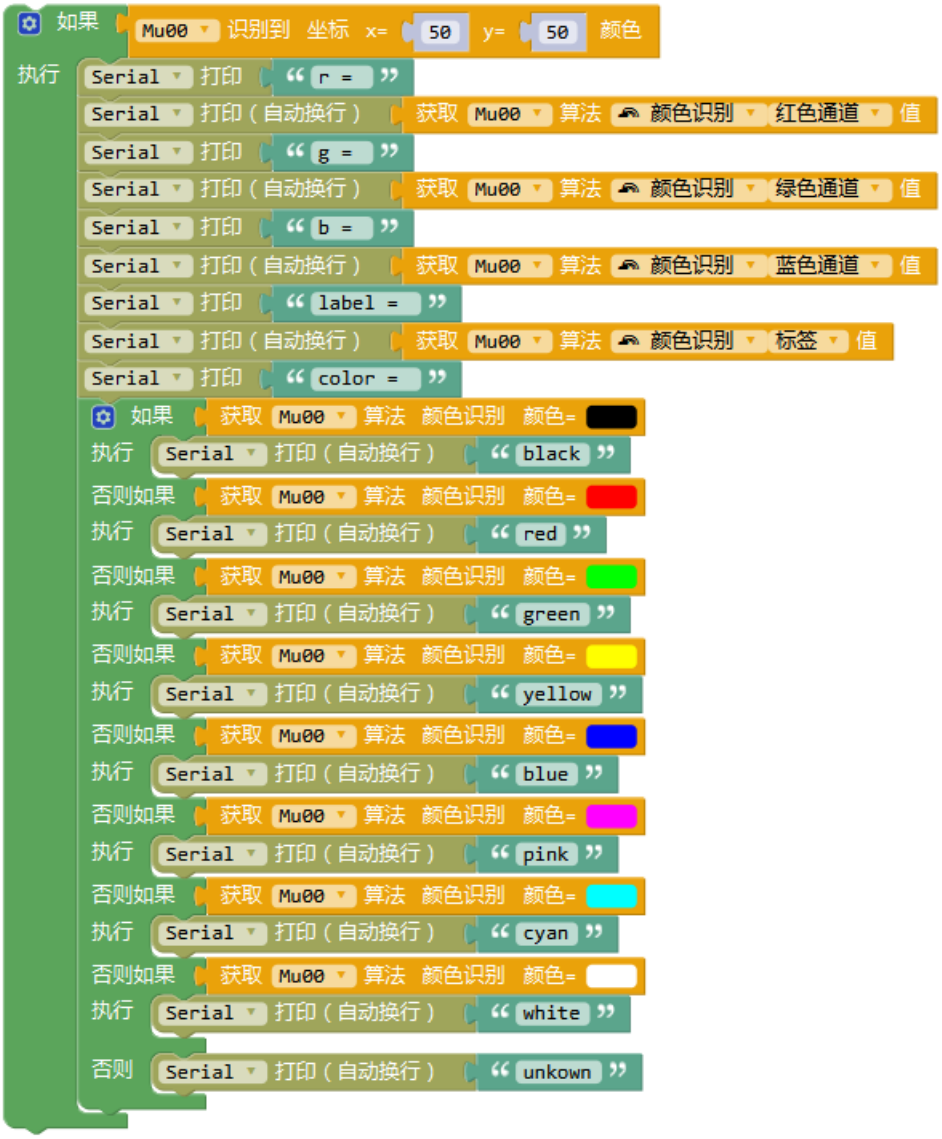
获取算法识别结果

1. 球、人体

1. 卡片


如果 Mu00 检测到 形状卡片
执行
Serial 打印 " x = "
Serial 打印（自动换行） 获取 Mu00 算法 形状卡片 横向坐标 值
Serial 打印 " y = "
Serial 打印（自动换行） 获取 Mu00 算法 形状卡片 纵向坐标 值
Serial 打印 " width = "
Serial 打印（自动换行） 获取 Mu00 算法 形状卡片 宽度 值
Serial 打印 " height = "
Serial 打印（自动换行） 获取 Mu00 算法 形状卡片 高度 值
Serial 打印 " label = "
Serial 打印（自动换行） 获取 Mu00 算法 形状卡片 标签 值
Serial 打印 " card type = "
如果 获取 Mu00 算法 形状卡片 卡片类型= ✔ 钩
执行 Serial 打印（自动换行） " tick "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= ✖ 叉
执行 Serial 打印（自动换行） " cross "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= ○ 圆形
执行 Serial 打印（自动换行） " circle "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= □ 方形
执行 Serial 打印（自动换行） " square "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= △ 三角形
执行 Serial 打印（自动换行） " triangle "

1. 颜色识别

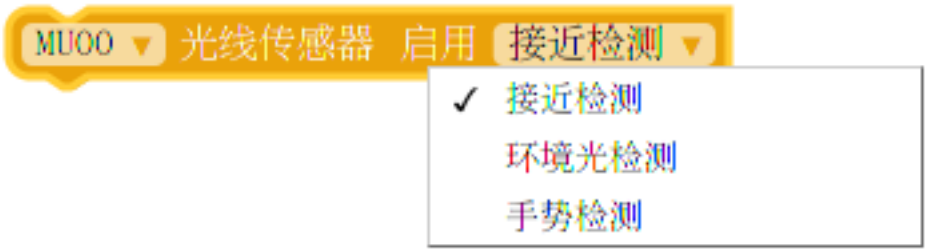

如果 Mu00 识别到 坐标 x= 50 y= 50 颜色
执行
Serial 打印 " r = "
Serial 打印 (自动换行) 获取 Mu00 算法 颜色识别 红色通道 值
Serial 打印 " g = "
Serial 打印 (自动换行) 获取 Mu00 算法 颜色识别 绿色通道 值
Serial 打印 " b = "
Serial 打印 (自动换行) 获取 Mu00 算法 颜色识别 蓝色通道 值
Serial 打印 " label = "
Serial 打印 (自动换行) 获取 Mu00 算法 颜色识别 标签 值
Serial 打印 " color = "
如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " black "
否则如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " red "
否则如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " green "
否则如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " yellow "
否则如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " blue "
否则如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " pink "
否则如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " cyan "
否则如果 获取 Mu00 算法 颜色识别 颜色=
执行 Serial 打印 (自动换行) " white "
否则 Serial 打印 (自动换行) " unkown "

1. 色块检测

2.3.3 光线传感器

启用功能

启用光线传感器中对应的功能，手势检测功能无法与其他功能同时工作。



设置灵敏度

设置光线传感器灵敏度，该功能只对接近检测、光线检测有效。

读取接近检测值

读取接近检测值，距离越近，返回值越大。

读取环境光检测值

读取环境光检测值，周围环境光越亮，返回值越大。

是否检测到手势

读取手势检测值，当未检测到时返回 0。

判断手势结果

比较手势检测结果是否为某一手势。

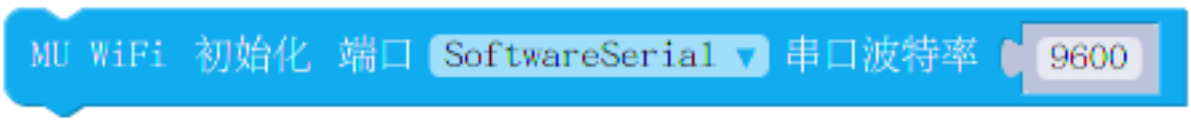
2.3.4 WiFi 模块

WiFi AT 指令设置

以下模块仅在图传/AT 指令模式下使用

WiFi 初始化端口

初始化 WiFi 对应的端口及其波特率



配置 WiFi

设置 WiFi 模式及账户密码

连接/创建 WiFi

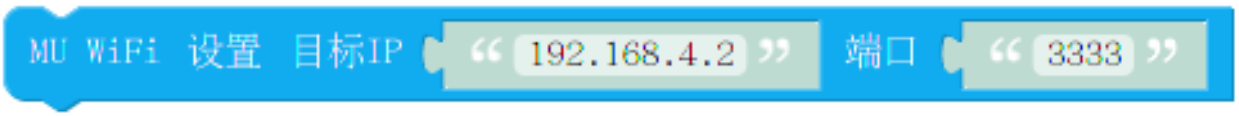
尝试连接/创建 WiFi，并返回当前连接状态

断开/关闭 WiFi

断开/关闭 WiFi

设置目标 IP

设置目标 IP



读取目标 IP

读取目标 IP

读取本地 IP

读取本地 IP

WiFi 读取

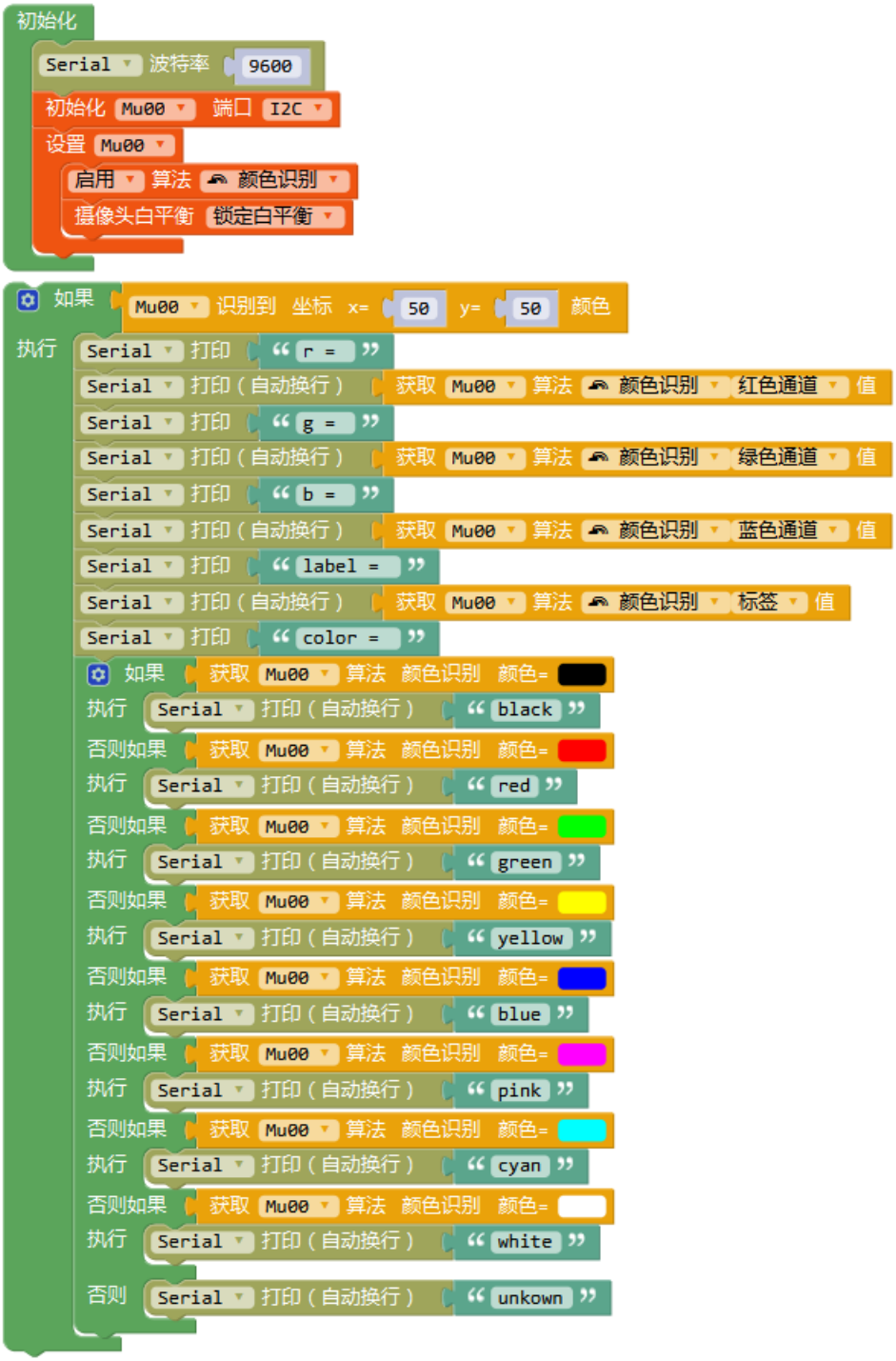
读取目标 IP 发送给 MU 的消息

WiFi 写入

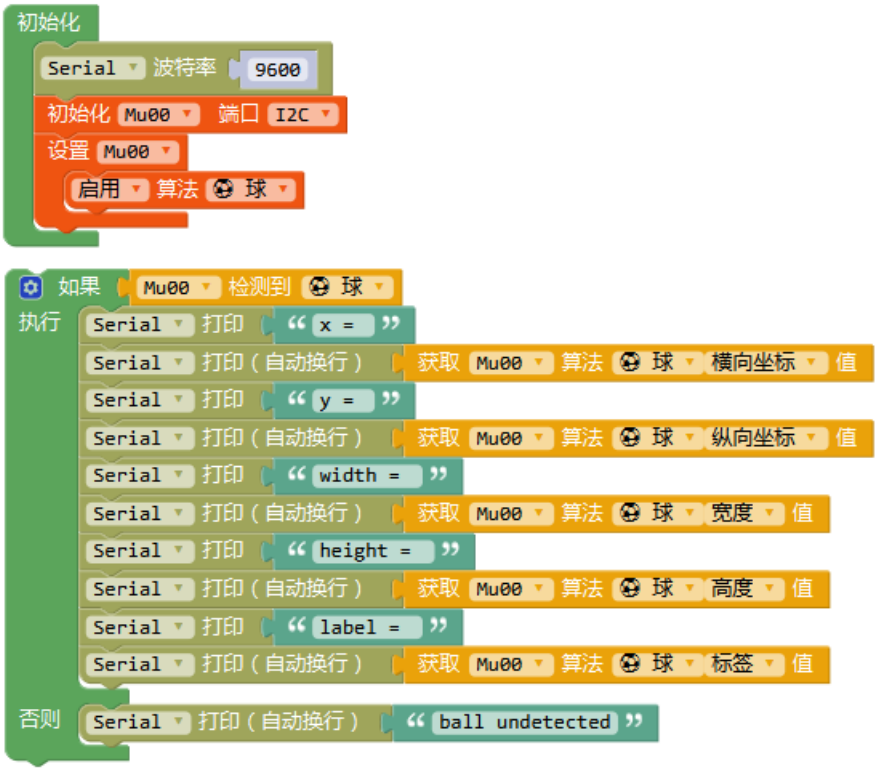
向目标 IP 发送消息

2.4 完整示例

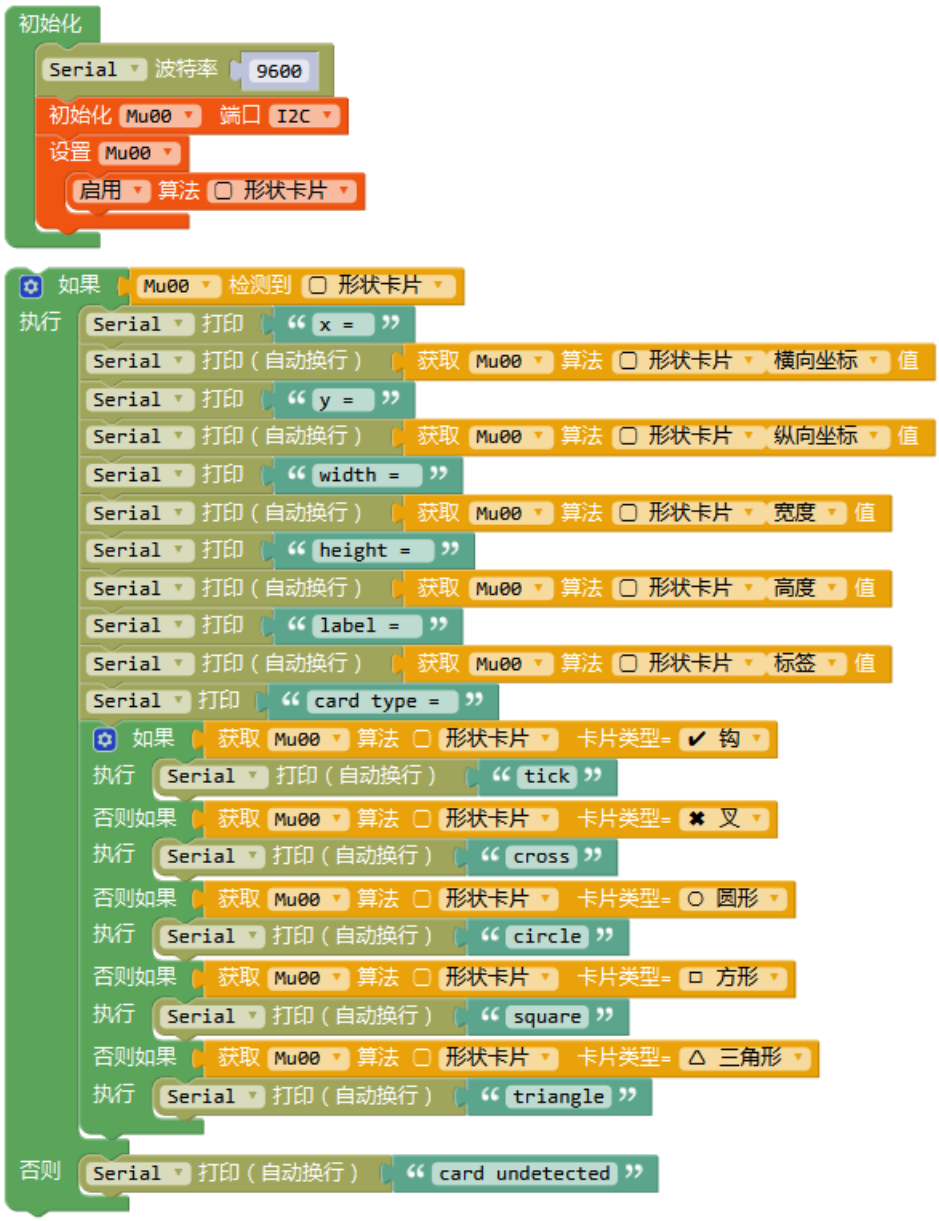
2.4.1 颜色识别



2.4.2 球体检测

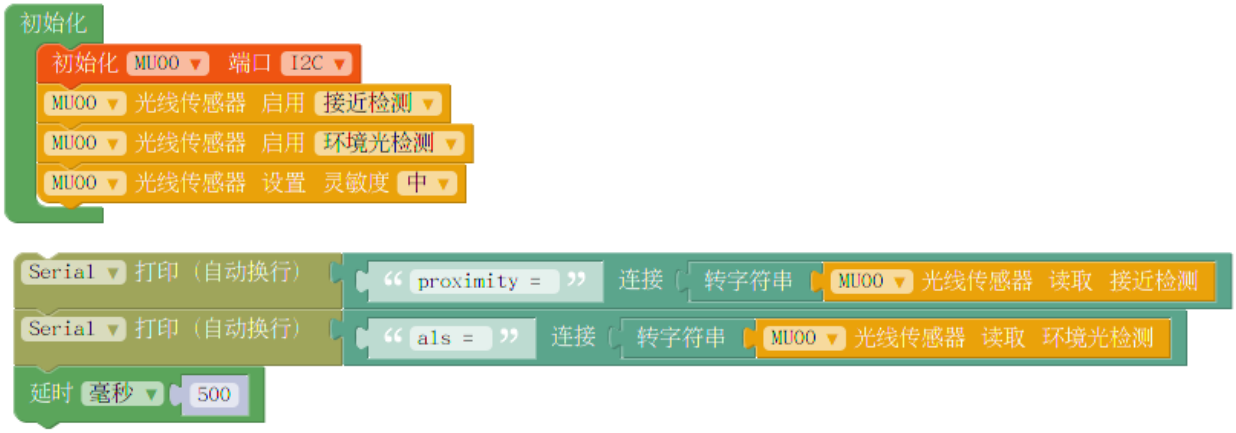

初始化
Serial 波特率 9600
初始化 Mu00 端口 I2C
设置 Mu00
启用 算法 球
如果 Mu00 检测到 球
执行
Serial 打印 " x = "
Serial 打印（自动换行） 获取 Mu00 算法 球 横向坐标 值
Serial 打印 " y = "
Serial 打印（自动换行） 获取 Mu00 算法 球 纵向坐标 值
Serial 打印 " width = "
Serial 打印（自动换行） 获取 Mu00 算法 球 宽度 值
Serial 打印 " height = "
Serial 打印（自动换行） 获取 Mu00 算法 球 高度 值
Serial 打印 " label = "
Serial 打印（自动换行） 获取 Mu00 算法 球 标签 值
否则
Serial 打印（自动换行） " ball undetected "

2.4.3 形状卡片检测

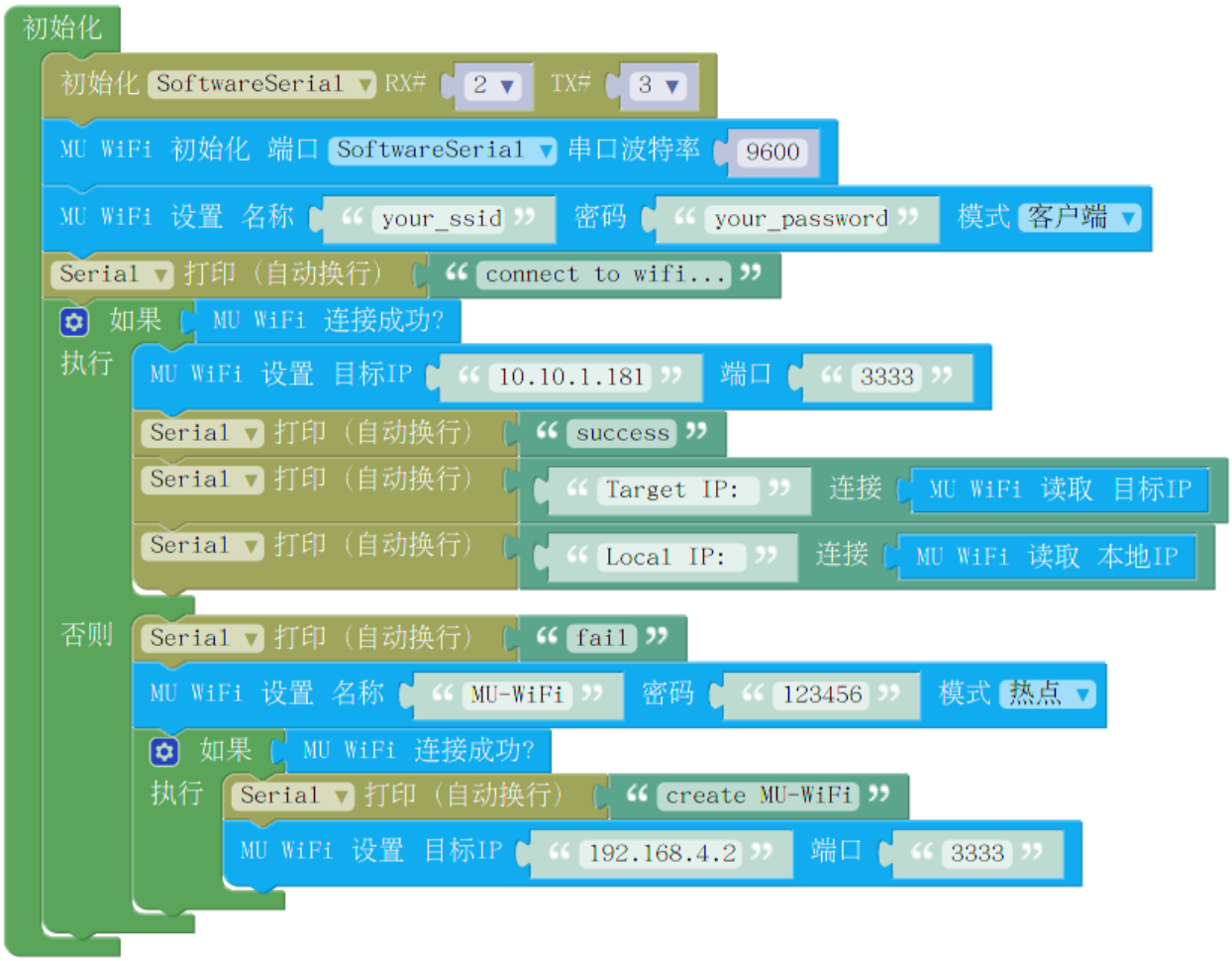

初始化
Serial 波特率 9600
初始化 Mu00 端口 I2C
设置 Mu00
启用 算法 形状卡片
如果 Mu00 检测到 形状卡片
执行
Serial 打印 " x = "
Serial 打印 (自动换行) 获取 Mu00 算法 形状卡片 横向坐标 值
Serial 打印 " y = "
Serial 打印 (自动换行) 获取 Mu00 算法 形状卡片 纵向坐标 值
Serial 打印 " width = "
Serial 打印 (自动换行) 获取 Mu00 算法 形状卡片 宽度 值
Serial 打印 " height = "
Serial 打印 (自动换行) 获取 Mu00 算法 形状卡片 高度 值
Serial 打印 " label = "
Serial 打印 (自动换行) 获取 Mu00 算法 形状卡片 标签 值
Serial 打印 " card type = "
如果 获取 Mu00 算法 形状卡片 卡片类型= ✔ 钩
执行 Serial 打印 (自动换行) " tick "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= ✖ 叉
执行 Serial 打印 (自动换行) " cross "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= ○ 圆形
执行 Serial 打印 (自动换行) " circle "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= □ 方形
执行 Serial 打印 (自动换行) " square "
否则如果 获取 Mu00 算法 形状卡片 卡片类型= △ 三角形
执行 Serial 打印 (自动换行) " triangle "
否则 Serial 打印 (自动换行) " card undetected "

2.4.4 光线传感器-手势识别

2.4.5 光线传感器-接近检测/环境光亮度检测



2.4.6 AT 指令连接 WiFi



2.5 常见问题

- **导入新版的库覆盖旧版的库后，调用任意模块都会报错怎么办？**

  这是由于旧版 Mixly 覆盖库时，直接覆盖对应的 Arduino 库，没有删除旧版库导致的；用户需要手动进入到 Mixly 安装目录下，进入 `arduino-xxxx/libraries/` 目录，手动删除 `MuVisionSensor/` 文件夹，重新导入库后才能使用。

  或下载最新版 Mixly，也可解决这个问题。

- **Mixly 部分例程打开后是空白的怎么办？**

  Mixly 不同版本编写的程序兼容性不佳，官方例程都是使用 Mixly 998(7.9) 版本书写，使用此版本即可打开官方例程。

- **导入库后无法打开模块或模块都是黑框怎么办？**

  请下载最新版 Mixly 程序，重新导入后即可。

- **我正确导入了库，下载了例程，但是模块没有反应，串口也没有任何输出怎么办？**

  1. 检查接线是否正确，是否有接触不良的现象。
  2. 检查模块背后的白灯是否常量，白灯不亮则表示电源口没有电压或电源线接线错误。
  3. 检查输出模式拨码开关和地址选择拨码开关是否是拨至正确位置。
  4. 模块从上电到初始化完成需要一段时间，建议在“设置”模块前加入一段不小于 500ms 的延时。
  5. 点击模块 Reset 按钮，模块正面两个 LED 会短暂闪烁一次光。红光则表示当前模式为串口模式，绿光则表示当前模式为 I2C 模式。若光的颜色与输出模式拨码开关不符，则可能为拨码开关松动，重新拨动拨码开关至正确位置即可。

- **我下载了程序，串口有正确的内容输出，但是 LED 灯光不亮怎么办？**

  1. 当算法为颜色识别算法时，程序默认会关闭 LED，防止 LED 灯光照射使物体偏色。
  2. 设置中打开 LED 灯光，调节灯光亮度大于 0。

- **我对比了文档和 Mixly 库，发现有些功能 Mixly 库没有怎么办？**

  为了使库易于理解和操作，在 Mixly 库中去除了一些不常用的功能，简化了一些参数的设置方式。若这些省去的功能影响到了您的项目，请发邮件至摩图科技售后：`support@morpx.com` 寻求技术支持和解决方案。

- **我发现烧录前一次程序的算法会对后一次程序的算法有影响，如：前一次烧录了颜色识别算法，后一次烧录球算法，就算没有识别到球 LED 也会显示识别到，而只烧录球算法则没有这种现象，怎么办？**

  这是因为前一种算法在程序结束后并没有被关闭导致的，可以在设置模块时加入“恢复默认设置”模块，或重新断电拔插模块即可，该问题在 V1.2.1 库以后已修复。

CHAPTER 3

MU 3 Arduino 教程

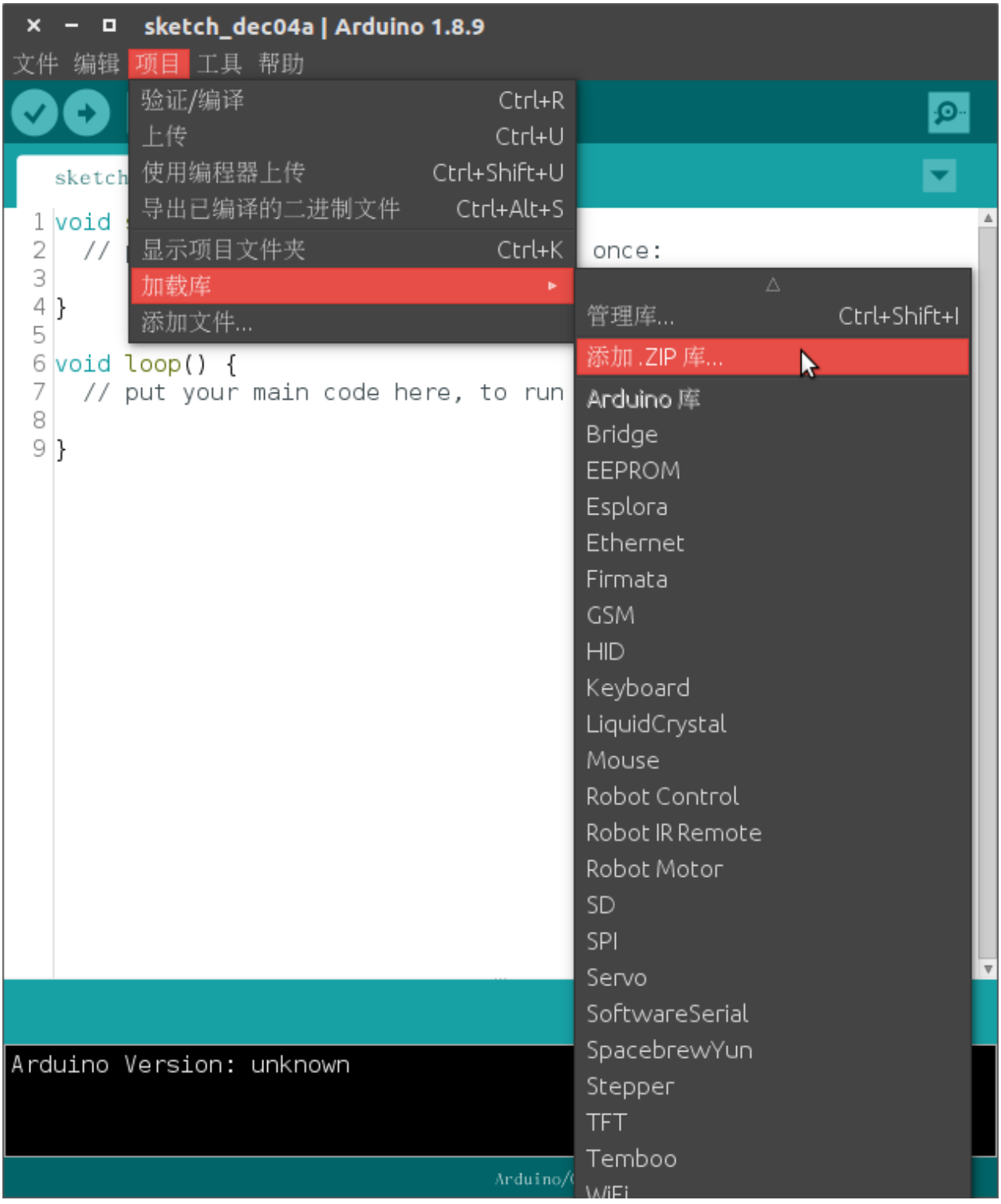
本文介绍 MU Vision Sensor 3 配合 Arduino 开发板使用 Arduino IDE 进行开发的教程。

3.1 Arduino 库导入

1. 在 Arduino 官网下载安装最新的 Arduino IDE：https://www.arduino.cc/en/Main/Software

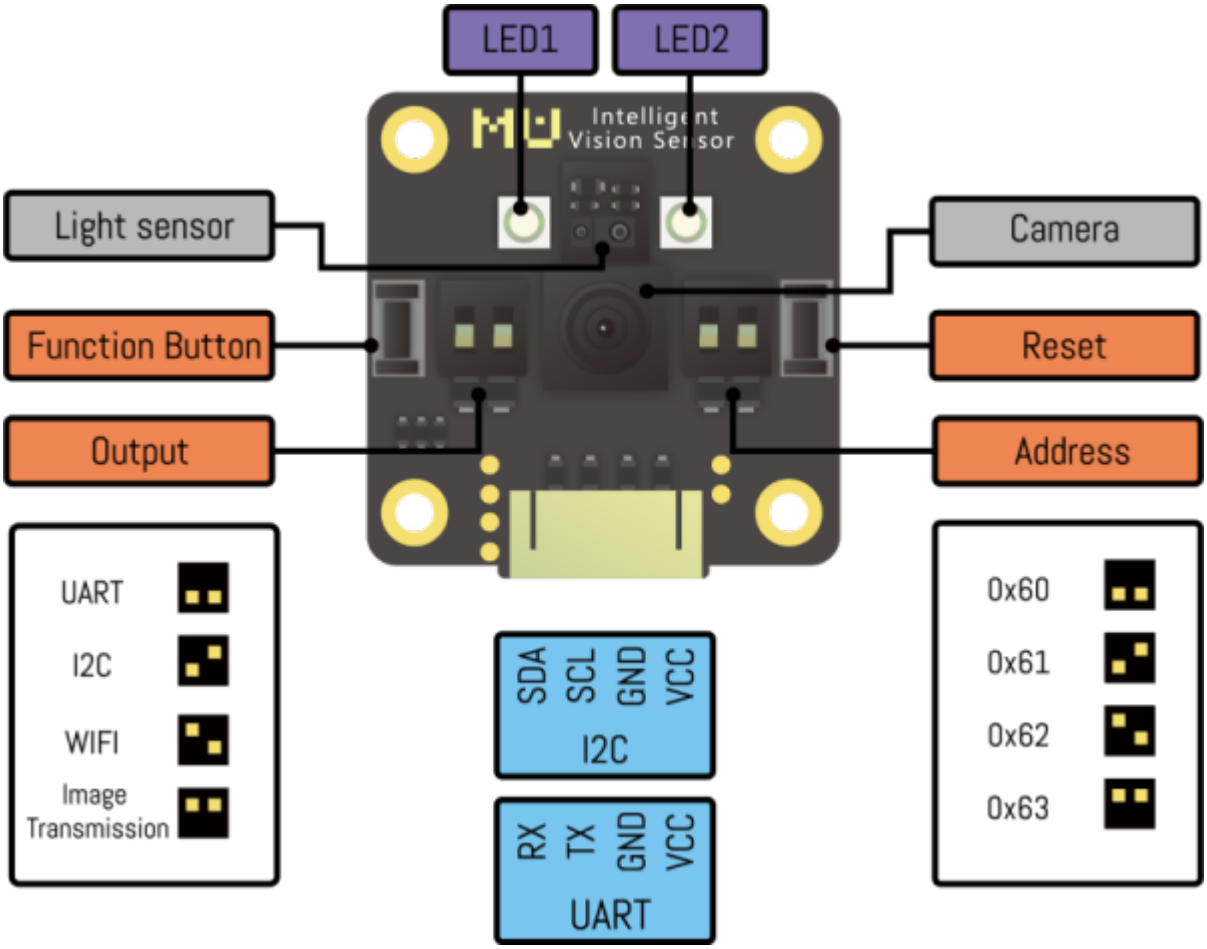
2. 在 github 下载最新的 MUVisionSensor3 的 Arduino 库（Source code(zip) 文件）：https://github.com/mu-opensource/MuVisionSensor3/releases/latest

3. 打开 Arduino IDE，点击项目–> 加载库–> 添加.ZIP 库，选择步骤 2 中下载 ZIP 文件，点击确定按钮完成库的添加


sketch_dec04a | Arduino 1.8.9
文件 编辑 项目 工具 帮助
验证/编译 Ctrl+R
上传 Ctrl+U
使用编程器上传 Ctrl+Shift+U
导出已编译的二进制文件 Ctrl+Alt+S
显示项目文件夹 Ctrl+K
加载库
添加文件...
管理库... Ctrl+Shift+I
添加 .ZIP 库...
Arduino 库
Bridge
EEPROM
Esplora
Ethernet
Firmata
GSM
HID
Keyboard
LiquidCrystal
Mouse
Robot Control
Robot IR Remote
Robot Motor
SD
SPI
Servo
SoftwareSerial
SpacebrewYun
Stepper
TFT
Temboo
WiFi
void loop() {
// put your main code here, to run
}
Arduino Version: unknown

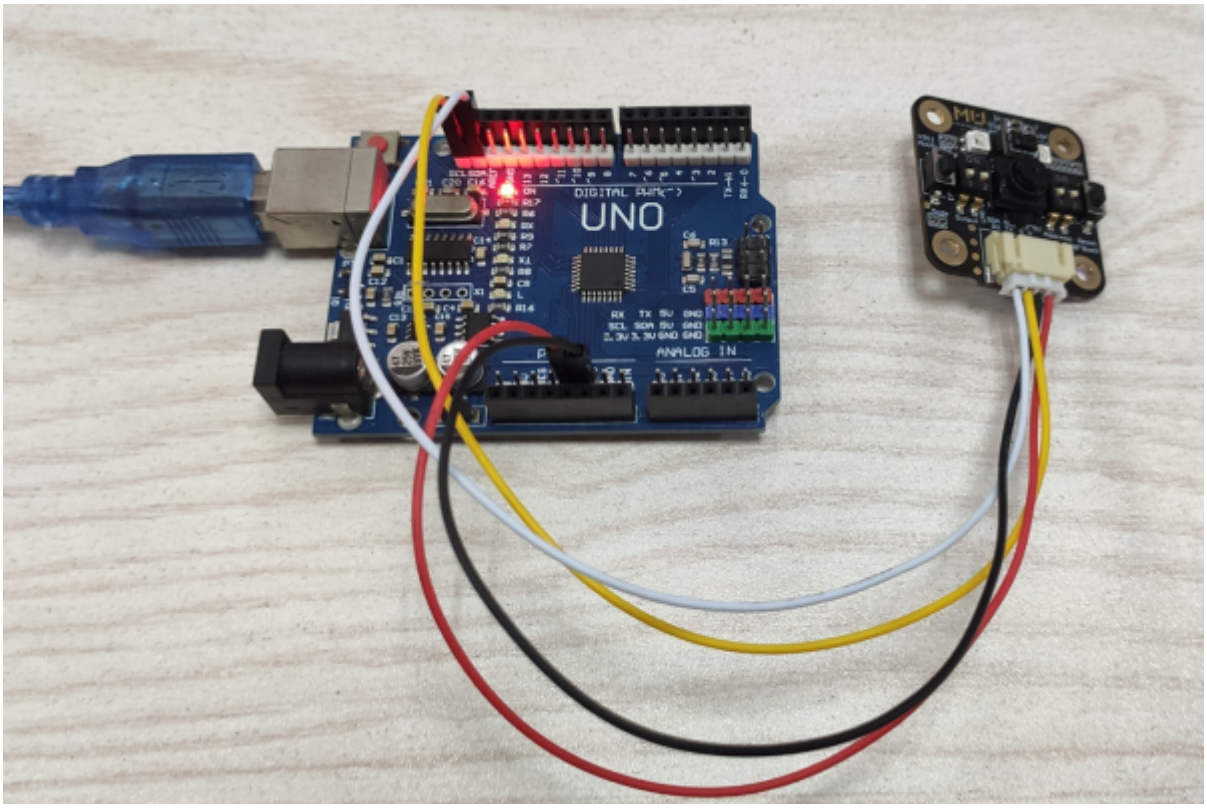
3.2 Arduino 硬件连接

MU Vision Sensor 3 的外设和接口如图所示：



3.2.1 I2C 模式

1. 将模块左侧输出模式拨码开关 1 拨至下方，2 拨至上方，切换至 I2C 模式；
2. （不推荐修改此设置）将模块右侧的地址选择拨码开关拨至对应位（默认地址 0x60，1、2 都在下方）；
3. 将 MU 输出接口 `SDA` 口接至 Arduino 对应的 `SDA` 口，`SCL` 口接至 Arduino 对应的 `SCL` 口。



3.2.2 串口模式

1. 将模块左侧输出模式拨码开关 1、2 都拨至下方，切换至串口模式；
2. （不推荐修改此设置）将模块右侧的地址选择拨码开关拨至对应位（默认地址 0x60，1、2 都在下方）；
3. 将 MU 输出接口 `RX` 口接至 Arduino 对应的 `TX` 口，`TX` 口接至 Arduino 对应的 `RX` 口。

3.2.3 AT 指令模式（适用于 V1.1.5 及以上版本的固件）

1. 将模块左侧输出模式拨码开关 1 拨至上方，2 拨至下方，切换至 AT 指令模式；
2. 将 MU 输出接口 `RX` 口接至 Arduino 对应的 `TX` 口，`TX` 口接至 Arduino 对应的 `RX` 口。

3.2.4 图传模式（适用于 V1.1.5 及以上版本的固件）

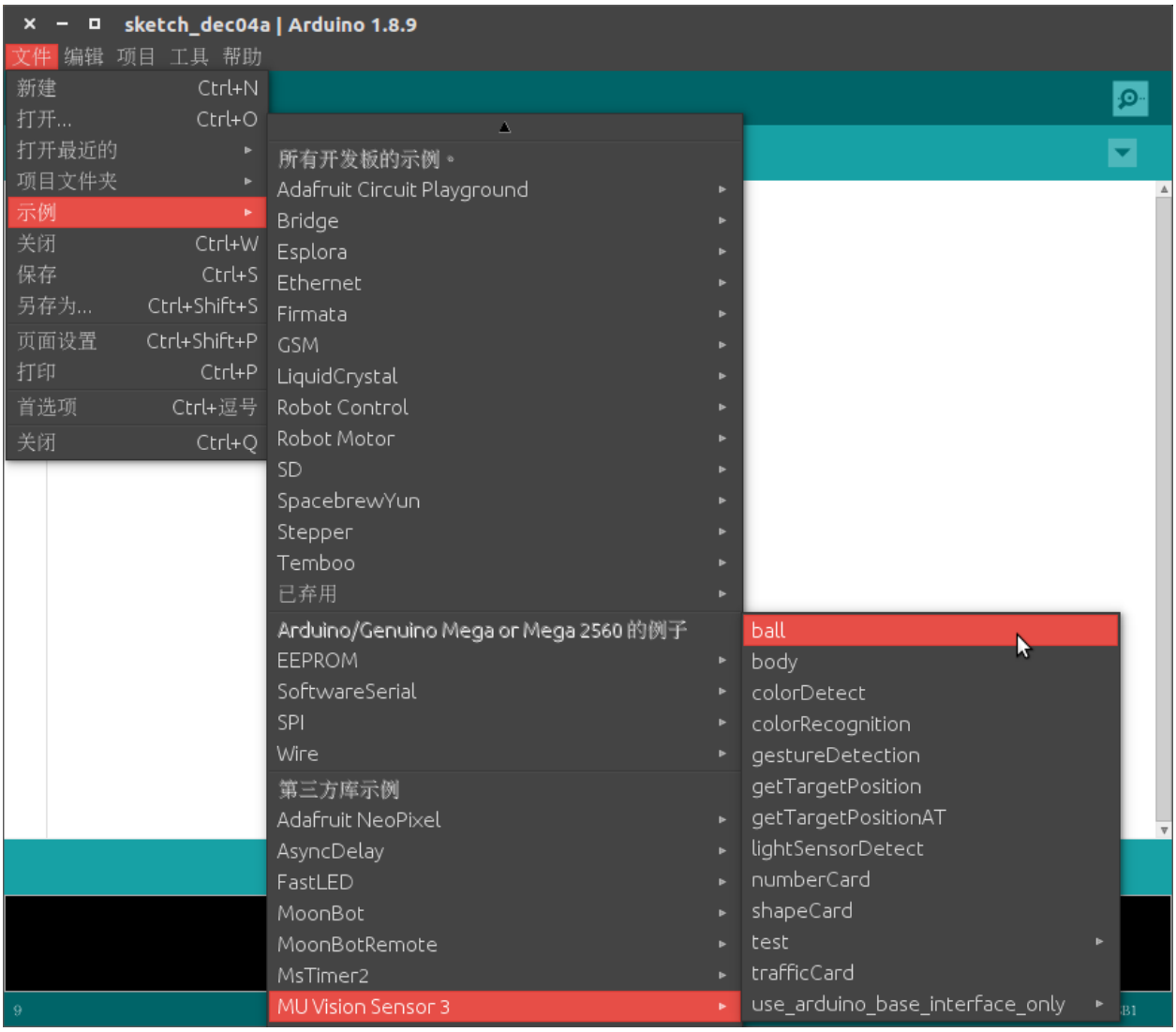
1. 将模块左侧输出模式拨码开关 1、2 都拨至上方，切换至图传模式；
2. 将 MU 输出接口 `RX` 口接至 Arduino 对应的 `TX` 口，`TX` 口接至 Arduino 对应的 `RX` 口。

3.3 示例说明

打开 Arduino IDE，选择顶部的工具-> 开发板，常用开发板为 Arduino Uno。

如果使用 MoonBot 主控，则选开发板为 `Arduino Mega 2560`，并选择处理器为 `ATmega 1280`。连接开发板后选择相应端口则完成 Arduino 开发板的连接。

如果成功导入了开发板兼容的库，选择顶部的文件-> 示例-> 第三方库示例->`MU Vision Sensor 3` 即可打开官方示例程序。



`ball`：球检测示例

`body`：人体检测示例

`colorDetect`：颜色检测示例

`colorRecognition`：颜色识别示例

`gestureDetection`：手势识别示例

getTargetPosition：获取目标位置示例

getTargetPositionAT：使用 AT 指令获取目标位置示例

lightSensorDetect：光线传感器通用功能示例（接近检测，颜色检测，光线检测）

numberCard：数字卡片示例

shapeCard：形状卡片示例

trafficCard：交通卡片示例

arduino_base_interface：串口直接发送字符示例

CHAPTER 4

MU 3 MakeCode 教程

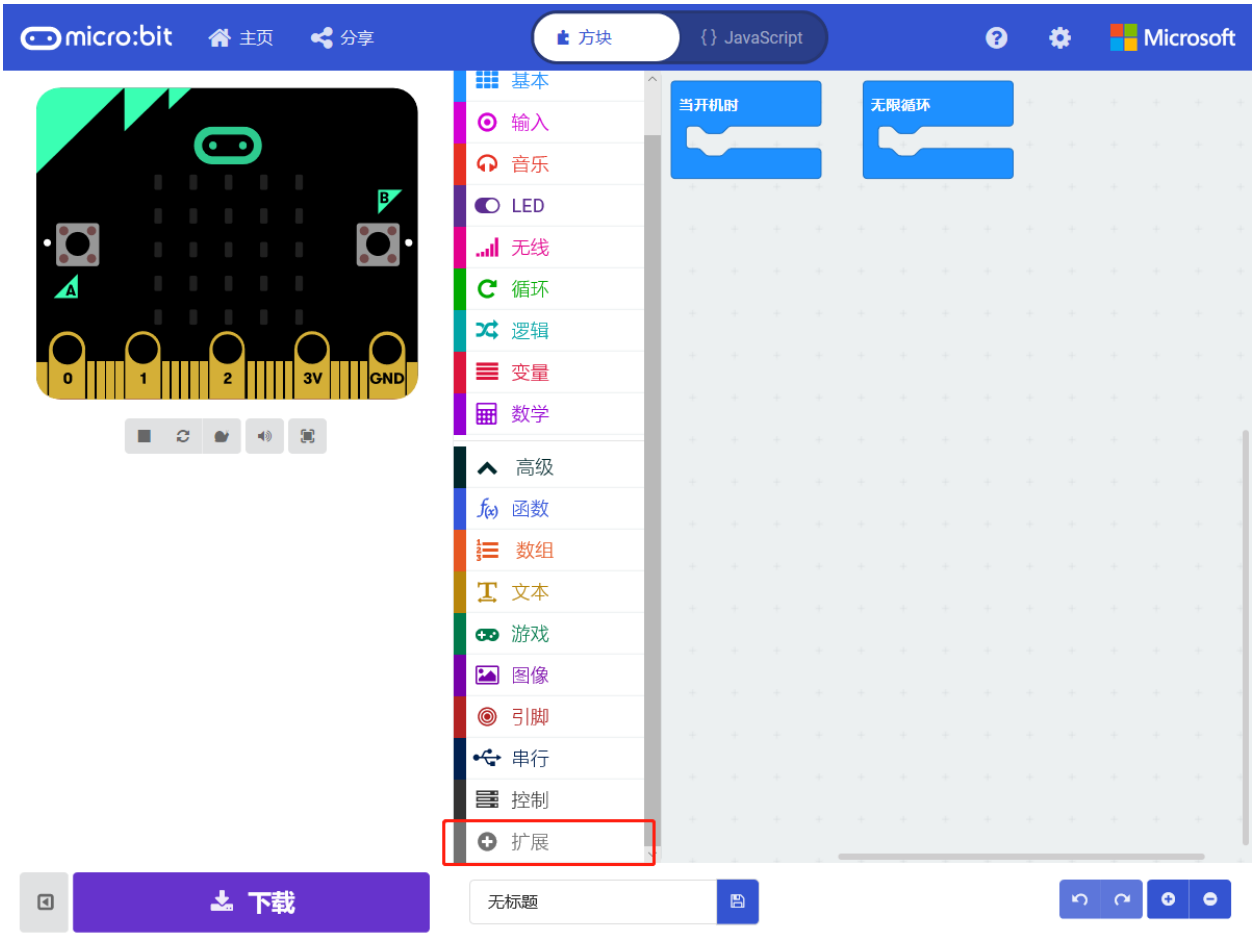
本文介绍 MU Vision Sensor 3 和 Micro:bit 开发板在 MakeCode 环境中开发的教程。

MakeCode 是一款由微软开发的可视化编程软件，适配 Micro:bit、乐高 EV3 等硬件平台。

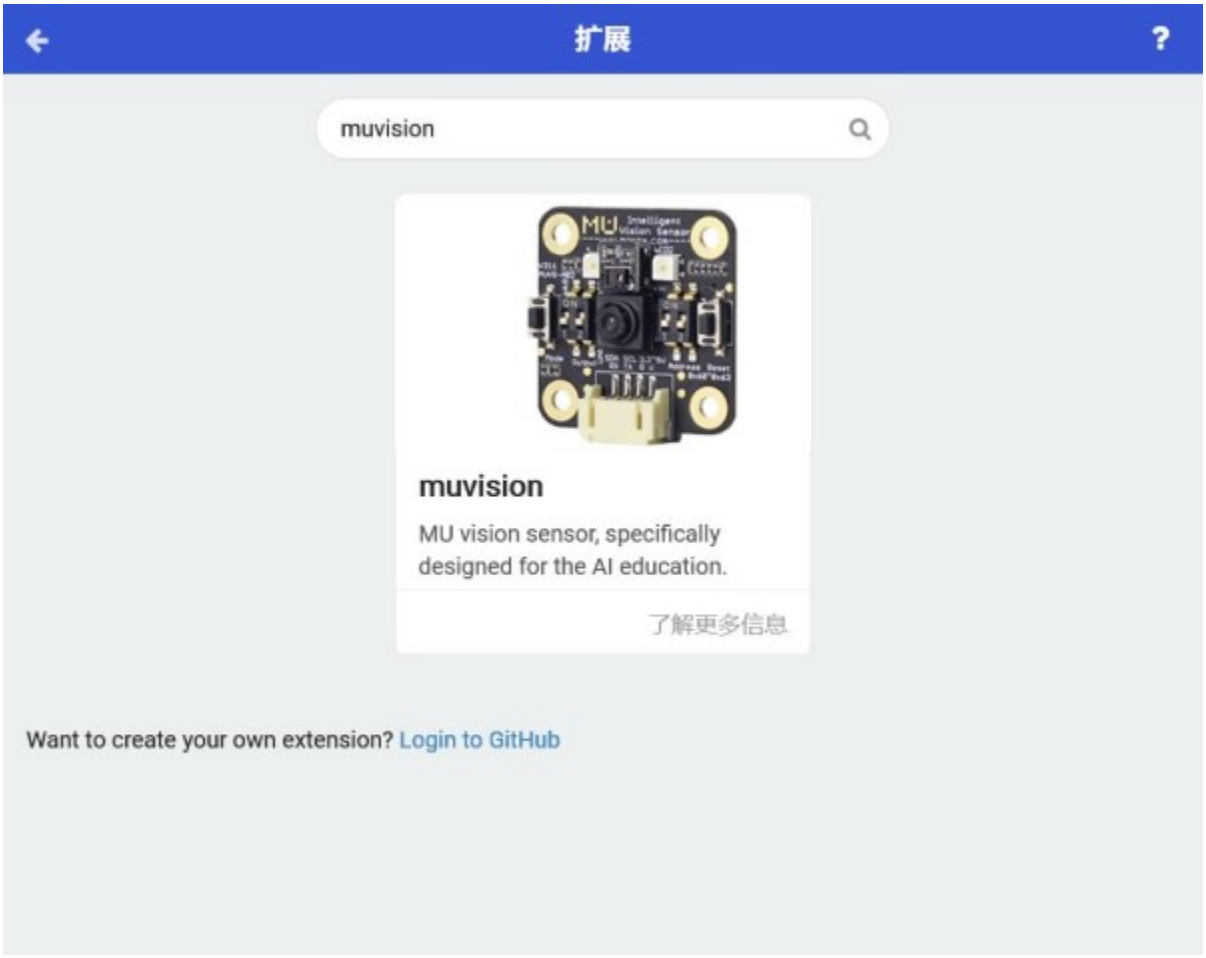
MakeCode 在线编程：MakeCode for micro:bit

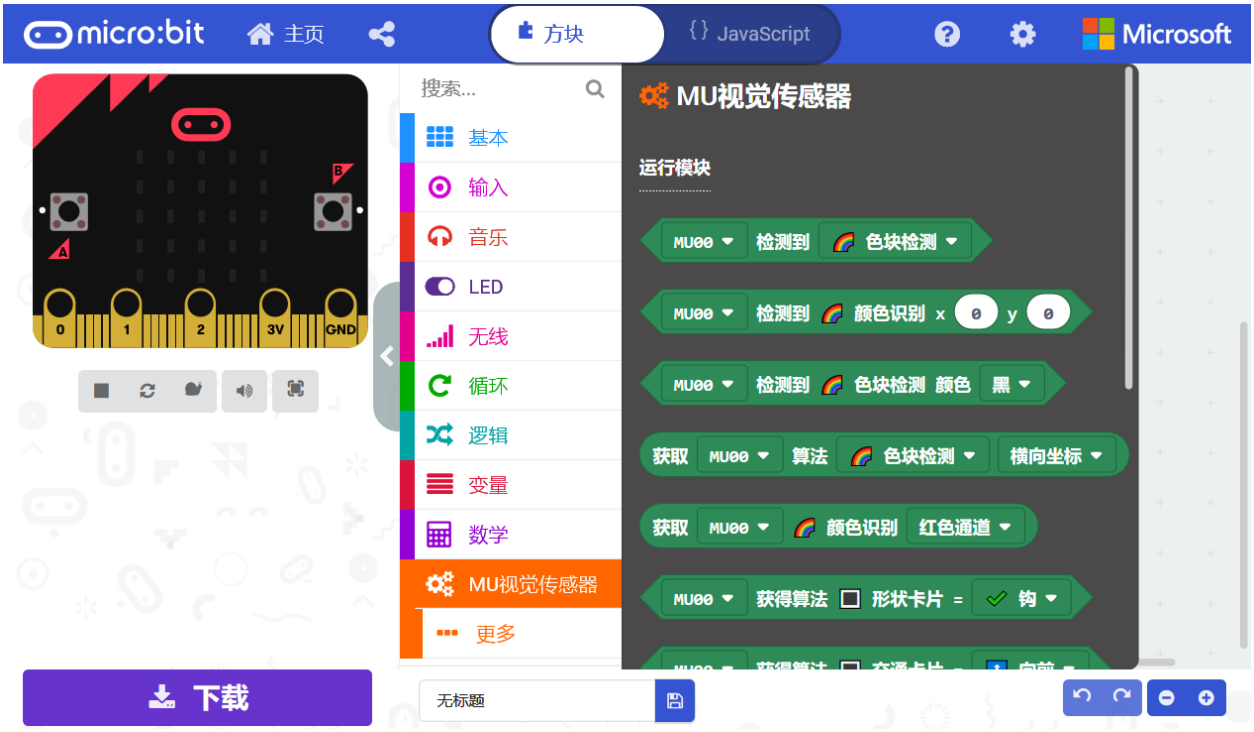
4.1 传感器扩展库导入

打开 MakeCode 并新建一个项目，在模块工具箱中点击高级-> 扩展；



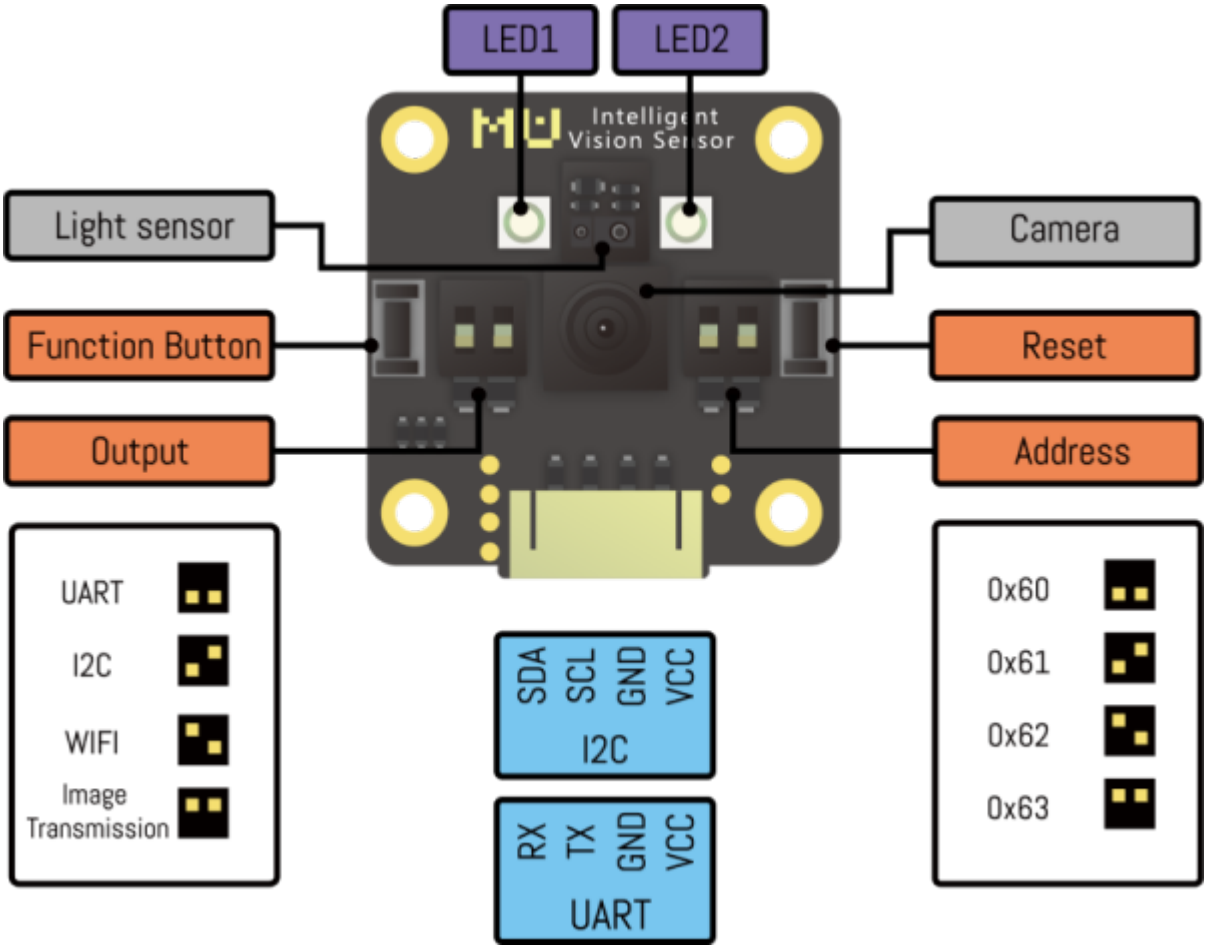
搜索 `muvision` 或 `mu` (旧版链接 `mu-opensource/pxt-MuVisionSensor3` 已弃用，请尽快转至新版库)，点击卡片完成添加。


扩展
muvision
muvision
MU vision sensor, specifically designed for the AI education.
了解更多信息
Want to create your own extension? Login to GitHub


micro:bit
主页
方块
{} JavaScript
Microsoft
搜索...
基本
输入
音乐
LED
无线
循环
逻辑
变量
数学
MU视觉传感器
更多
MU视觉传感器
运行模块
MU00 检测到 色块检测
MU00 检测到 颜色识别 x 0 y 0
MU00 检测到 色块检测 颜色 黑
获取 MU00 算法 色块检测 横向坐标
获取 MU00 颜色识别 红色通道
MU00 获得算法 形状卡片 = 钩
下载
无标题

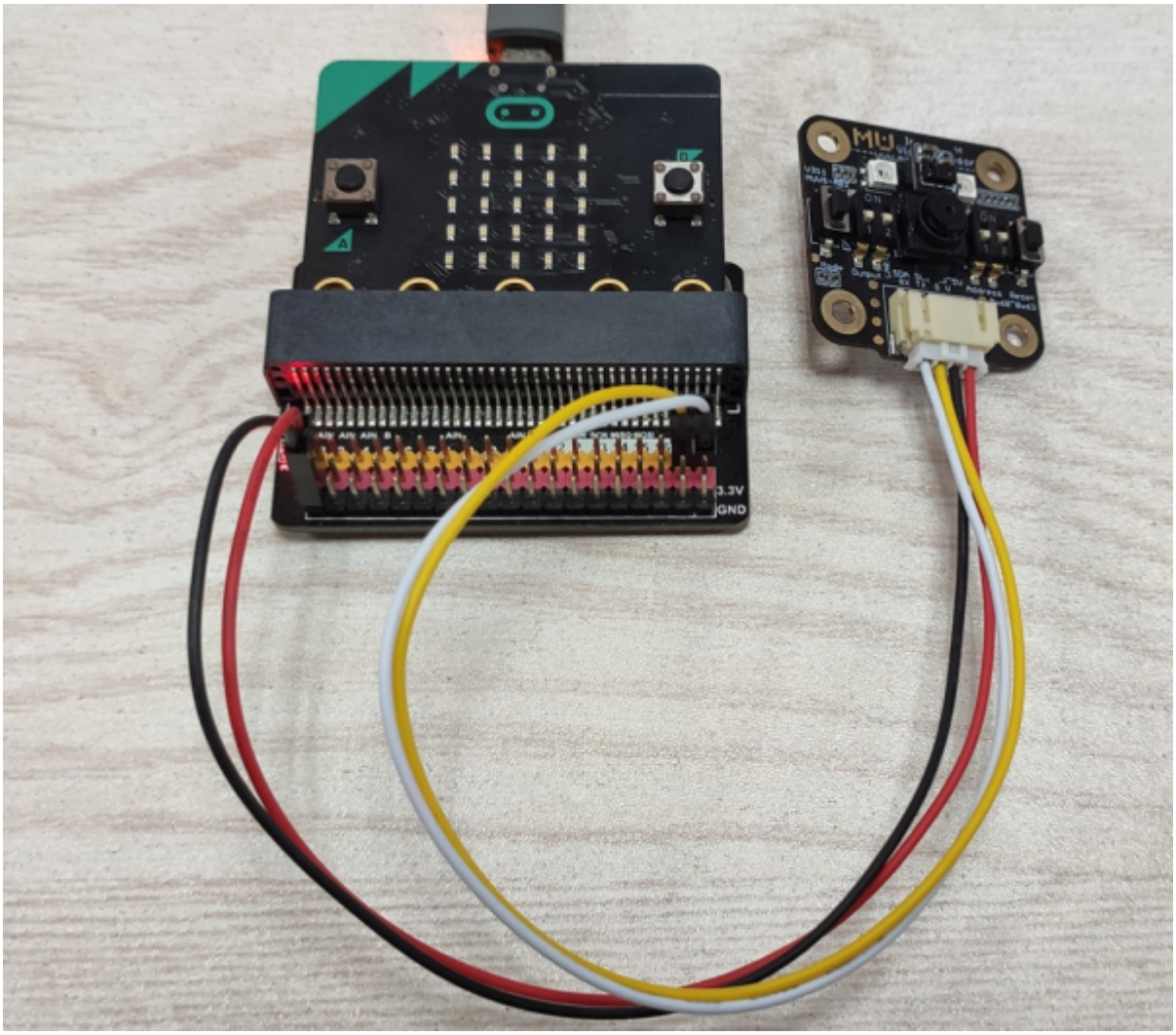
4.2 Micro:bit 硬件连接

MU Vision Sensor 3 的外设和接口如图所示：



4.2.1 I2C 模式

1. 将传感器左侧输出模式拨码开关 1 拨至下方，2 拨至上方；
2. 将传感器输出接口 SDA 引脚（P1）和 SCL 引脚（P2）接至 Micro:bit 对应的 SDA 引脚（P20）与 SCL 引脚（P19），同时将 P3 接地，P4 接电源（3.3-5V）；
3. 将传感器的地址选择拨码开关拨至对应位（默认地址 `0x60` 则 1、2 都在下方，不推荐修改此设置）。



4.2.2 串口模式

1. 将传感器左侧输出模式拨码开关 1、2 均拨至下方；
2. 将传感器输出接口 RX 引脚（P1）接至 Micro:bit 对应的 TX 引脚，TX 引脚（P2）口接至 Micro:bit 对应的 RX 引脚，同时将 P3 接地，P4 接电源（3.3-5V）；
3. 将传感器的地址选择拨码开关拨至对应位（默认地址 0x60 则 1、2 都在下方，不推荐修改此设置）。

当前版本中串口模式下 Micro:bit 将无法通过 USB 串口打印调试信息，串口波特率固定为 9600。

4.2.3 AT 指令模式（适用于 V1.1.5 及以上版本的固件）

1. 将模块左侧输出模式拨码开关 1 拨至上方，2 拨至下方，切换至 AT 指令模式；
2. 将 MU 输出接口 `RX` 口接至 Micro:bit 对应的 `TX` 口，`TX` 口接至 Micro:bit 对应的 `RX` 口。

4.2.4 图传模式（适用于 V1.1.5 及以上版本的固件）

1. 将模块左侧输出模式拨码开关 1、2 都拨至上方，切换至图传模式；
2. 将 MU 输出接口 `RX` 口接至 Micro:bit 对应的 `TX` 口，`TX` 口接至 Micro:bit 对应的 `RX` 口。

4.3 模块使用介绍

4.3.1 MU 视觉传感器

初始化模块

1. Serial 模式：根据 Micro:bit 与小 MU 视觉传感器的硬件连接自定义串口重定向模块中的 TX、RX 引脚号，示例中采用了 Micro:bit 的 P12 与 P13。

1. I2C 模式：Micro:bit 与小 MU 视觉传感器的连接初始化为 I2C 模式。

开启算法

当前版本共有如下 7 种算法，每种算法的具体分类和返回结果详见小 MU 视觉传感器技术规格书文档。

设置算法性能

不同算法性能下识别的速度与准确率会有所差异，可根据实际的应用需要选择合适的性能参数。

速度优先：简单环境下使用，识别速度快，误报率稍高；

性能均衡：默认模式；

准确率优先：复杂场景情况下使用，识别速度稍慢，误报率低；当多类识别算法同时开启时，譬如形状卡片与交通卡片混合摆放识别时，请采用该模式，以消除不同组卡片间的误报。

关闭/开启摄像头高帧率模式

默认使用高帧率模式，相对普通模式有更快的识别速度，但功耗和发热量随之增加，可用于快速检测的场景，如需要低功耗使用则可关闭。

设置摄像头白平衡

当摄像头视野中出现大面积有颜色的物体时，摄像头会发生白平衡失真，产生偏色现象，通过事先锁定白平衡能够防止此问题的发生。在调用此编程模块时，需要将摄像头朝向白纸距离约 20 厘米进行测光，数秒后摄像头的白平衡会自动被锁定。

设置摄像头数码变焦

数码变焦等级越大可检测的距离越远，但视野范围会变窄。

数码变焦等级 1(距离近, 视野广)~ 数码变焦等级 5(距离远, 视野窄)。

针对不同距离的物体通过试验测试合理设置数码变焦等级值可以取得较好的识别效果。

板载 LED 灯光设置

小 MU 视觉传感器正面板载的两颗 LED 灯每闪烁一次表示执行一帧图像识别。

可通过设置识别到目标与未识别到目标时灯光的颜色来获得反馈。

默认设置：未检测到闪红灯，检测到则闪蓝灯。当进行颜色识别时，默认 LED 关闭。

恢复模块默认设置

关闭所有算法，重置所有硬件设置

光线传感器开启功能

开启光线传感器对应功能，手势检测功能无法与其他功能共用。

光线传感器设置灵敏度

光线传感器读取接近检测数值

光线传感器读取环境光检测数值

光线传感器读取手势检测状态

光线传感器读取手势检测结果

4.3.2 MU 视觉传感器 WiFi

WiFi 配置模块只能在 `WiFi` 和 图传模式下使用。

读取本地 IP

读取 MU 的 IP。

读取目标 IP

读取目标 IP。

WiFi 配置

配置 WiFi 账号密码及模式。

WiFi 连接

尝试开启/关闭 WiFi 连接，若成功，则返回 `true`。

WiFi 配置目标 IP

配置目标 IP，需 WiFi 连接成功后调用才生效。

WiFi 读取透传数据

读取目标设备向 MU 发送的数据。

4.4 完整示例

4.4.1 球体与人体检测

初始程序：采用 I2C 连接，启用球体检测算法，其余设置为默认。

循环程序：如果视觉传感器检测到球，会通过 I2C 向 Micro:bit 发送检测到的数据，Micro:bit 会通过串口向电脑发送检测到的信息，否则循环显示未检测到球，人体检测同理。

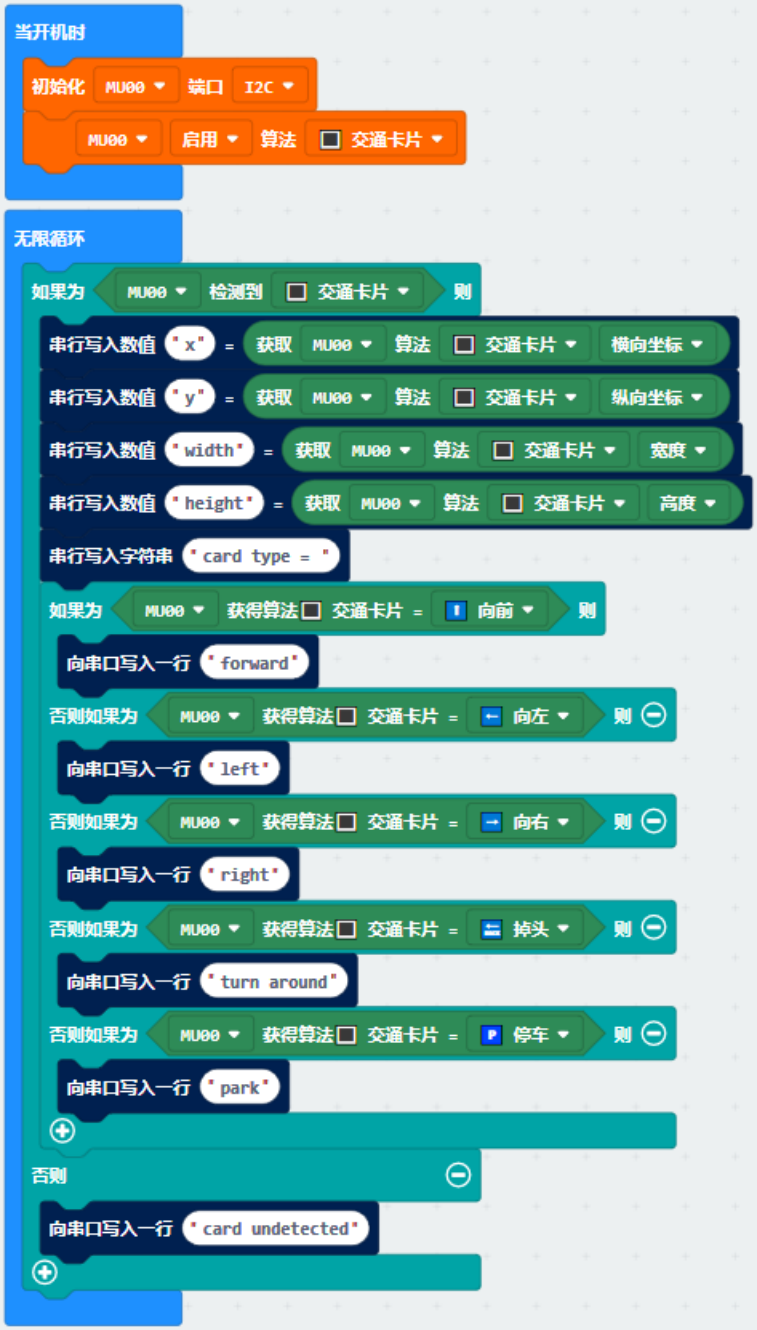
实验现象：未检测到球则视觉传感器闪红灯，控制台显示”ball undetected”。检测到球则视觉传感器闪蓝灯，控制台显示返回的坐标等信息。

4.4.2 卡片识别

初始程序：采用 I2C 连接，启用交通卡片识别算法，其余设置为默认。

循环程序：如果视觉传感器检测到交通卡片，会通过 I2C 向 Micro:bit 发送检测到的数据，Micro:bit 会通过串口向电脑发送识别到的位置和类型信息，否则循环显示未识别，其他类型的卡片识别同理。

实验现象：未检测到卡片则视觉传感器 LED 闪红灯，控制台显示”card undetected”。识别到则视觉传感器 LED 闪蓝灯，控制台显示返回的坐标信息和交通卡片的具体图案信息。



4.4.3 颜色识别

初始程序：采用 I2C 连接，启用颜色识别算法，锁定摄像头白平衡防止偏色，其余设置为默认。

循环程序：视觉传感器识别坐标 (50,50) 处的颜色，通过 I2C 向 Micro:bit 发送检测到的数据，Micro:bit 会通过串口向电脑发送识别到的通道信息和颜色类别。

实验现象：视觉传感器始终不闪灯，控制台显示返回的通道值和颜色的类别。


当开机时
初始化 MU00 端口 I2C
MU00 启用 算法 颜色识别
MU00 摄像头白平衡 自动
无限循环
如果为 MU00 检测到 颜色识别 x 50 y 50 则
串行写入数值 "R" = 获取 MU00 颜色识别 红色通道
串行写入数值 "G" = 获取 MU00 颜色识别 绿色通道
串行写入数值 "B" = 获取 MU00 颜色识别 蓝色通道
串行写入字符串 "color type = "
如果为 MU00 获得算法 颜色识别 颜色 = 红 则
向串口写入一行 "red"
否则如果为 MU00 获得算法 颜色识别 颜色 = 绿 则
向串口写入一行 "green"
否则如果为 MU00 获得算法 颜色识别 颜色 = 蓝 则
向串口写入一行 "blue"
否则
向串口写入一行 "others"

4.4.4 色块检测

初始程序：采用 I2C 连接，启用色块检测算法，锁定摄像头白平衡防止偏色，其余设置为默认。

循环程序：如果视觉传感器检测到红色块，会通过 I2C 向 Micro:bit 发送检测到的数据，Micro:bit 会通过串口向电脑发送识别到的位置和类型信息，否则循环显示未检测到。

实验现象：未识别到时视觉传感器闪红灯，控制台显示”color block undetected”。识别到红色块则视觉传感器闪蓝灯，控制台显示获得的红色块坐标大小信息。

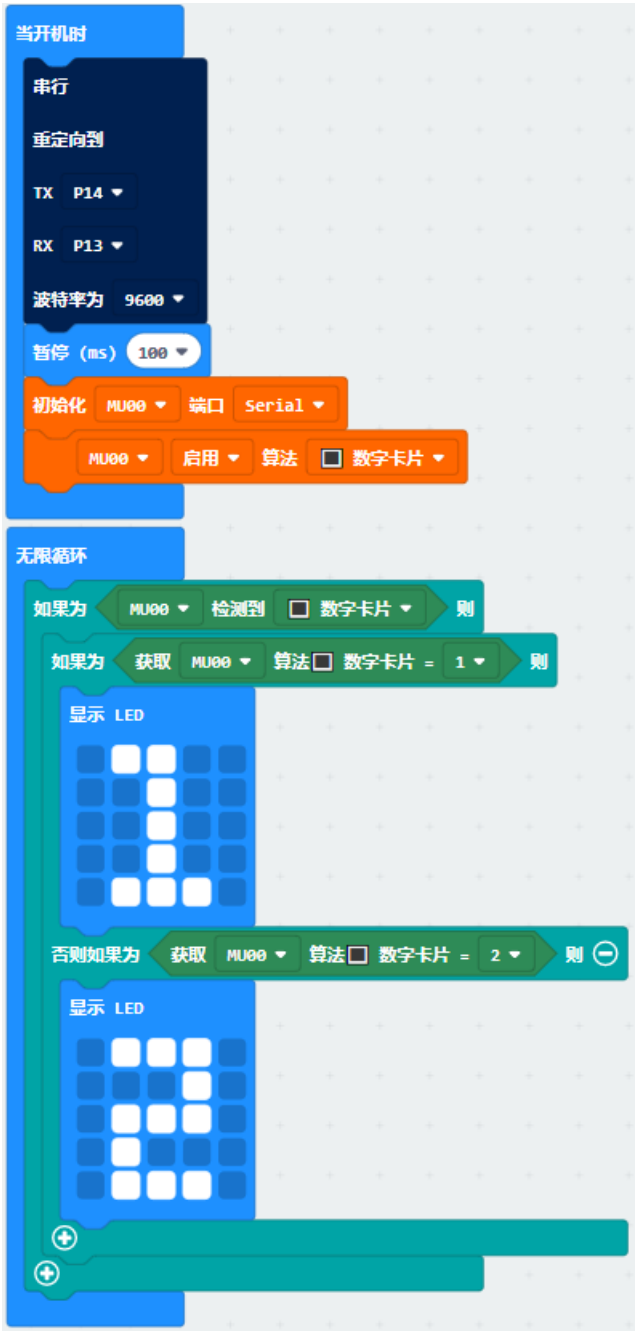
4.4.5 串口模式示例

拨动左侧输出模式拨码开关至串口模式，MU 可采用串口连接 Micro:bit。由于此模式下电脑无法与 Micro:bit 串口通讯，所以使用 Micro:bit 自带的点阵屏直接显示检测结果。

初始程序：重新定义串口引脚到 P14 和 P13，采用串口连接，启用数字卡片识别算法，其余设置为默认。

循环程序：如果视觉传感器检测到数字卡片，会通过串口向 Micro:bit 发送数据，使用 Micro:bit 自带的 LED 显示识别到的数字，以 1 和 2 为示例，其他卡片类型同理。

实验现象：未检测到数字卡片时视觉传感器闪红灯，检测到时闪蓝灯，当识别到卡片为 1 时 Micro:bit 显示数字 1，识别到卡片 2 时显示数字 2。



更多例程可参考：https://makecode.microbit.org/pkg/mu-opensource/pxt-muvision

CHAPTER 5

MU 3 MicroPython 教程

本文介绍 MU Vision Sensor 3 与 Micro:bit 通过 MicroPython 进行编程开发的教程。

5.1 配置 Mu Editor 和 Micro:bit

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 Micro:bit 固件：

GitHub：https://github.com/mu-opensource/MuVisionSensor3-MicroPython

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

（2）更新 Micro:bit 固件：

将 Micro:bit 通过 USB 线连接电脑，出现 Micro:bit 的磁盘，将下载的固件 microbit-micropython-MuVisionSensor-x.x.x.hex 文件拖入磁盘中，Micro:bit 将自动更新固件并重启。

（3）下载并安装 Mu Editor：https://codewith.mu/

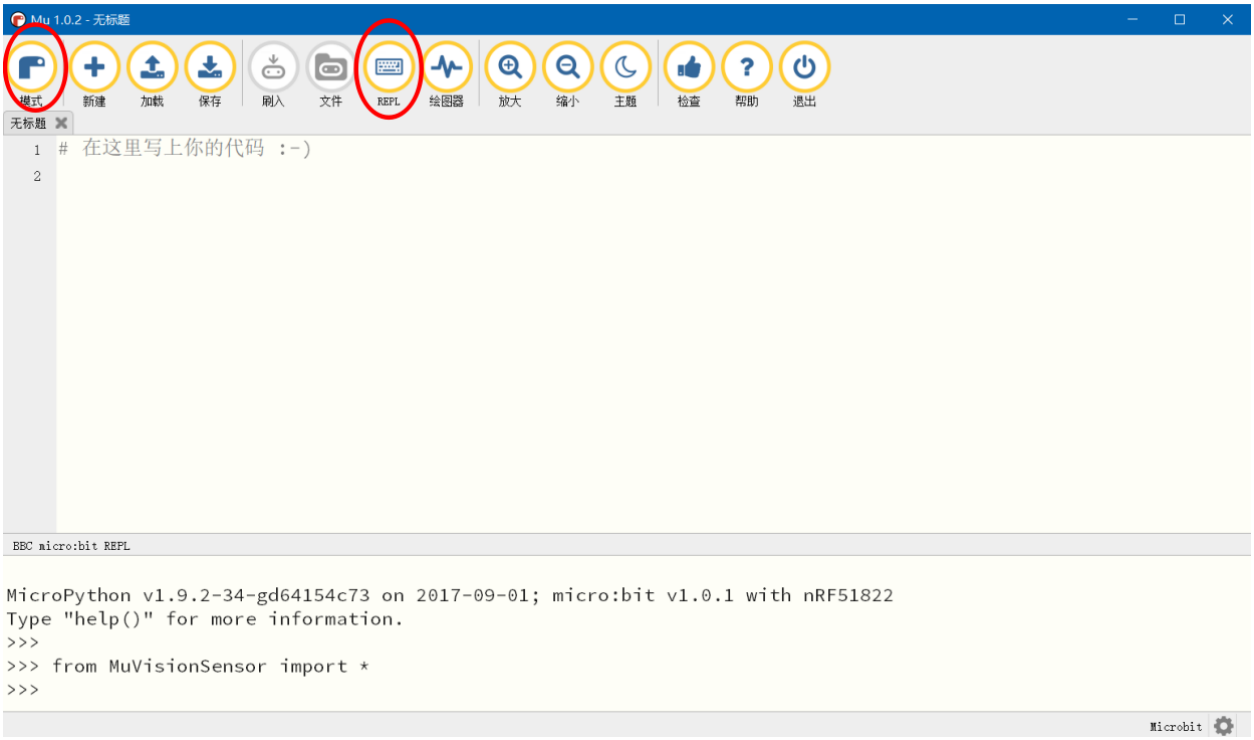
（4）导入传感器

打开 Mu Editor，在顶部选择模式为 BBC micro:bit，连接 micro:bit 后左下角显示连接到新的 `micro:bit` 设备即可进行编程。单击顶部 REPL 按钮进入串口实时模式，micro:bit 将返回固件版本信息。输入：

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

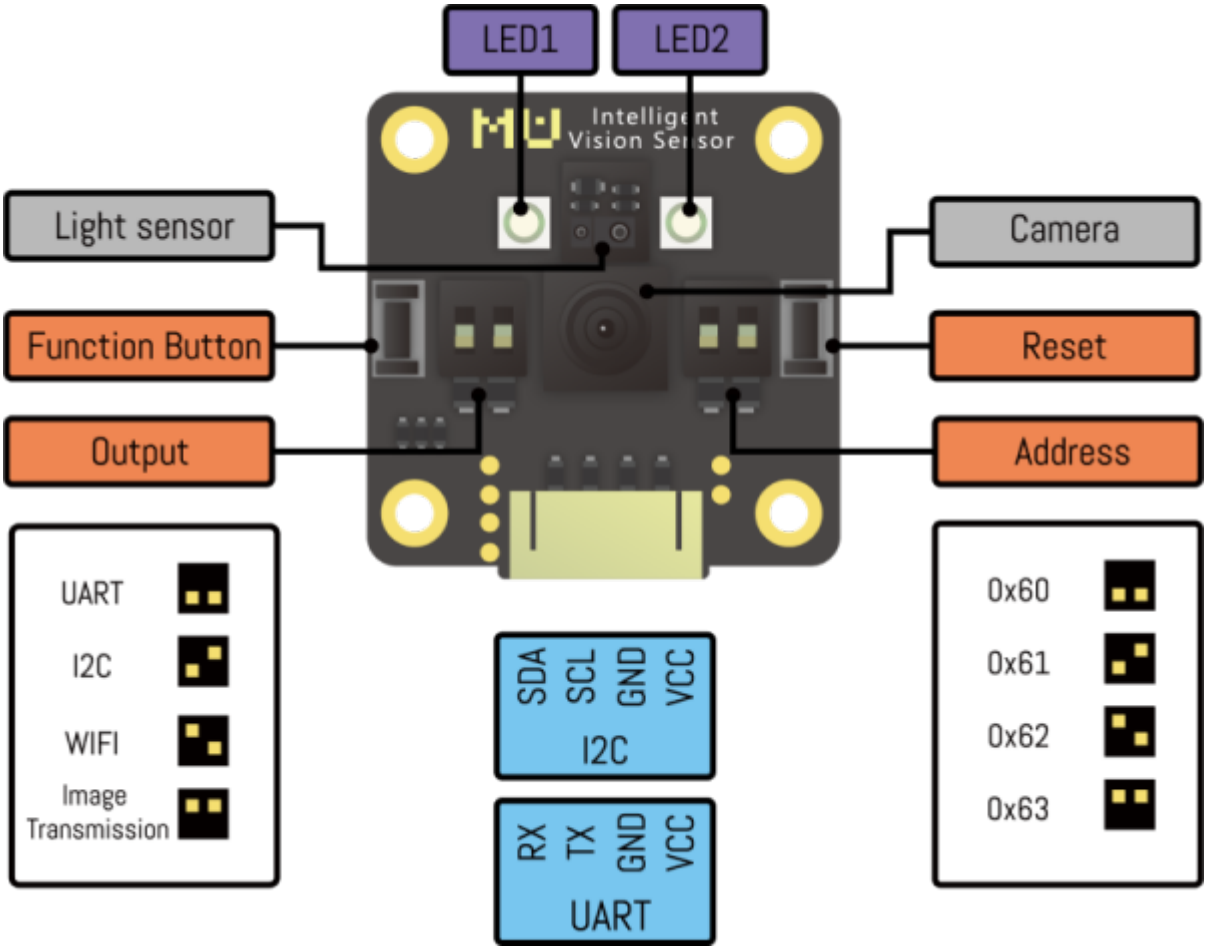
导入传感器后即可使用 MuVisonSensor 类中的所有公开 API

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



5.2 Micro:bit 硬件连接

MU Vision Sensor 3 的外设和接口如图所示：

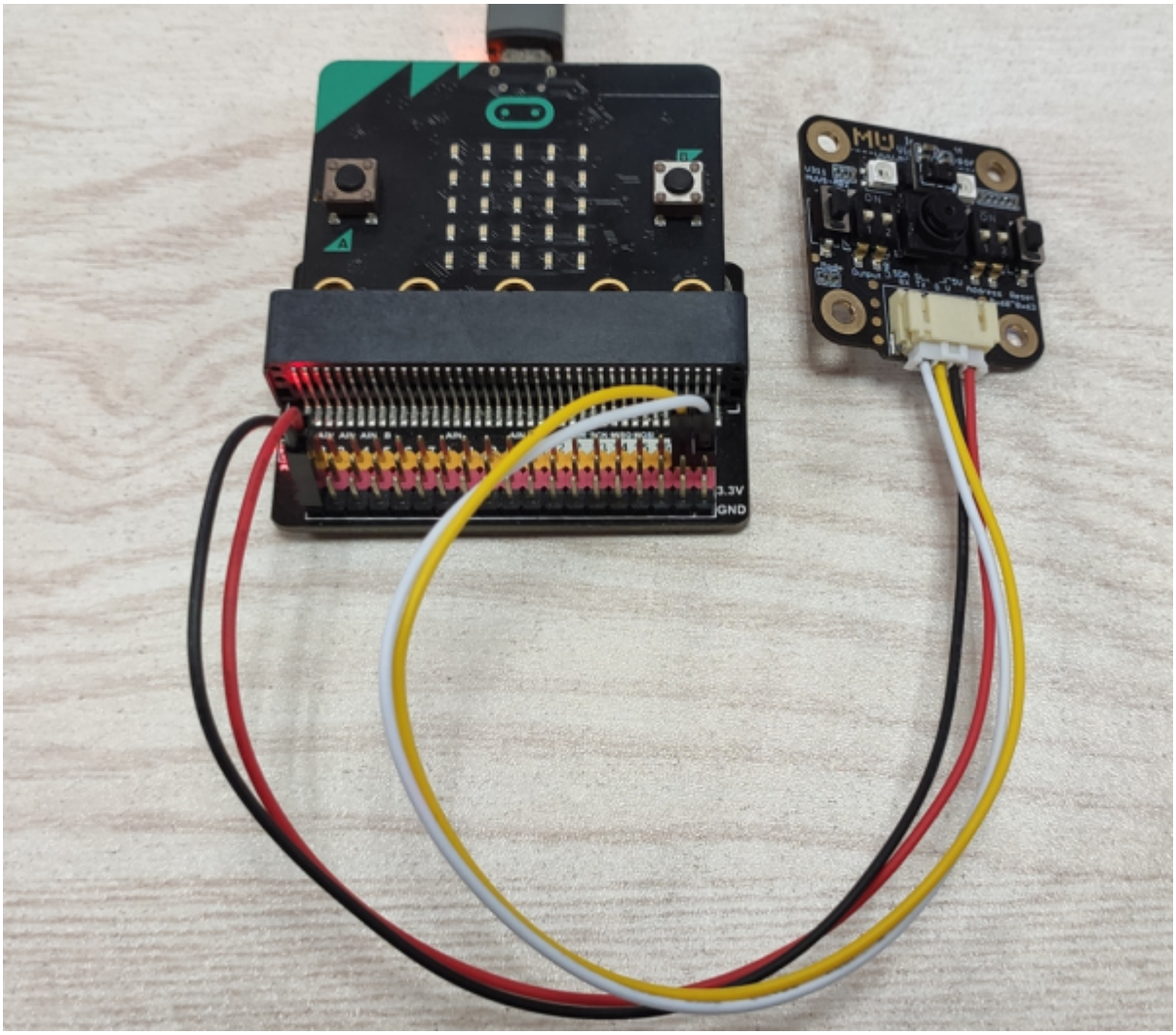


（1）将传感器左侧输出模式拨码开关 1 拨至下方，2 拨至上方；

（2）将传感器输出接口 SDA 引脚（P1）和 SCL 引脚（P2）接至 Micro:bit 对应的 SDA 引脚（P20）与 SCL 引脚（P19），同时将 P3 接地，P4 接电源（3.3-5V）；

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

*** 目前仅支持 *I2C* 模式**



5.3 API 使用说明

MuVisionSensor 库内所有的的函数及可选参数的枚举可以通过以下代码获取：

```
import MuVisionSensor                    # 导入库
help(MuVisionSensor)                     # 获取可选枚举类型
help(MuVisionSensor.MuVisionSensor)      # 获取所有函数
```

5.3.1 构造函数

API:

实例化一个对象，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，默认地址为 0x60

```
MuVisionSensor.MuVisionSensor(address=0x60)
```

5.3.2 初始化

初始化传感器

API:

```
MuVisionSensor.begin()
```

5.3.3 开启算法

API:

```
MuVisionSensor.VisionBegin(vision_type)
```

目前支持的 vision_type 有：

VISION_COLOR_DETECT 颜色检测

VISION_COLOR_RECOGNITION 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

示例：

```
from MuVisionSensor import *                    # 导入库
mu=MuVisionSensor(0x60)                         # 实例化 MU 变量
mu.begin()                                      # 初始化 MU

mu.VisionBegin(VISION_COLOR_DETECT)             # 开启颜色检测算法
mu.VisionBegin(VISION_SHAPE_CARD_DETECT | VISION_BALL_DETECT) # 同时开启形状卡片检测和球体
检测算法
```

(下页继续)

(续上页)

5.3.4 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

示例:

```
mu.VisionSetLevel(VISION_BALL_DETECT, LevelSpeed)
```

获取算法性能

API:

```
mu.VisionGetLevel(vision_type)
```

返回值 0~3 代表四种算法性能

5.3.5 设置摄像头帧率模式

高帧率模式下识别速度增加，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

获取摄像头帧率模式

API:

```
MuVisionSensor.CameraGetFPS()
```

返回值为 0(FPSNormal) 或 1(FPSHigh)

5.3.6 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

5.3.7 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

5.3.8 板载 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` 板载 LED1

`Led2` 板载 LED2

`LedAll` 板载所有 LED

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

5.3.9 恢复模块默认设置

关闭所有算法，重置所有硬件设置

API:

```
MuVisionSensor.SensorSetDefault()
```

5.3.10 重启传感器

API:

```
MuVisionSensor.SensorSetRestart()
```

5.3.11 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）

5.3.12 光线传感器开启功能

开启光线传感器一项或几项功能

API:

```
MuVisionSensor.LsBegin(ls_type)
```

ls_type 的可选值为:

LS_PROXIMITY_ENABLE 接近检测

LS_AMBIENT_LIGHT_ENABLE 环境光检测

LS_COLOR_ENABLE 颜色检测

LS_GESTURE_ENABLE 手势检测

5.3.13 光线传感器关闭功能

关闭光线传感器一项或几项功能

API:

```
MuVisionSensor.LsBegin(ls_type)
```

ls_type 的可选值同上

5.3.14 光线传感器设置灵敏度

设置光线传感器灵敏度，该项设置对手势检测无效

API:

```
MuVisionSensor.LsSetSensitivity(sensitivity)
```

sensitivity 的可选值为:

SensitivityDefault 默认灵敏度

Sensitivity1 灵敏度 1

Sensitivity2 灵敏度 2

Sensitivity3 灵敏度 3

5.3.15 光线传感器白平衡校准

校准光线传感器白平衡，该设置仅对光线传感器颜色检测有效

API:

```
MuVisionSensor.LsWhiteBalanceEnable()
```

5.3.16 光线传感器读取接近检测值

API:

```
MuVisionSensor.LsReadProximity()
```

返回接近检测值，取值范围 0~255

5.3.17 光线传感器读取环境光检测值

API:

```
MuVisionSensor.LsReadAmbientLight()
```

返回环境光检测值

5.3.18 光线传感器读取颜色检测值

光线传感器读取颜色检测经白平衡校正后的值

API:

```
MuVisionSensor.LsReadColor(color_t)
```

返回颜色检测对应的值

color_t 可选值有：

LsColorLabel 颜色标签值

LsColorRed 颜色红色通道值

LsColorGreen 颜色绿色通道值

LsColorBlue 颜色蓝色通道值

LsColorHue 颜色色调值

LsColorSaturation 颜色饱和度值

LsColorValue 颜色亮度值

5.3.19 光线传感器读取颜色检测原始值

光线传感器读取颜色检测原始值

API:

```
MuVisionSensor.LsReadRawColor(color_t)
```

返回颜色检测对应的原始值

color_t 可选值有：

LsRawColorRed 颜色红色通道原始值

LsRawColorGreen 颜色绿色通道原始值

LsRawColorBlue 颜色蓝色通道原始值

5.3.20 光线传感器读取手势检测结果

API:

```
MuVisionSensor.LsReadGesture()
```

返回手势检测对应的手势类型，可选值有：

GestureNone-0 无手势

GestureUp-1 上划手势

GestureDown-2 下划手势

GestureLeft-3 左划手势

GestureRight-4 右划手势

GesturePush-5 向前推进手势

GesturePull-6 向后拉手势

5.4 示例程序

5.4.1 获取球算法结果

```
from MuVisionSensor import *          # import MuVisionSensor library
mu = MuVisionSensor()                 # create MU
mu.begin()                            # initialized MU
mu.VisionBegin(VISION_BALL_DETECT)    # enable vision type: Ball
while True:
    if mu.GetValue(VISION_BALL_DETECT, Status):
        print("X = "+str(mu.GetValue(VISION_BALL_DETECT, XValue)))              #␣
↪print X value
        print("Y = "+str(mu.GetValue(VISION_BALL_DETECT, YValue)))              #␣
↪print Y value
        print("Width = "+str(mu.GetValue(VISION_BALL_DETECT, WidthValue)))      #␣
↪print width
        print("Height = "+str(mu.GetValue(VISION_BALL_DETECT, HeightValue)))    #␣
↪print height
    else:
        print("Nothing Detected")
```

5.4.2 获取颜色识别算法结果

```
from MuVisionSensor import *                 # import MuVisionSensor library
mu = MuVisionSensor()                        # create MU
mu.begin()                                   # initialized MU
mu.VisionBegin(VISION_COLOR_RECOGNITION)     # enable vision type: Color Recognize
mu.CameraSetAwb(LockWhiteBalance)           # camera lock white balance
while True:
    if mu.GetValue(VISION_COLOR_RECOGNITION, Status):                    # if color␣
↪deteced
        color_label = mu.GetValue(VISION_COLOR_RECOGNITION, Label)       # print color␣
↪type
        if color_label == MU_COLOR_BLACK:
            print("Black")
        if color_label == MU_COLOR_WHITE:
            print("White")
        if color_label == MU_COLOR_RED:
            print("Red")
        if color_label == MU_COLOR_YELLOW:
            print("Yellow")
        if color_label == MU_COLOR_GREEN:
```

(下页继续)

(续上页)

```
            print("Green")
        if color_label == MU_COLOR_CYAN:
            print("Cyan")
        if color_label == MU_COLOR_BLUE:
            print("Blue")
        if color_label == MU_COLOR_PURPLE:
            print("Purple")
```

5.4.3 获取光线传感器手势检测结果

```
from MuVisionSensor import *                    # import MuVisionSensor library
mu = MuVisionSensor()                           # create MU
mu.begin()                                      # initialized MU
mu.LsBegin(LS_GESTURE_ENABLE)                   # light sensor enable gesture
print("Gesture Dtetect Start:")
while True:
    gesture = mu.LsReadGesture()                # get gesture
    if gesture == GestureUp:
        print("gesture:up")
    if gesture == GestureDown:
        print("gesture:down")
    if gesture == GestureLeft:
        print("gesture:left")
    if gesture == GestureRight:
        print("gesture:right")
    if gesture == GesturePush:
        print("gesture:push")
    if gesture == GesturePull:
        print("gesture:pull")
```

5.4.4 获取光线传感器环境光、接近检测结果

```
from microbit import *
from MuVisionSensor import *                    # import MuVisionSensor library
mu = MuVisionSensor()                           # create MU
mu.begin()                                      # initialized MU
mu.LsBegin(LS_PROXIMITY_ENABLE | LS_AMBIENT_LIGHT_ENABLE) # light sensor enable␣
↪proximity/amblent light detect
while True:
    print("(proximity,%d)"%mu.LsReadProximity()) # read proximity
```

(下页继续)

(续上页)

```
    print("(als,%d)"%mu.LsReadAmbientLight())     # read ambient light
    sleep(500)
```

5.4.5 获取光线传感器颜色检测结果

```
from microbit import *
from MuVisionSensor import *                # import MuVisionSensor library
mu = MuVisionSensor()                       # create MU
mu.begin()                                  # initialized MU
mu.LsWhiteBalanceEnable()                   # enable white balance
mu.LsBegin(LS_COLOR_ENABLE)                 # light sensor enable color detect
while True:
    # read color
    label = mu.LsReadColor(LsColorLabel)
    if label == MU_COLOR_BLACK:
        print("Label:Black")
    elif label == MU_COLOR_WHITE:
        print("Label:White")
    elif label == MU_COLOR_RED:
        print("Label:Red")
    elif label == MU_COLOR_YELLOW:
        print("Label:Yellow")
    elif label == MU_COLOR_GREEN:
        print("Label:Green")
    elif label == MU_COLOR_CYAN:
        print("Label:Cyan")
    elif label == MU_COLOR_BLUE:
        print("Label:Blue")
    elif label == MU_COLOR_PURPLE:
        print("Label:Purple")
    else:
        print("Label:Unknow")
    print("R:%d, G:%d, B:%d"%(mu.LsReadColor(LsColorRed),mu.LsReadColor(LsColorGreen),
↪mu.LsReadColor(LsColorBlue)))
    print("H:%d, S:%d, V:%d"%(mu.LsReadColor(LsColorHue),mu.
↪LsReadColor(LsColorSaturation),mu.LsReadColor(LsColorValue)))
    sleep(500)
```

CHAPTER 6

MU Vision Sensor 资源

6.1 技术资料

在以下网址可获取最新的 MU 视觉传感器技术资料：

官网技术支持：http://mai.morpx.com/page.php?a=sensor-support

GitHub：https://github.com/mu-opensource/

6.2 3D 打印支架

对于购买 MU 裸板的用户，我们提供了 3D 打印的外壳和折叠支架文件，可以在小车等使用场景下固定和调节传感器的角度，有 3D 打印机的创客可自行打印。

MU3 3D 打印支架

6.3 平台链接

MU 视觉传感器可以和各类开源软硬件配合使用，查看各类开源平台以学习其基础知识。

Mixly:

Mixly 官网 http://mixly.org/

Mixly 帮助文档 https://mixly.readthedocs.io/zh_CN/latest/contents.html

Arduino：

Arduino 官网 https://www.arduino.cc/

Arduino 中文社区 https://www.arduino.cn/forum.php

DF 创客社区 http://mc.dfrobot.com.cn/

DF wiki MU 3 资料 SKU:SEN0314 小 MU 视觉传感器

极客工坊 https://www.geek-workshop.com/forum.php

Micro:bit:

Micro:bit 官网 https://microbit.org/zh-CN/

MakeCode 在线编程 https://makecode.microbit.org/#

MicroPython:

MicroPython 官网 http://micropython.org/

MicroPython 中文社区 http://www.micropython.org.cn/bbs/forum.php

Mu IDE https://codewith.mu/

CHAPTER 7

MU Vision Sensor 应用

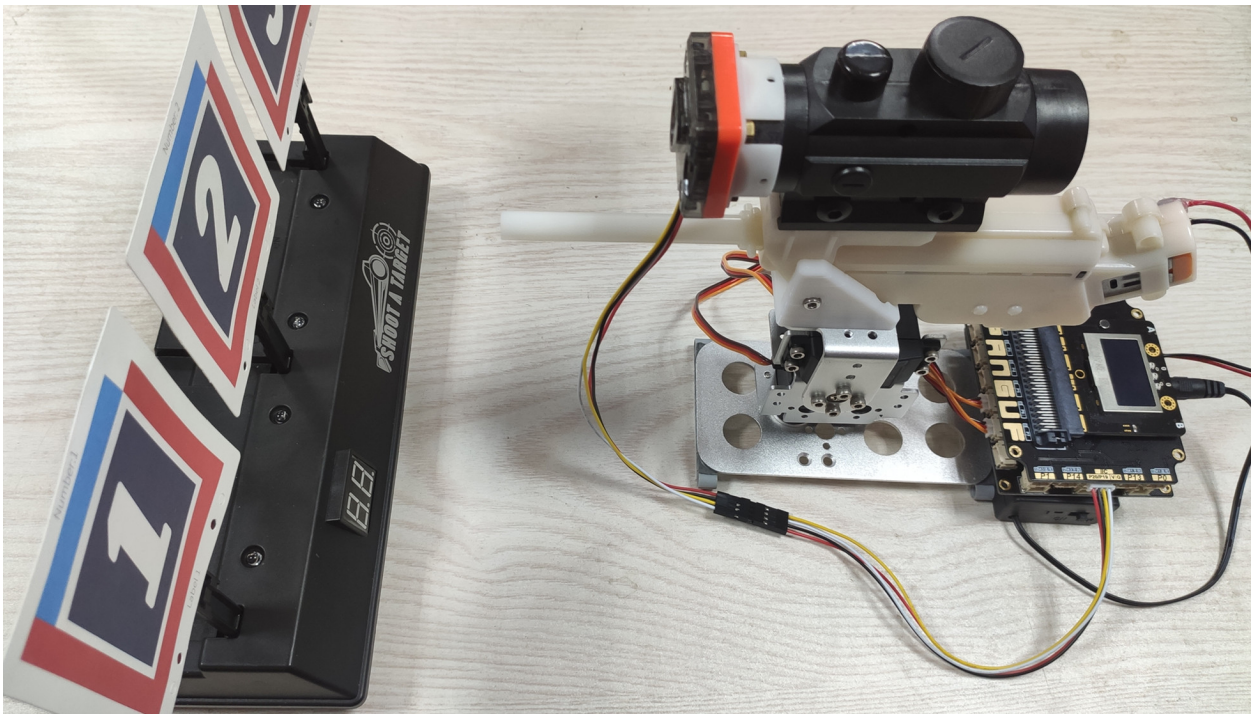
7.1 自动炮台

7.1.1 简介

本项目是改装狄仁杰小车平台制作的自动寻找并射击靶子的炮台。通过安装在炮台前端的 MU 识别靶子，精确定位其坐标，反馈控制水弹枪射击。

7.1.2 自动炮台组成

- 水弹发射器
- 舵机云台
- MU3 视觉传感器
- 掌控板 + 盘古斧
- 聚合物锂电池 + 18650 动力锂电池供电
- 改装可被识别的靶子

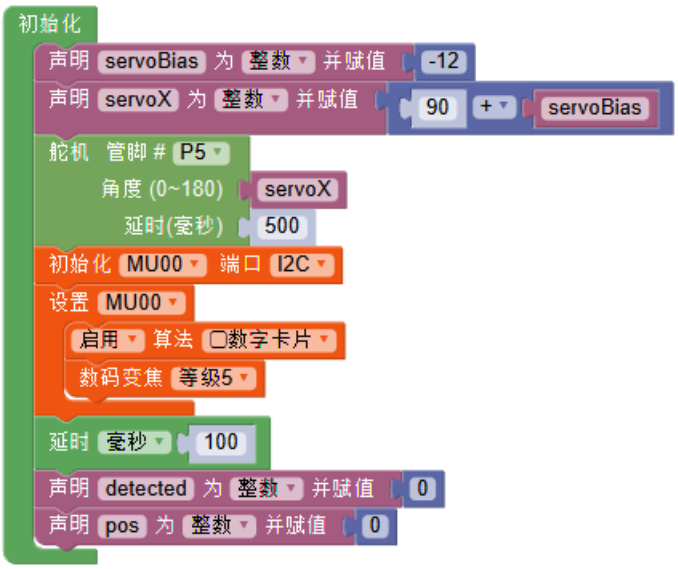


这个就是我的自动炮台，采用 MoonBot 的金属钣金制作舵机云台，实现双舵机控制炮台两个自由度的运动。额外加了一个固定水弹枪体的 3D 打印支架和固定用的钣金底座。将 MU 可识别的数字卡片贴在原本的电动靶上，同时还增大了目标更容易射中🄴。。。

7.1.3 程序解析

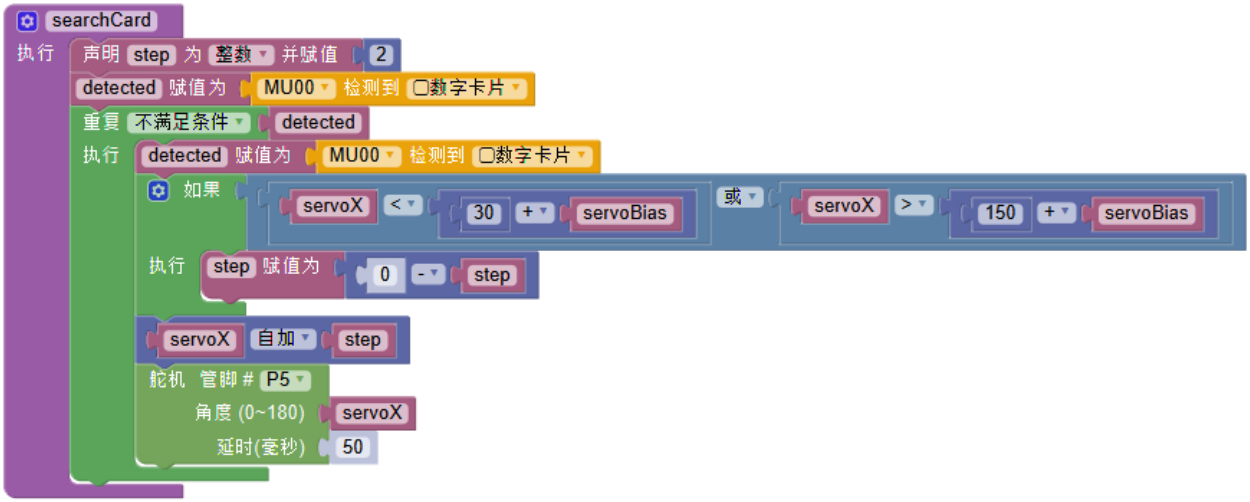
- 初始设置程序

初始化中对将要用到的外设进行设置，包括舵机和 MU。



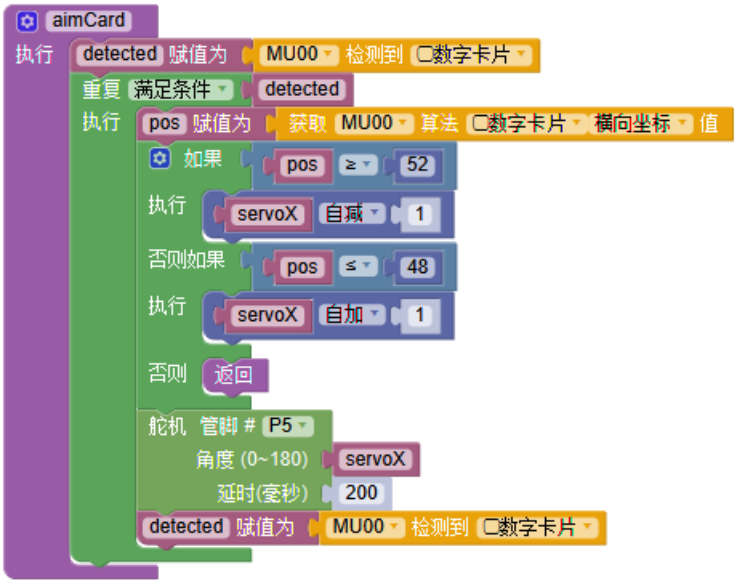
- 寻找卡片函数

寻找卡片采用舵机左右移动的方式，配合上下微调，找寻 180 度视角范围内的卡片。



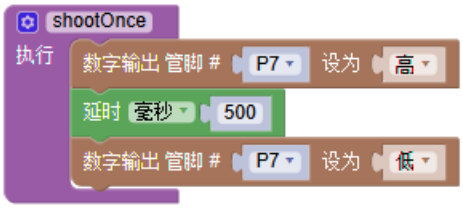
- 对准卡片函数

当寻找到卡片时，进入对准卡片的程序。通过舵机移动修正炮和卡片之间的距离。



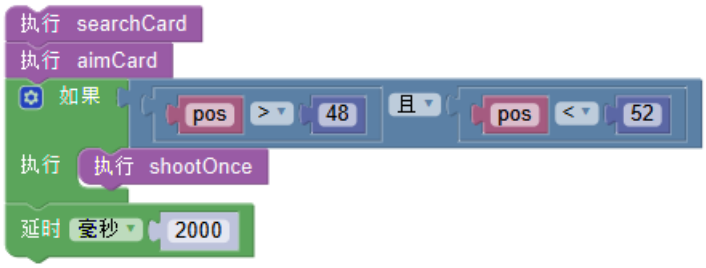
- 发射函数

比较简单，通过控制 P 脚直接输出高电平发射，约 0.5 秒时间，可以打出约 3 发水弹，如果击倒比较容易则可以时间更短些。



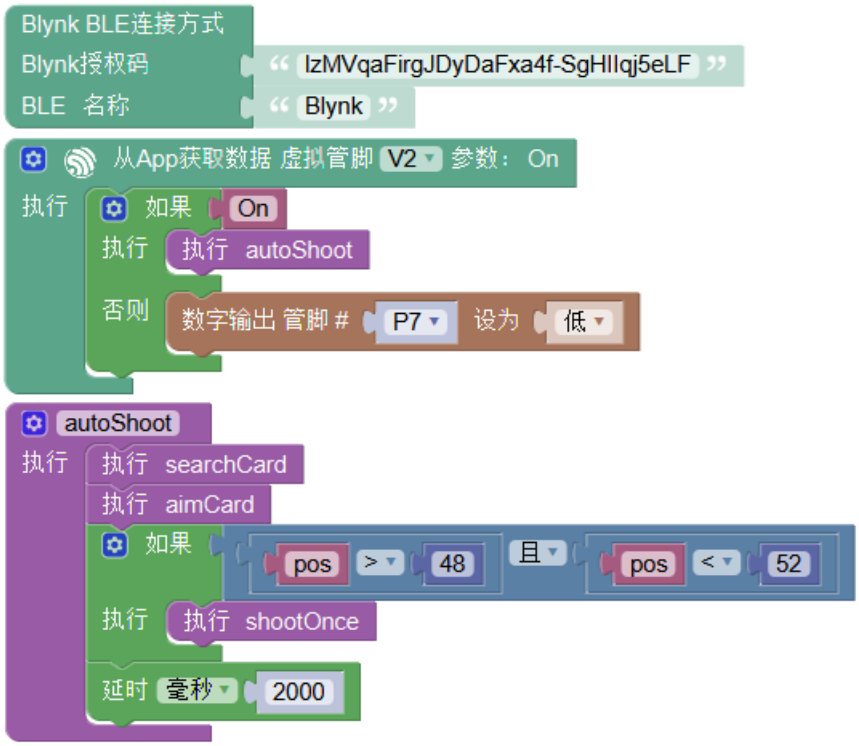
- 循环程序

将以上函数组合后就是循环程序的内容。



- 远程控制程序

另外可以加入 Blynk 远程开启炮台的程序，将循环程序作为子程序整体放入 Blynk 中。按键按下则触发循环程序执行自动发射。



7.1.4 实战演习

程序开始，舵机复位至正中。主方向为左右方向，检测并扫描。在遇到数字卡片时跳出寻找卡片，进入对准卡片程序。在确定视觉模块对准坐标 48 至 52 位置时射击，两秒后再次进入扫描。

如果加入远程控制程序，则在 blynk 端按下按键后开启整段自动射击程序。

视频

7.2 网帖汇总

掌控慧眼小 MU 系列教程

小 MUl 小狗狗，认数字

人工智能（障）——麦昆人体识别与追踪

自动跟随的麦克纳姆轮小车

小 MU 掌控板之乘法连连看

CHAPTER 8

MoonBot Kit 简介

MoonBot Kit 是摩图科技出品的一套教育套件。产品包含多种硬件模块，配合钣金、塑料外壳等结构件可以搭建各式机器人。通过手机端、电脑端的编程软件，结合 AI 技术，让青少年打造人工智能机器人，学习 STEAM 教育的课程内容，向未来的卓越工程师迈进。

CHAPTER 9

MoonBot Kit 硬件指南

MoonBot Kit 目前共有 9 种硬件模块。可以通过米思奇或 Arduino 对模块单独操控，或者将模块拼搭成具体的形态后设计交互程序。以下模块介绍中会单独介绍每个模块的引脚定义和简单的米思奇程序，帮助用户快速上手。

软件方面的详细教程参考

MoonBot Kit Mixly 教程

下载所有硬件示例程序

MoonBot 硬件示例

9.1 主控模块

9.1.1 简介

主控模块是机器人的交互核心。主控芯片为 Atmega1280，兼容 Arduino。板载舵机、电机和通用输入输出的接口，可以连接外部设备。同时板载按键、LED 等外设，便于快速编程控制。

9.1.2 参数

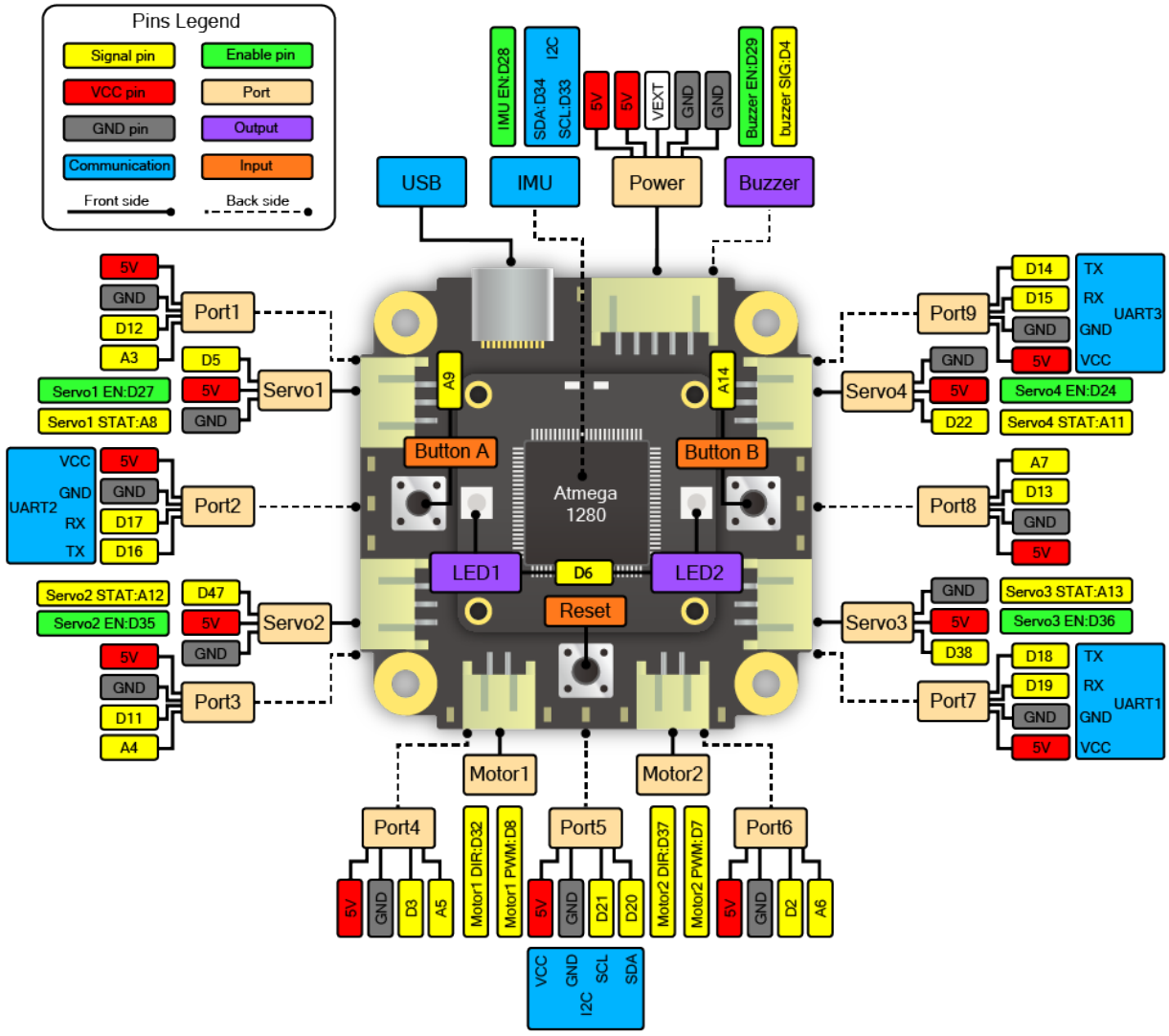
尺寸：53 x 53 x 17.6 mm

主控芯片：ATmega1280

接口：4 路舵机、2 路电机、9 路通用输入输出

板载资源：按键、LED 指示灯、蜂鸣器、电子罗盘、加速度计

外设和接口图



9.1.3 使用示例

主控 LED 和按键示例

按键和 LED 灯是最基础的输入和输出设备，可以用于其他设备的功能调试。首先介绍这两项，之后的调试会更方便。以下示例通过主控模块板载的 2 个可编程按键控制板载的 2 个 RGB 彩灯。

程序介绍：循环检测按键 A 和按键 B 的状态，当 A 被按下时 LED1 亮红色，当按键 B 被按下时 LED2 亮绿色，同时按下则都亮蓝色，无按键按下时关闭 LED。

实物图：

主控蜂鸣器示例

该程序介绍主控蜂鸣器的编程方法，用两种方式让蜂鸣器发出警报声。

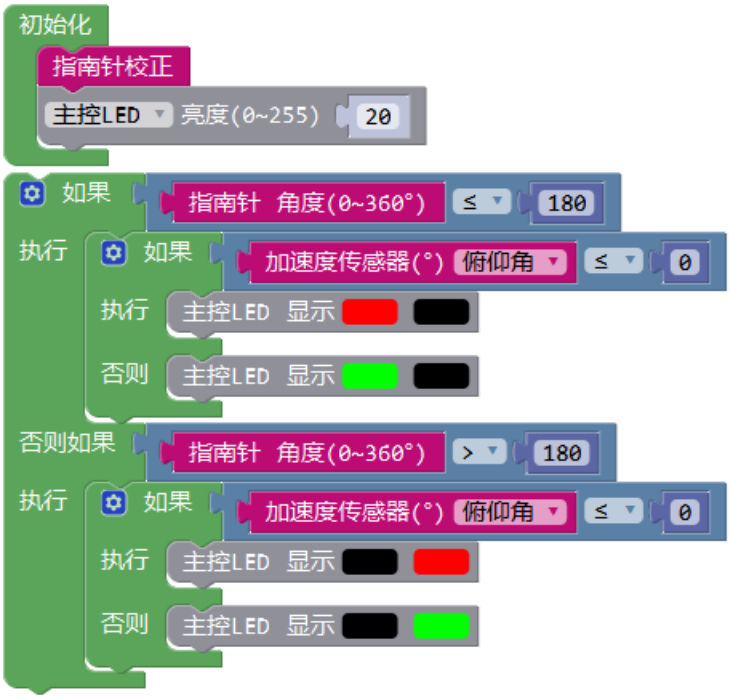
程序介绍：设置蜂鸣器的节拍为 BPM 120，即每 1 拍为 0.5 秒。蜂鸣器可以直接播放已定义的音调，或者直接播放频率，播放频率时需要加上延时，否则会被跳过。循环播放 5 次高低警报音后停止播放，结尾添加停止程序块防止循环播放。

主控 IMU 示例

IMU(Inertial Measurement Unit) 是指惯性测量单元，含有陀螺仪、加速度计、指南针等一种或多种传感器，主要用于测量当前姿态，常用于无人机、机器人等设备。主控模块的 IMU 含有加速度计和指南针，并带有温度补偿，可以通过这些传感器反馈知道机器人当前是否有翻倒、跌落、震动等状态。以下示例简单介绍主控的 IMU 使用方法。

程序介绍：初始化时校正指南针并调整主控灯光颜色，循环程序通过判断指南针和加速度计的角度定义 4 个方位，通过灯光颜色来指示方位。

实验现象：主控复位后在空中画 ∞ 来校正指南针，当 LED 灯闪烁时完成校正，此时将主控平放，指南针 180 度为正南方向，向左/右旋转则左/右边灯亮，俯/仰则亮绿/红色。



9.2 视觉模块

9.2.1 简介

视觉模块是一个带有特定视觉算法的识别模块。

模块与主控之间通过串口连接，通过主控编程控制视觉模块数据传输，也可通过 wifi 连接手机端 app 实现远程编程功能。

9.2.2 参数

尺寸：37 x 37 x 15 mm

主控芯片：ESP32

摄像头：OV7725

视角：85°

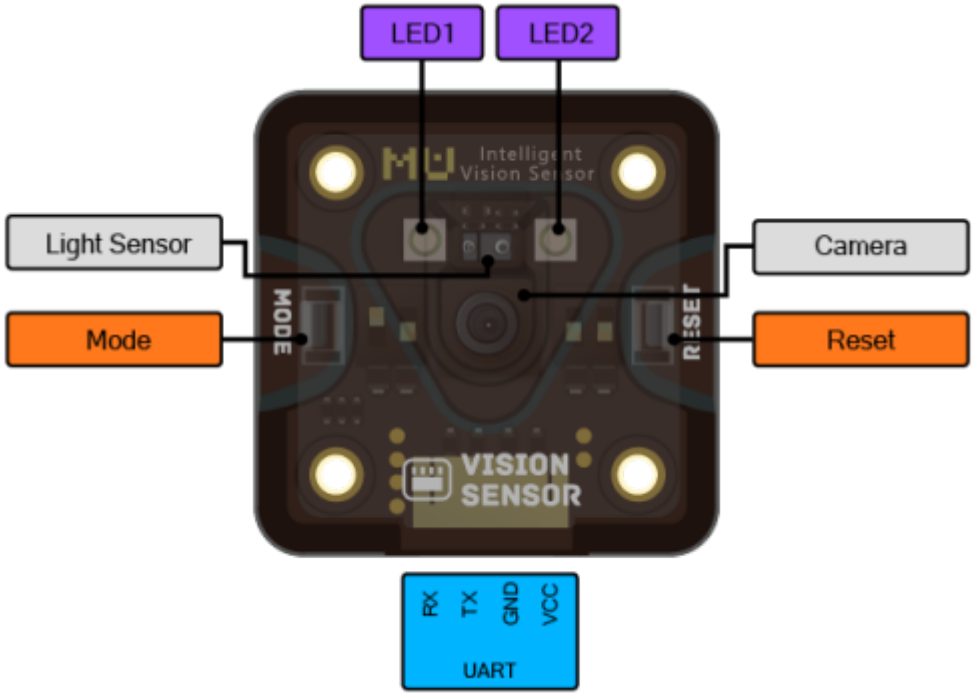
光线传感器：APDS9960

板载资源：按键、LED 指示灯

连接方式：UART、wifi

接口：PH2.0 4P

接口图



9.2.3 使用示例

串口开发

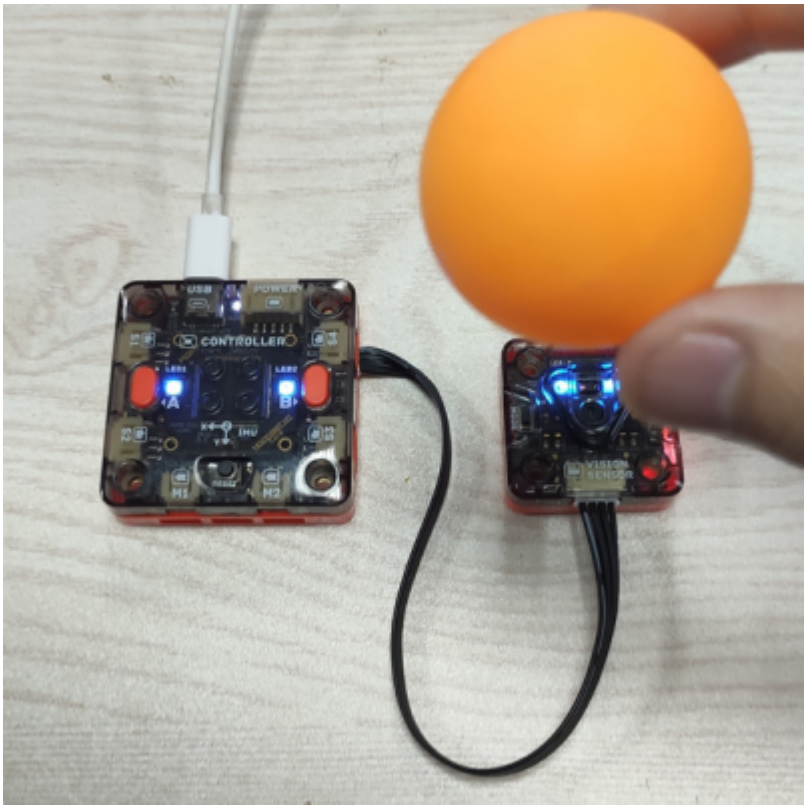
视觉模块可和主控连接，通过主控编程使用指令控制决定视觉模块的识别功能，并通过串口向主控传输数据。

硬件连接：将视觉模块连至主控 P9 口，作为 UART3(Serial3) 串口设备运行。

程序介绍：在初始化中开启主控和视觉模块间串口 Serial3，波特率默认 115200，初始化视觉模块至 Serial3，启用球检测算法。循环程序中，将主控的亮灯状态配置为和视觉模块的灯相同，即检测到球亮蓝灯，未检测到球亮红灯。

实验现象：先按视觉模块的 RESET 键初始化视觉模块，板载两个 LED 灯会常亮进入等待指令状态，然后初始化主控，主控的初始化程序就向视觉模块发送指令，此时视觉模块进入检测球状态，两个灯闪红色，此时主控进入循环接收状态。当检测到球时，主控和视觉模块的 LED 都亮蓝色。

实物图：



注意 MoonBot 视觉模块和 MU 视觉传感器的区别，视觉模块默认和主控采用串口连接，使用 Mixly 或 Arduino 进行开发，且独有 wifi 连接 MU Bot App 的功能，初始化程序参考以上示例。

该程序较为简单，主要为了展示视觉模块的连接和程序初始化。更多编程块和示例程序可参考

MU 3 Mixly 教程

wifi 连接 app

视觉模块自带 wifi，可以连接 MU Bot App，进行遥控或手机端编程等，详情参考

MoonBot Kit MU Bot App 教程

光线传感器

在最新版本视觉模块的固件中，加入了板载光线传感器的控制功能。该光线传感器为红外传感器的点阵，可以测得环境光强度，或用红外测距判断前方物体的移动信息，即可以识别挥动手势。以下示例展示视觉模块的手势识别功能。

硬件连接：同上

程序介绍：初始化中开启串口 Serial 和 Serial3，Serial 用于主控和 PC 通讯，Serial3 用于主控和视觉模块通讯。启用光线传感器的手势检测功能。循环程序中，循环检测手势，若识别到手势，则判断具体的手势类型并通过串口发送至 PC。

实验现象：本示例未使用视觉算法，所以初始化视觉模块后板载 LED 未开启。打开 Mixly 的串口监视器，当手在视觉模块前约 20cm 处划过时，串口打印手划过的方向，就是此时的挥动手势。

9.3 电池模块

9.3.1 简介

电池模块用于主控供电，可驱动各类执行器和传感器，最大输出功率可达 14W。

模块含有锂电池电压转换、电量管理、过载保护、充电等单元，可直接通过 USB type-c 接口充电（不可传输数据）。

9.3.2 参数

尺寸：67.6 x 56 x 33.7 mm

电池类型：聚合物锂电池

电量：7.4V 1800mAh

输出功率：最大 5V 2.8A

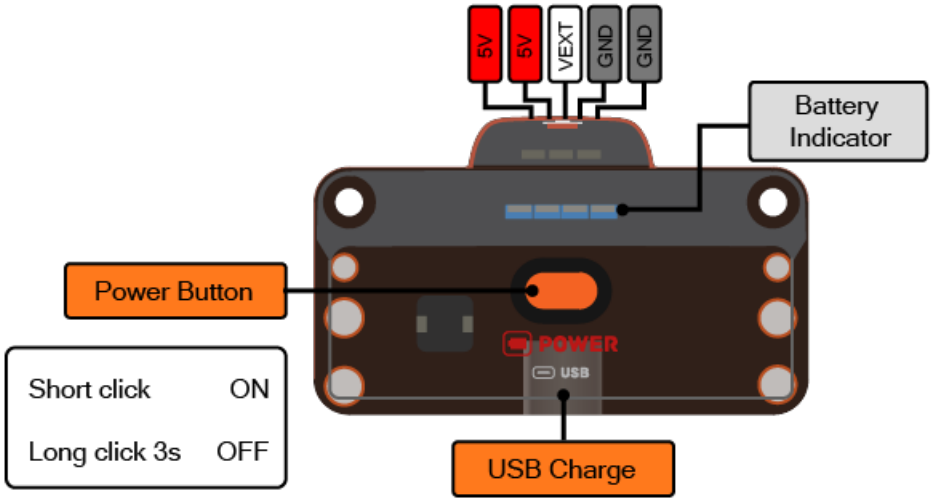
输出接口：PH2.0 5P

充电功率：最大 5V 1A

充电时间：约 3.5 小时

充电接口：USB type-C

接口图



9.3.3 使用说明

- 电池短按开启，长按 3 秒关闭，无连接时 30 秒后自动关闭。
- 顶部橙灯为充电指示，白灯为电源指示，红灯为 USB 接入指示，4 颗蓝灯为剩余电量指示。
- 电池通过 PH2.0 5P 线连接主控来给主控供电，可以同时驱动舵机、电机等大功率执行器，注意最高输出为 5V 2.8A。
- 电池通过 USB-C 接口充电，最大充电电流 1A，充满需 3.5 小时。电脑端充电电流最大约 0.6A，建议配正规的 USB 充电器。该 USB 接口不具备通信功能。

9.3.4 注意事项

- 电池模块只可用于和主控模块连接，不得改装，防止短路。
- 电池含有电路保护模块，但日常使用电流不过大，不过放等有利于延长电池寿命。
- 锂电池为易燃易爆产品，防止挤压、跌落、接触水、高温或金属异物等行为。
- 长时间不用请将电池单独存放，电量控制在 50% 左右，并防止误按按键开启电池。

9.4 电机模块

9.4.1 简介

电机模块内含减速电机和测速编码器。外部有一个主动轮和一个被动轮，主动轮连接减速电机，被动轮则由螺丝和轴承连接。轮子外接拼装式的履带，用两个电机模块即可搭出一个履带车底盘。

9.4.2 参数

尺寸：109 x 40 x 39.1 mm

减速比：120:1

空载转速：100rpm

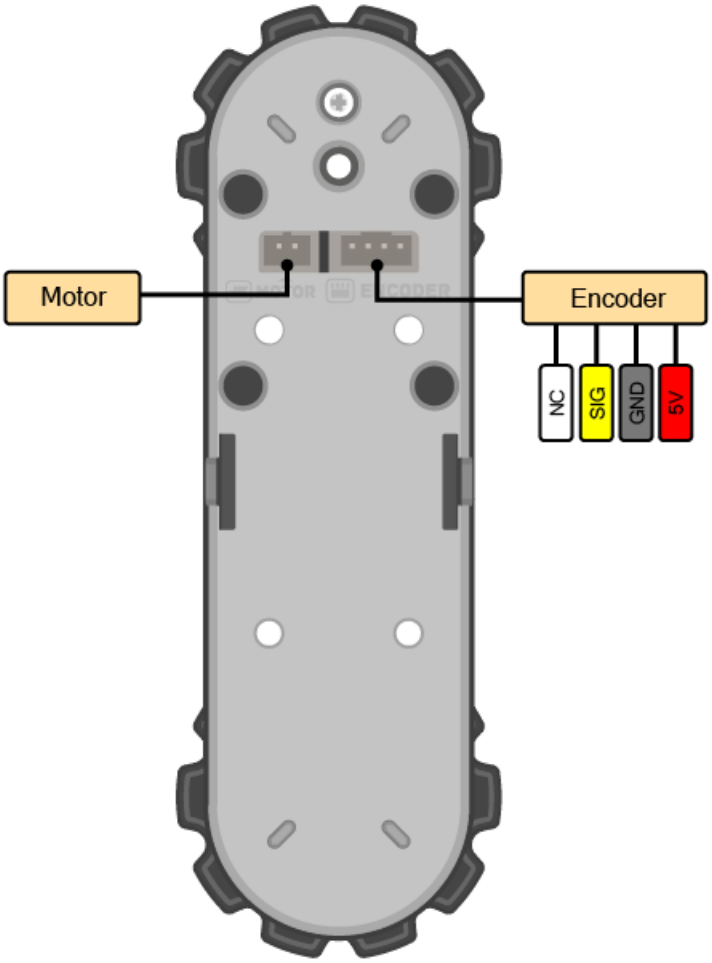
额定转速：70rpm

额定电流：300mA

测速器：光电编码器

接口：PH2.0 2P + PH2.0 4P

接口图



9.4.3 使用示例

底盘控制

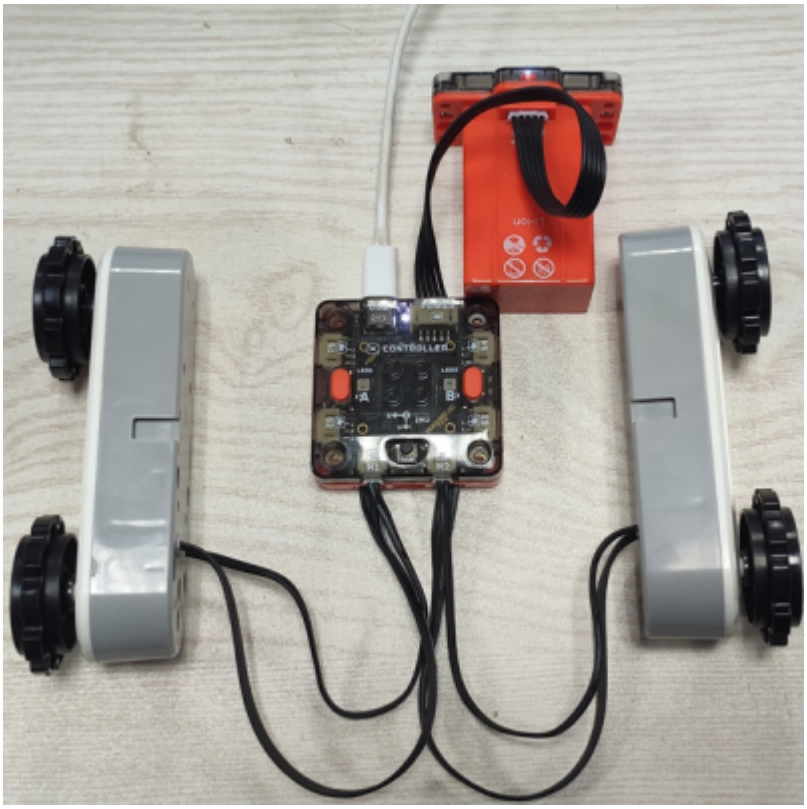
在车形、人形、机械臂等形态下，均可直接控制电机底盘前进、后退、转弯等。通过以下程序测试底盘的运行功能。

硬件连接：将电机和测速器接到主控对应接口，电机 M1 对应测速器 P4，电机 M2 对应测速器 P6。电机为大电流设备，主控必须连接电池模块供电来驱动电机。主控可同时连接电池和 USB 调试，此时使用电池供电。

程序介绍：初始化设定底盘的方向、校正直行偏移、距离和转弯半径。底盘可设定前进、后退、左转、右转，输入距离/角度和电机转速。最后停止程序防止循环运行。


初始化
底盘控制 翻转方向
底盘控制 直行偏移校正(%) 100
底盘控制 直行距离校正(%) 100
底盘控制 转弯角度校正(%) 100
底盘控制 前进(cm) 10 转速(0~100RPM) 30
底盘控制 后退(cm) 10 转速(0~100RPM) 30
底盘控制 左转(°) 90 转速(0~100RPM) 30
底盘控制 右转(°) 90 转速(0~100RPM) 30
底盘控制 停止
停止程序

实物图:



电机独立控制

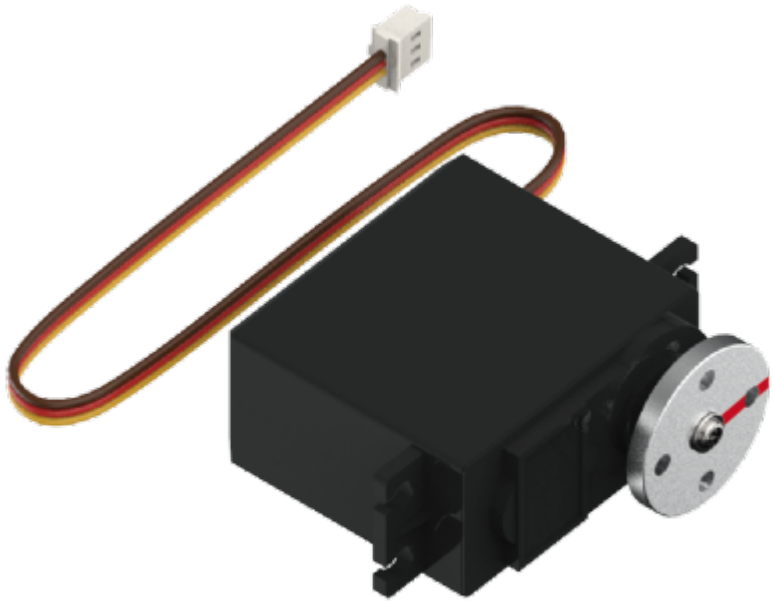
除了集成的底盘控制方式外，也可直接控制电机 1 和 2。

硬件连接：同上

程序介绍：初始化打开串口，用来接收测速器数据。直接写入值通过 PWM 控制电机的电压，让底盘前进 2 秒。然后写入带编码器测速的电机速度值，让底盘原地旋转 2 秒，通过串口返回测速值。通过写入转速 0 关闭电机，停止运行。

9.5 舵机模块

9.5.1 简介



舵机是通过 PWM 信号控制转动角度的执行器，内含直流电机、减速器、角度反馈和电流控制电路。

9.5.2 参数

尺寸：54 x 20 x 47.2 mm

舵机：55g 金属齿轮舵机

扭矩：9.4kg.cm

可动角度：180 度

工作电流：250ma

堵转电流：1A

接口：PH2.0 3P

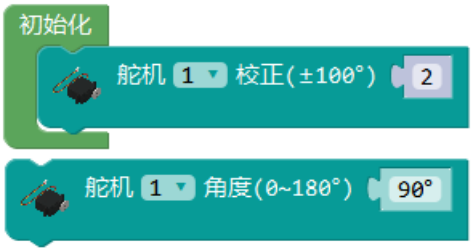
9.5.3 使用示例

舵机校正

舵机转动范围为 0~180 度，顺时针方向为角度增大，初始角度设定为 90 度，舵机盘上带有一个向前的标记。

舵机为角度控制装置，通过内部电位器反馈来获得当前角度。由于电位器角度和舵机盘安装等影响，舵机初始角度可能会有较小的偏差，通常小于 10 度，可通过程序来校正舵机的初始角度。

程序介绍：初始化设定舵机角度校正，将舵机转至 90 度，观察实际舵机盘角度是否有偏差，调整校正角度值确定该舵机的校正角度。注意校正角度值为舵机的初始值而不是端口 S1 的初始值。如果将该舵机换端口则需要将程序中也改为对应端口。

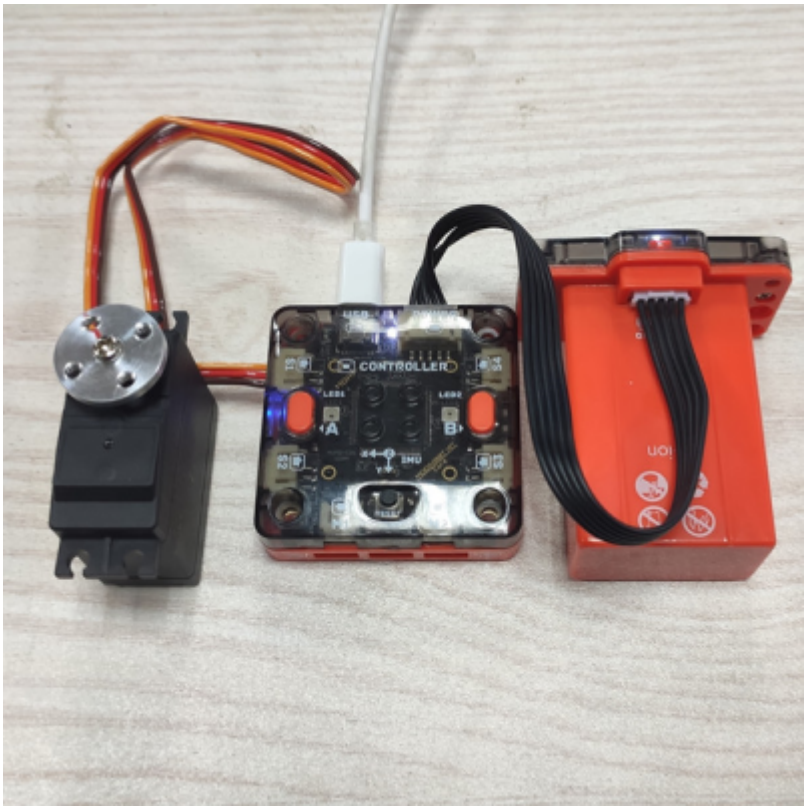


舵机转动

舵机转动有两种方式，设定转至角度和时间，或者预设角度和速度后同步移动。前一种方式通常用于单舵机转动，后一种方式用于多个舵机同时转动。通过以下示例展示两种转动方式。

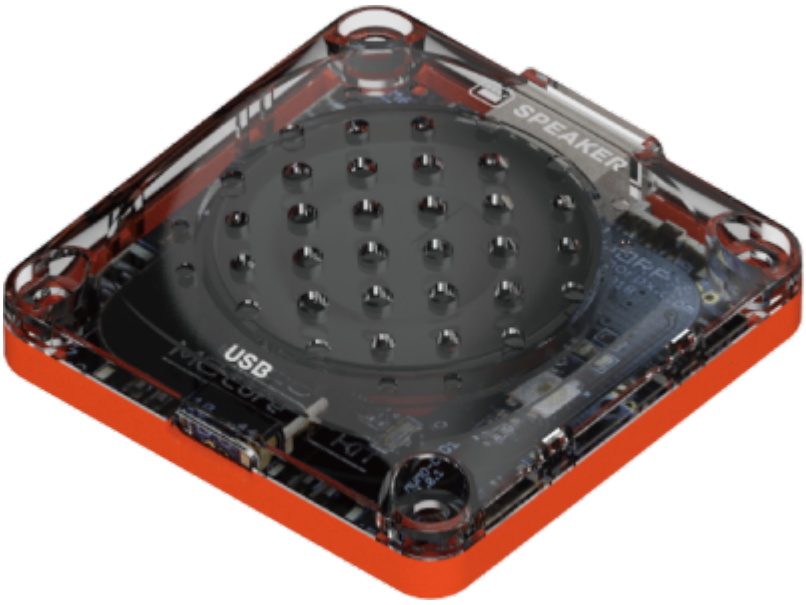
程序介绍：初始化设定舵机校正角度和转动方向。循环程序先使用舵机角度和延时控制舵机转至 30 度，再使用同步移动方式控制舵机快速转至 150 度。

实物图：



9.6 扬声器模块

9.6.1 简介



扬声器模块是一个通过串口指令控制的 mp3 模块，可通过 USB-C 口连接电脑放入 mp3 声音文件，后通过主

控控制播放。

9.6.2 参数

尺寸：48 x 48 x 11.6 mm

输出功率：1W

支持文件：mp3

存储空间：16MB

接口：PH2.0 4P, USB-C

接口图

9.6.3 使用示例

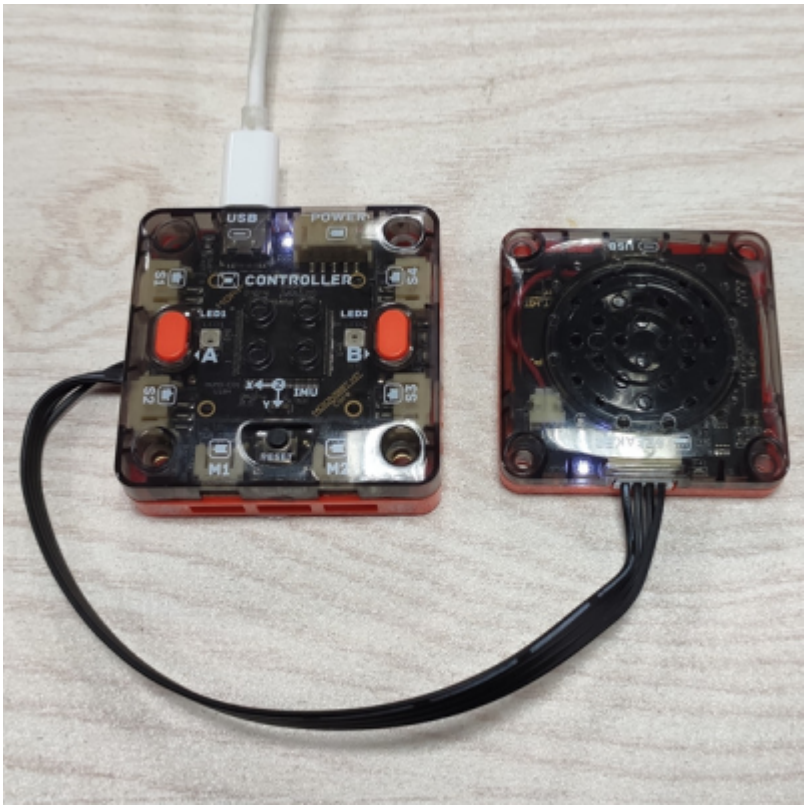
按键控制播放

扬声器模块可与主控连接，通过主控控制播放内置语音。

程序介绍：让扬声器随机播放预置音乐，通过按键使其暂停/播放。将扬声器连接至主控口，设置播放模式为随机播放，自定义扬声器音量大小，循环检测按键 A 的状态，当 A 被按下时，扬声器播放/暂停。


初始化
扬声器 连接至 端口 2
扬声器 设置播放模式 随机播放
扬声器 音量(0~32) 15
如果 按键 A 被按下
执行
扬声器 播放/暂停
重复 满足条件 按键 A 被按下
执行
否则如果 按键 B 被按下
执行
扬声器 播放/暂停
重复 满足条件 按键 B 被按下
执行

实物图:


CONTROLLER
A
B

音频导入

音源获取

扬声器模块通过播放音频文件使机器人发声。音频文件来源可以是：

1. 直接录音
2. 文字转语音 (TTS) 软件

直接录制的声音会更生动，在特定的场景中会让机器人更有亲和力，如待人接物的服务机器人；而文字转语音制作的声音用于各类 AI 助手，更标准通用，在问答机器人等交互场景中更加适合。

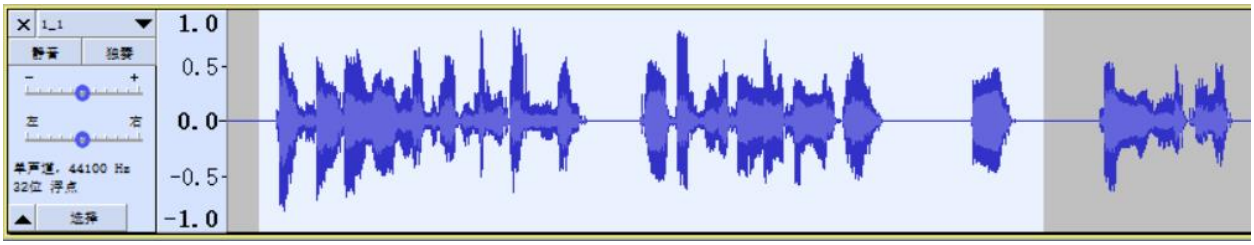
音频处理

为了使机器人的发声更加合理，将原始的语音处理后再放入扬声器模块。比如截取片段，标准化音量，改变音高，压缩大小等操作。

这里使用免费的Audacity软件处理音频。在顶栏菜单中的文件-导入-音频导入原始音频，可看到音频的频谱。

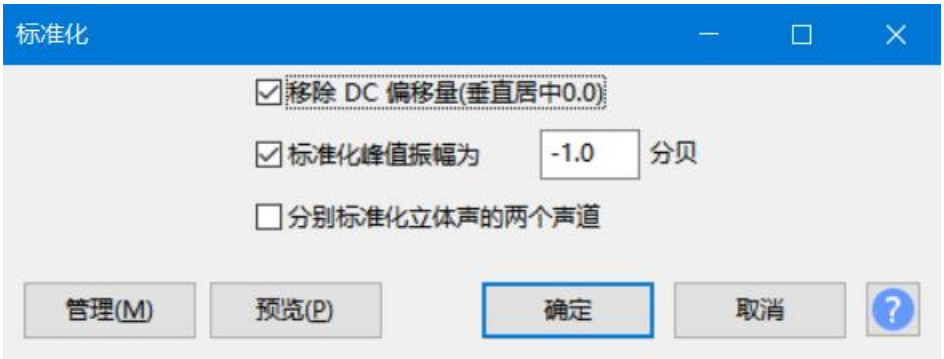
截取片段

在语音中停顿时间过长或者只需要部分语音时，可通过截取片段裁剪语音。拖动选取语音段，**ctrl+c** 后 **ctrl+v** 复制到另一条音轨或使用 **delete** 删除。



标准化音量

在录音文件中常会出现音量大小不一的问题，可通过效果-标准化来调节声音的振幅。

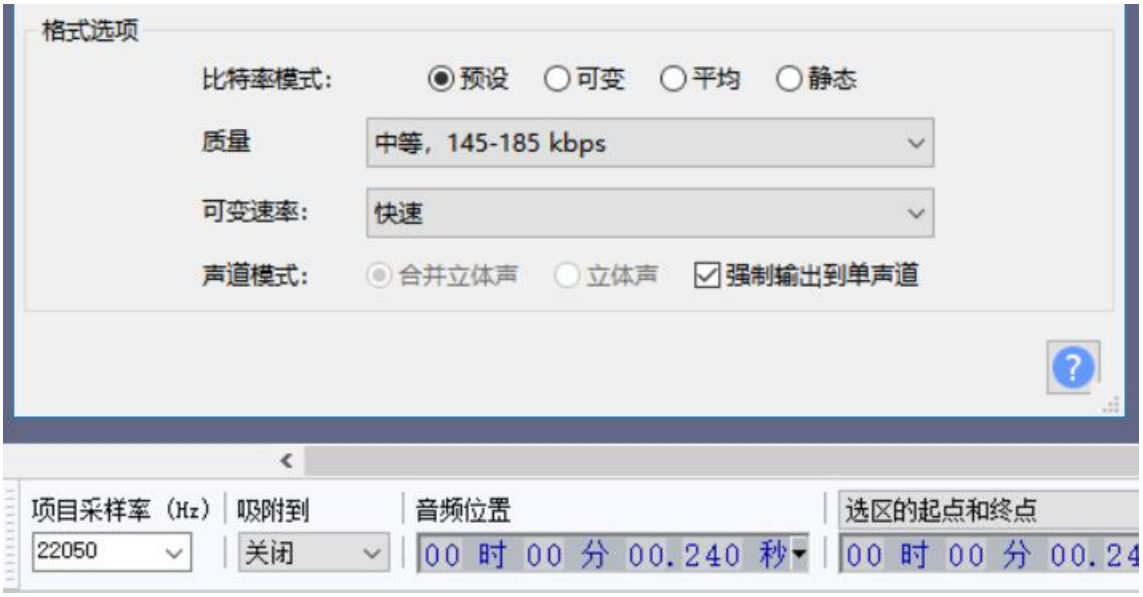


改变音高

为了使机器人的声音与人声区别开，可通过效果-改变音高来调节声音风格。将人声音高增加 30%-40% 会更接近 MoonBot 的说话风格。

压缩大小

扬声器模块有 16 MB 的存储空间，使用 mp3 格式的压缩音频文件，通常可存储上百个文件。影响音频文件大小的主要是采样率和输出比特率。当音频文件数量多、时间长时，可通过调节项目采样率至 22050，导出 MP3 格式选择质量中等，145-185 kbps，勾选强制输出到单声道等操作有效降低文件大小。



文件导入

扬声器模块可以直接用 USB-C 线连接电脑，作为模拟 U 盘来存放音频文件，需要将音频文件重命名来调用。扬声器模块可以指定根目录文件名的前 4 位进行播放。例如根目录下 T002_ask.mp3 音频文件，在米思齐中通过编程块调用扬声器播放 T002 即可播放该文件，并可加入按键控制等方式切换或暂停。

9.7 眼睛模块

9.7.1 简介

眼睛模块由 12 颗串行 RGB LED 组成。

9.7.2 参数

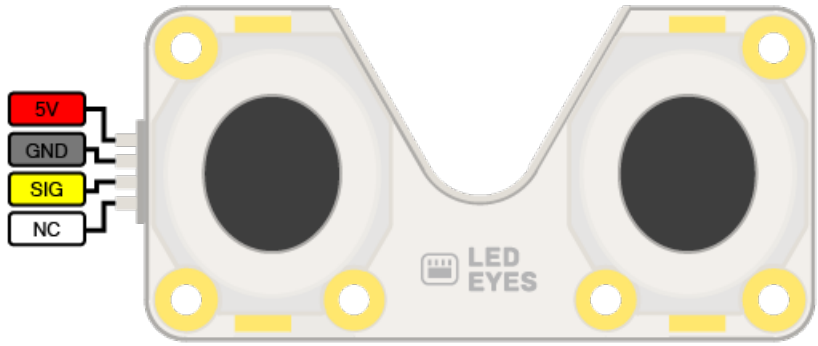
尺寸：64 x 32 x 12 mm

LED 灯类型：串行 LED

LED 灯数量：12

颜色：1600 万色

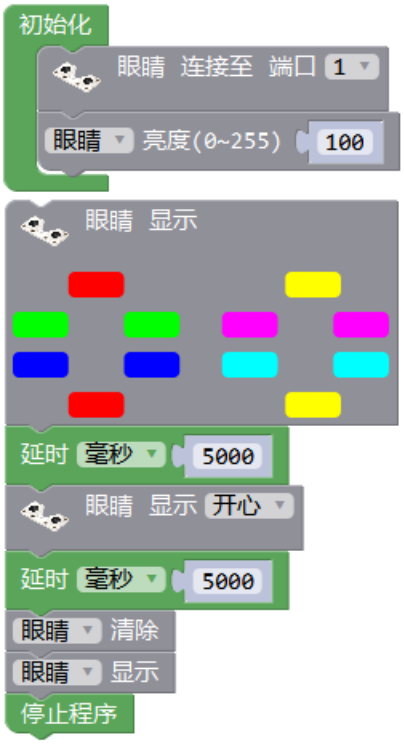
接口图



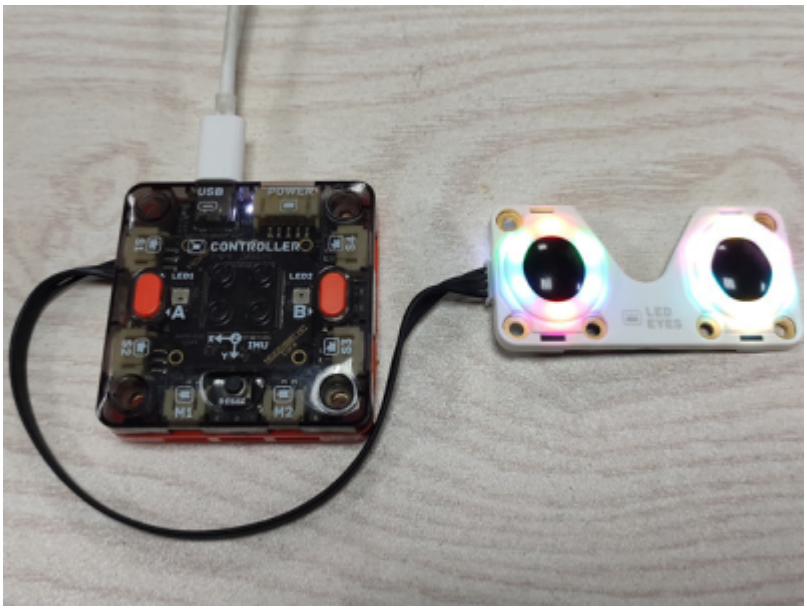
9.7.3 使用示例

点阵控制

程序介绍：将眼睛模块连接至主控口，自定义眼睛亮度。12 颗 LED 灯显示程序中对应的颜色，显示 5 秒后眼睛做出开心的表情，显示 5 秒后灯光关闭。

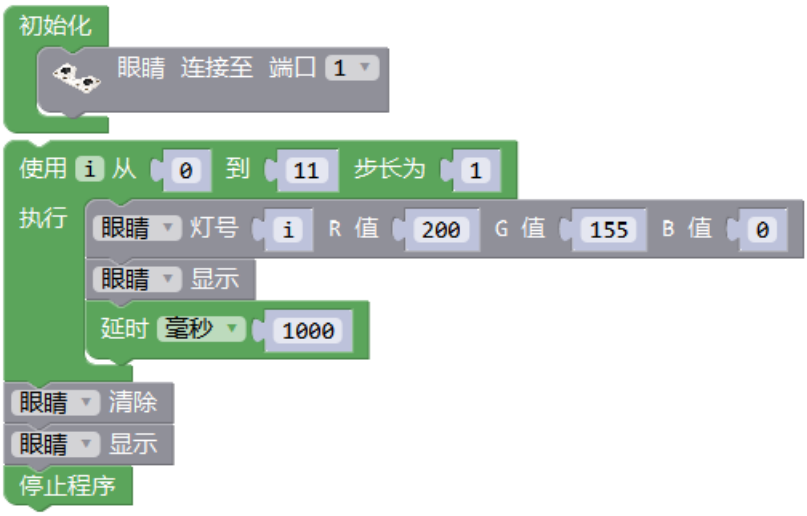


实物图：



单灯控制

程序介绍：LED 灯从第 1 颗到第 12 颗逐颗亮起设置的 RGB 颜色灯，等待 1 秒后灯光关闭



9.8 触摸模块

9.8.1 简介

触摸模块是一个单点触发式的触摸开关。

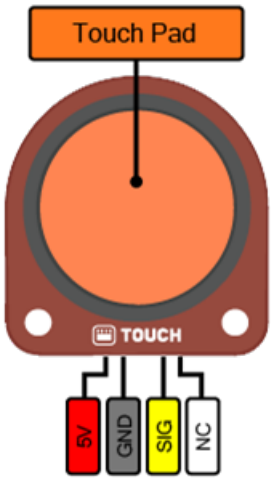
9.8.2 参数

尺寸：34 x 32 x 9.6 mm

触摸形式：非自锁模式单点触摸

触摸面规格：直径 14mm

接口图



9.8.3 使用示例

程序介绍：将触摸模块连接至主控口 P1，循环检测触摸传感器状态，被触摸时主控 LED 亮红灯，否则主控 LED 不亮灯。

实物图：

9.9 红外模块

9.9.1 简介

红外模块含有两个红外开关，可检测一定距离内的障碍。

模块有远近两种模式，用于巡线或避障等不同安装方式。

9.9.2 参数

尺寸：34 x 32 x 12 mm

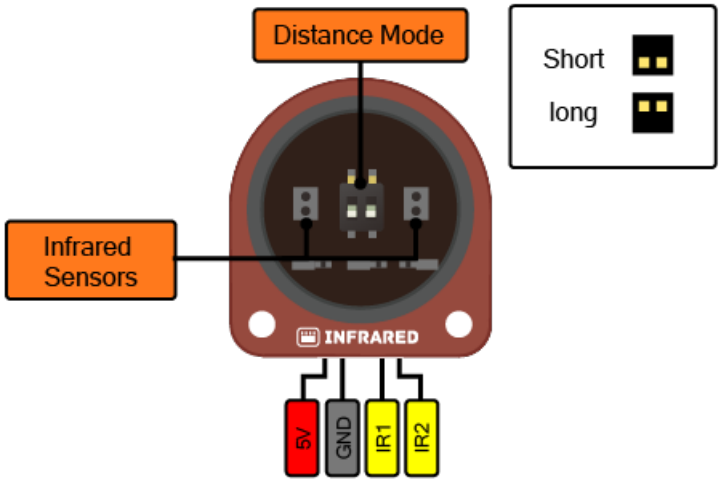
红外形式：2 路反射式红外开关

检测距离：

近距离模式约 10mm

远距离模式约 110mm

接口图

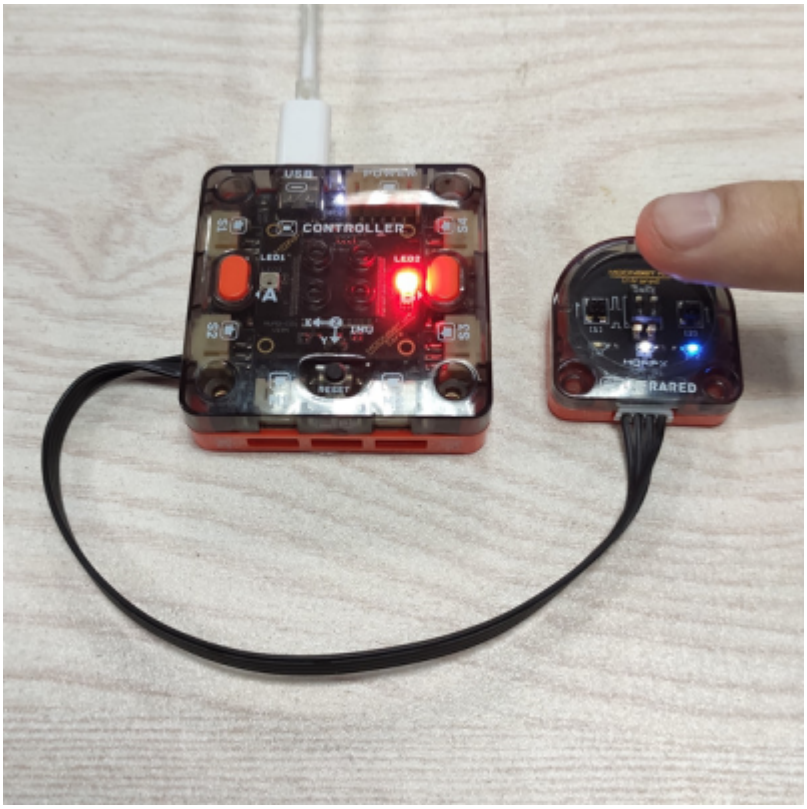


9.9.3 使用示例

程序介绍：将红外模块连接至主控口，循环检测红外传感器两路状态。当红外传感器两路同时被检测到有障碍时，主控 LED 两个灯亮红色，检测到一路障碍时，主控 LED 亮程序中对应的红灯。


初始化
红外传感器 连接至 端口 3
如果
红外传感器 读取 端口 3 引脚 1
且
红外传感器 读取 端口 3 引脚 2
执行
主控LED 显示
否则如果
红外传感器 读取 端口 3 引脚 2
执行
主控LED 显示
否则如果
红外传感器 读取 端口 3 引脚 1
执行
主控LED 显示
否则
主控LED 显示

实物图：


CONTROLLER

CHAPTER 10

MoonBot Kit 形态指南

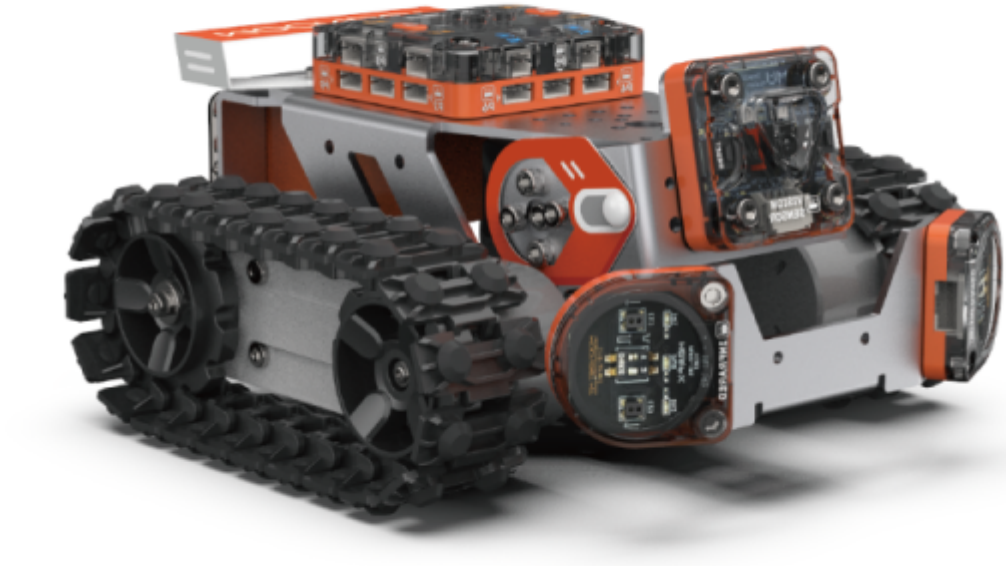
MoonBot Kit 目前共有三种官方搭建形态,从简单到复杂分别是 MoonRover 智能车、MoonMech 机械臂、MoonBot 机器人。每种形态下有独特的功能，用户可根据喜好和拼装能力选择形态搭建。通过米思奇、Arduino、MU Bot APP 等编程控制，也可以通过 APP 遥控，更可以进一步发挥创意，设计和控制自己独特的机器人。

10.1 MoonRover 指南

10.1.1 介绍

MoonRover 由钣金车身和硬件模块组成。智能车整体运动由履带底盘驱动，顶部安装主控连接各外设，前面和底部可安装避障、视觉等传感器，后部安装电池。

MoonRover 可以用于学习避障、巡线、无人驾驶等车类应用。



10.1.2 参数

尺寸：177 x 157 x 87 mm

功能

动作：头部底盘

传感：视觉避障测速

10.1.3 搭建指南

下载 MoonRover pdf 搭建指南或观看 MoonRover 搭建视频，包括基础搭建部分和拓展部分。

MoonRover 入门指南

MoonRover 搭建扩展

MoonRover 搭建视频

10.1.4 示例程序

下载 MoonRover 米思奇示例程序

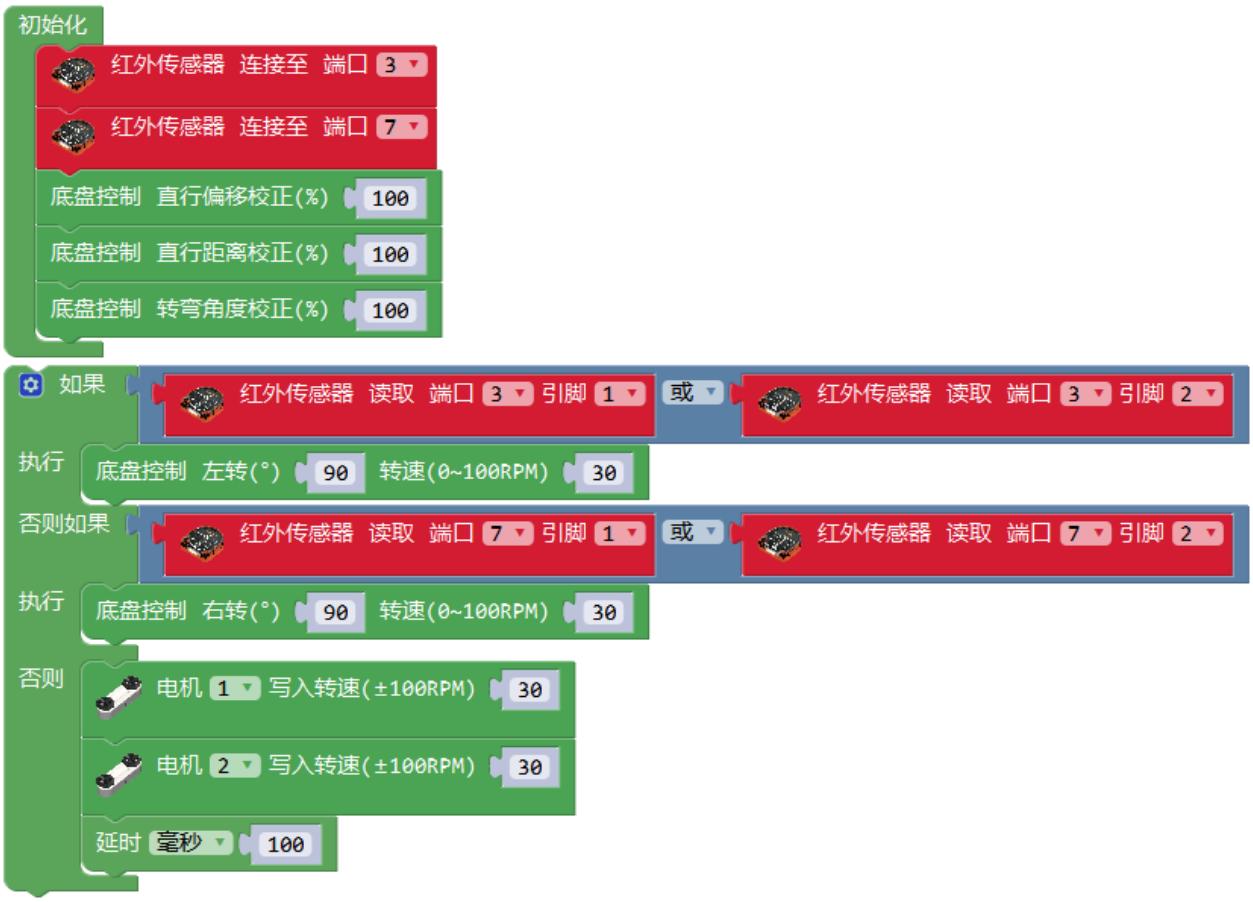
MoonRover 示例程序

避障小车

智能车前面安装了两个红外避障模块后可以成为一台避障小车。

硬件连接：搭建好小车的基础形态，如顶部图所示。车右边的红外模块连至端口 P3，左边的红外模块连至 P7，红外模块需全部切换至远距离模式。

程序介绍：初始化设定两个红外传感器的连接口，设定底盘的校正。循环程序为三种运行状态，当右边的传感器检测到障碍后则底盘左转，当左边的传感器检测到障碍后则底盘右转，未检测障碍则直行。



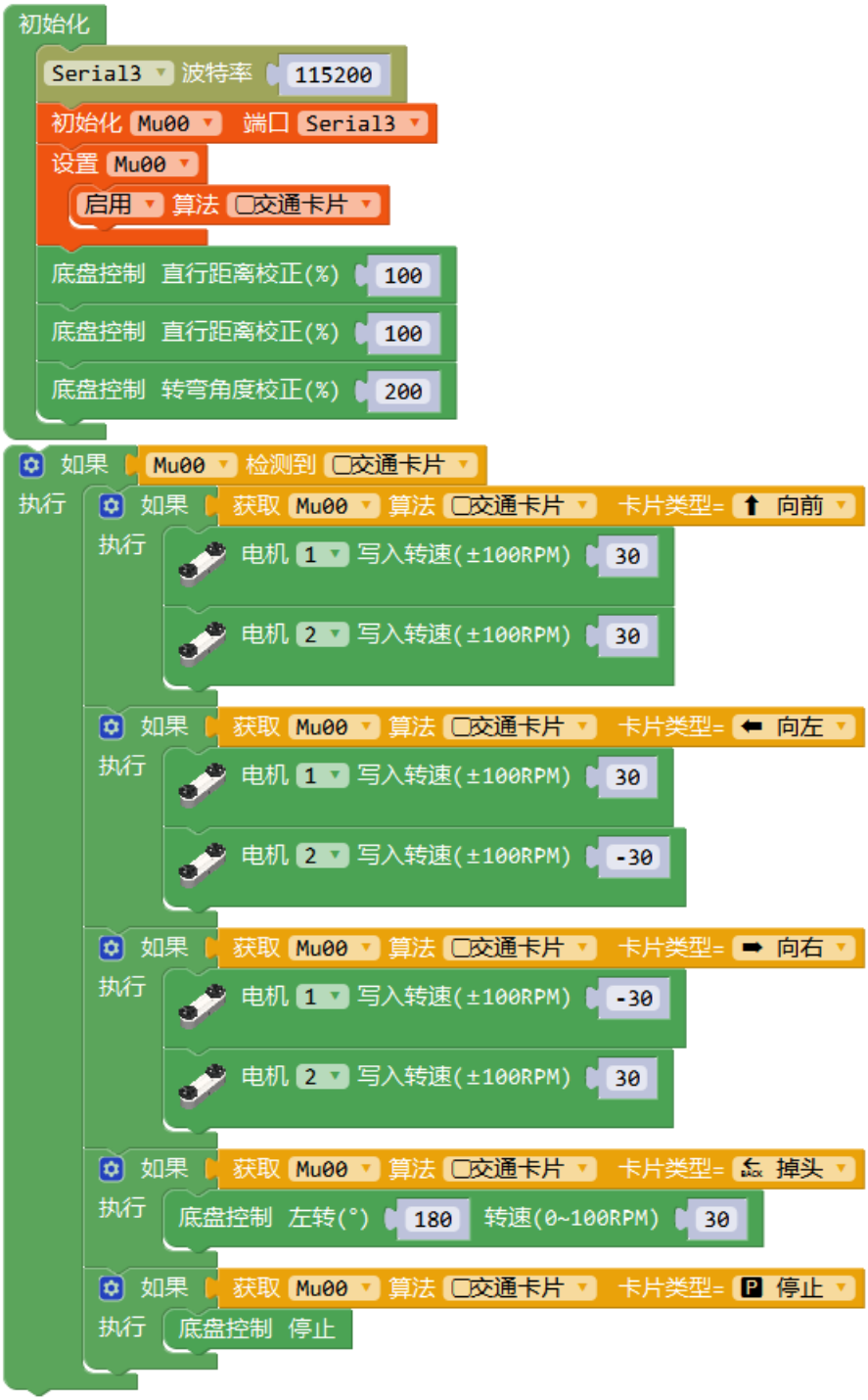
自动驾驶小车

智能车前面安装了视觉和调节视觉角度的舵机后就成为一台可以用视觉导航的自动驾驶小车。

硬件连接：按照 MoonRover 入门指南完成小车的搭建，注意视觉模块连接在 P9 口。

程序介绍：初始化设定视觉模块在串口 3（P9 口），启用算法交通卡片，底盘校正。循环程序中视觉模块检测交通卡片，5 种交通卡片对应 5 种小车运动状态，如果检测到相应的卡片则对应运动。

实验现象：烧录程序后打开电池开关，视觉模块初始化完成后闪红灯进入检测状态，将前进卡片放在小车前 20 厘米左右处即可被识别，小车前进。在运动过程中小车可以检测到其他卡片切换运动状态。在小车运动道路上立好不同的卡片让小车自动驾驶吧。

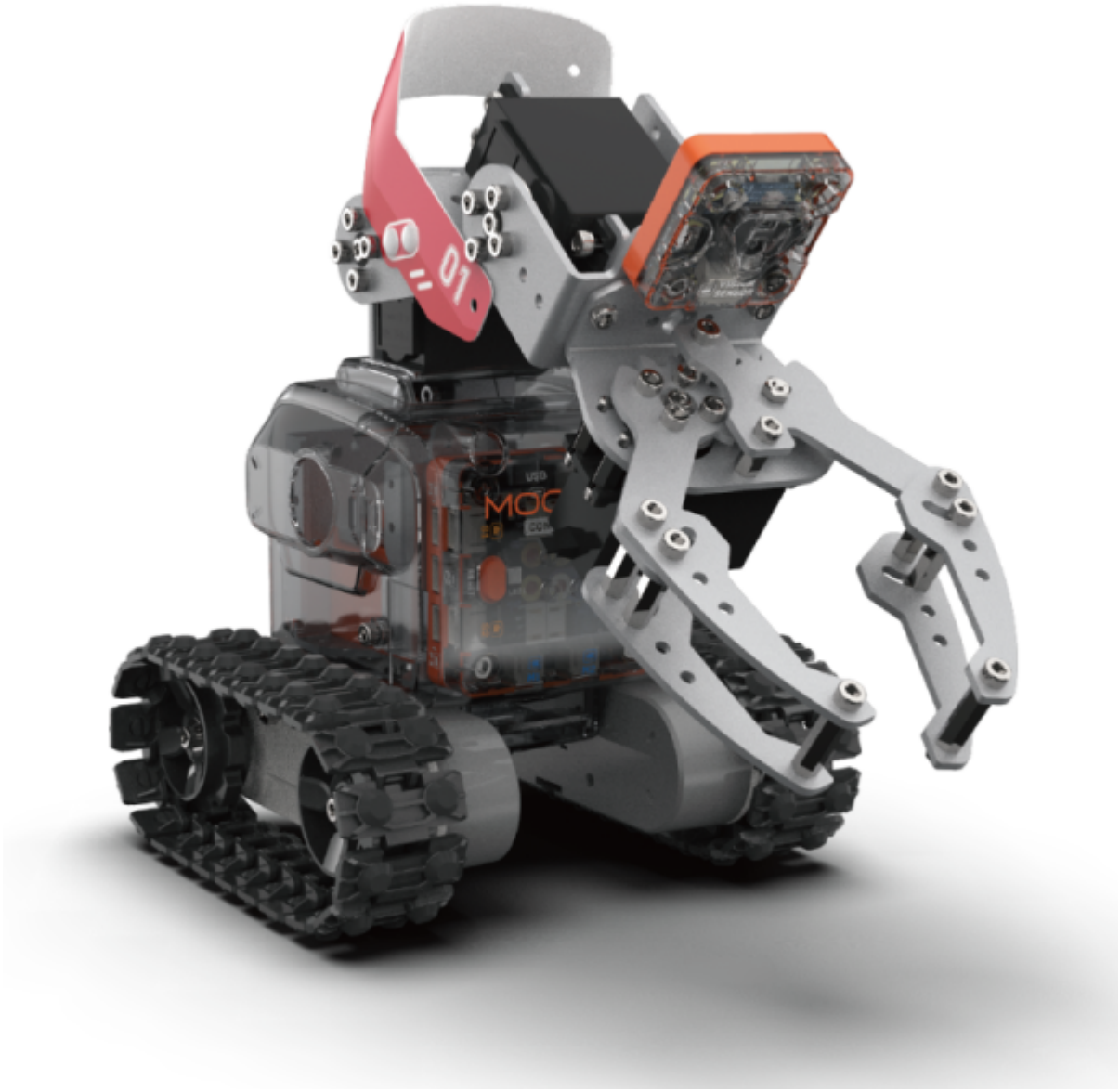

初始化
Serial3 波特率 115200
初始化 Mu00 端口 Serial3
设置 Mu00
启用 算法 交通卡片
底盘控制 直行距离校正(%) 100
底盘控制 直行距离校正(%) 100
底盘控制 转弯角度校正(%) 200
如果 Mu00 检测到 交通卡片
执行
如果 获取 Mu00 算法 交通卡片 卡片类型= 向前
执行
电机 1 写入转速(±100RPM) 30
电机 2 写入转速(±100RPM) 30
如果 获取 Mu00 算法 交通卡片 卡片类型= 向左
执行
电机 1 写入转速(±100RPM) 30
电机 2 写入转速(±100RPM) -30
如果 获取 Mu00 算法 交通卡片 卡片类型= 向右
执行
电机 1 写入转速(±100RPM) -30
电机 2 写入转速(±100RPM) 30
如果 获取 Mu00 算法 交通卡片 卡片类型= 掉头
执行
底盘控制 左转(°) 180 转速(0~100RPM) 30
如果 获取 Mu00 算法 交通卡片 卡片类型= 停止
执行
底盘控制 停止

10.2 MoonMech 指南

10.2.1 介绍

MoonMech 是一台可移动的机械臂。底盘和人型类似，由塑料外壳包裹钣金骨架搭建而成。整体移动依靠底部履带底盘，机械臂的角度和调节则依靠两个舵机。前置机械爪配合视觉反馈可以夹持各种物体。

MoonMech 可以用于学习物料搬运、投篮等工程和竞技类应用。



10.2.2 参数

尺寸：271 x 137 x 244 mm

功能

动作：机械臂机械爪底盘

传感：视觉测速

10.2.3 搭建指南

下载 MoonMech pdf 搭建指南或观看 MoonMech 搭建视频来学习如何搭建 MoonMech。

MoonMech 搭建指南

MoonMech 搭建视频

10.2.4 示例程序

下载 MoonMech 米思奇示例程序

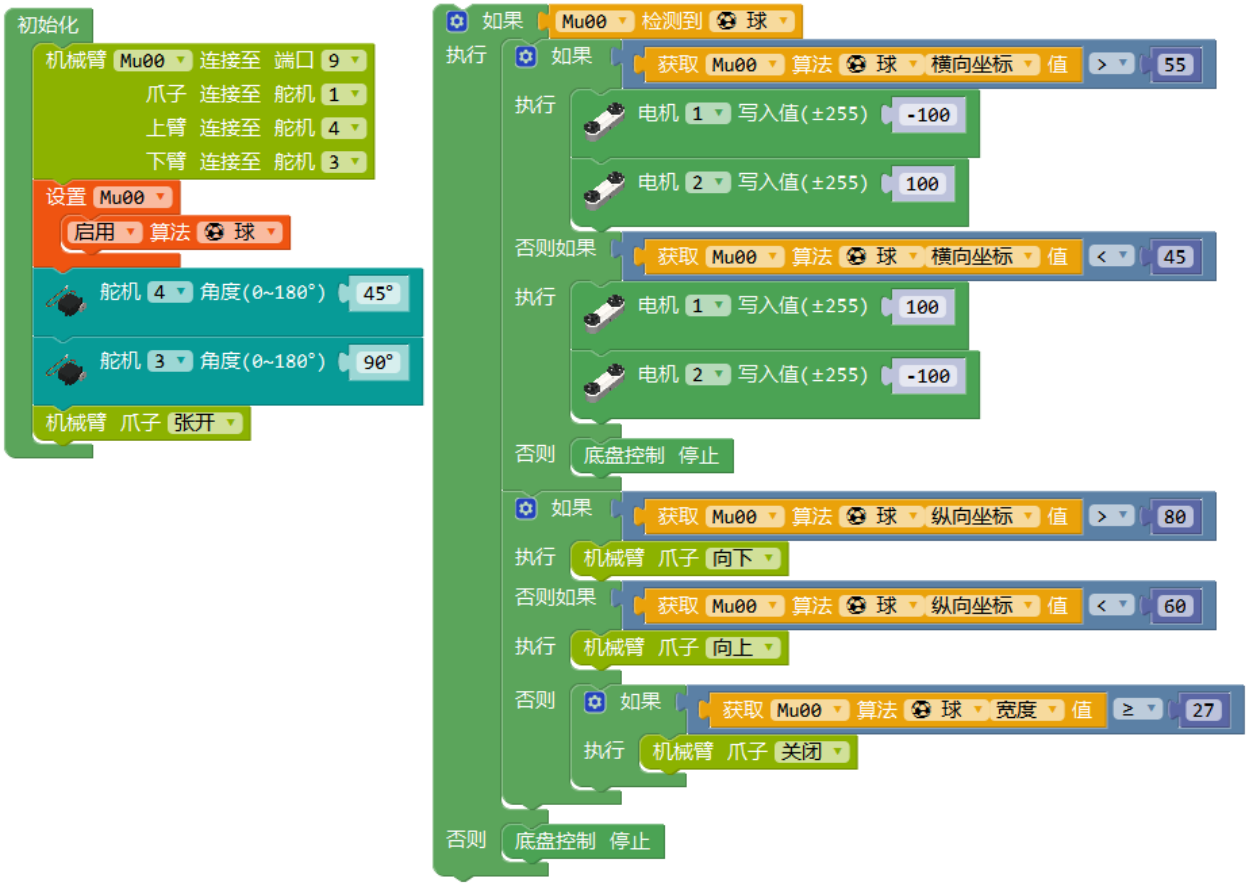
MoonMech 示例程序

抓球机器

本示例介绍机械臂通过视觉识别抓取乒乓球。

程序介绍：初始化设定 MU 和舵机的连接口，视觉启用球检测算法，舵机运动到初始位置。循环程序中视觉检测到球，则判断球的横向坐标，底盘左右移动；判断球的纵向坐标，爪子上下移动。以球的宽度估计距离，当球宽度大于或等于 27 时则机械爪抓取，就可以抓到球。

实验现象：打开机械臂后机械爪朝前张开，视觉闪红灯开始检测。将乒乓球放在机械爪前方，视觉识别到时闪蓝灯，调整球到爪子抓取范围时机械爪会自动抓取。



10.3 MoonBot 指南

10.3.1 介绍

MoonBot 是一台具有丰富的传感和交互的半人形机器人。主体由钣金骨架配合塑料外壳搭建而成，使用履带底盘整体移动，头部、手部则使用舵机实现转动。通过触摸、视觉、方位的反馈，眼睛和扬声器等都可作出交互。

MoonBot 可用于学习接待、巡逻等服务型机器人应用。

10.3.2 参数

尺寸：150 x 137 x 216 mm

功能

动作：头部手臂底盘

交互：眼睛喇叭

传感：视觉触摸测速

10.3.3 搭建指南

下载 MoonBot pdf 搭建指南或观看 MoonBot 搭建视频来学习如何搭建 MoonBot。

MoonBot 搭建指南

MoonBot 搭建视频

10.3.4 示例程序

下载 MoonBot 米思奇示例程序

MoonBot 示例程序

摇摆身体

MoonBot 的手部和脖子有舵机，履带底盘用电机模块驱动，可通过编程实现相应的动作。通过一个简单的示例让 MoonBot 活动一下。

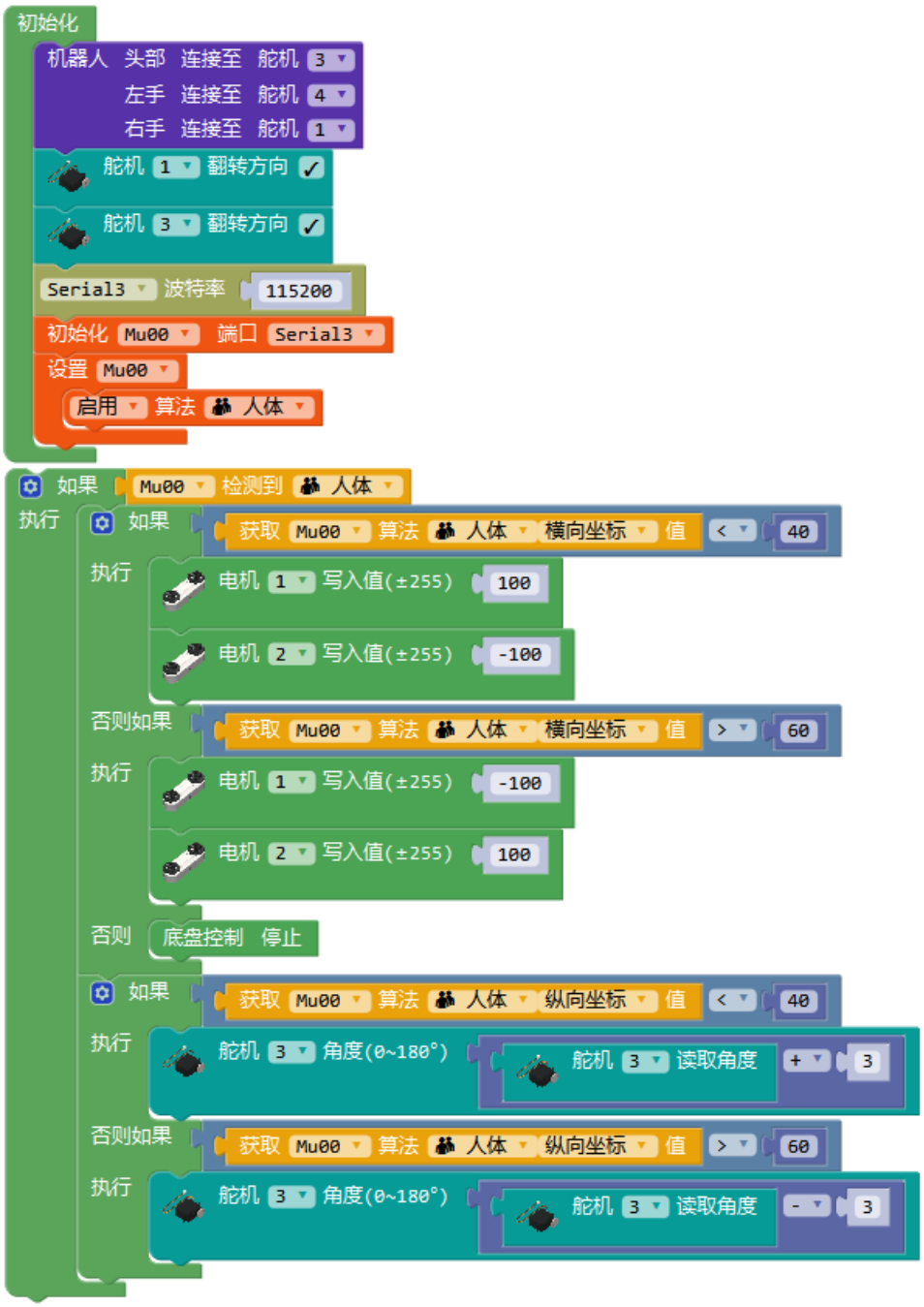
程序介绍：初始化设定头和手部的舵机端口，循环程序里一次使用机器人动作块，让机器人左手右手一个慢动作。

跟随人体

MoonBot 可以靠视觉来识别人体，并通过底盘和头部的运动跟随人体。

程序介绍：初始化连接舵机，根据实际转动方向调整舵机的方向，设置连接在串口 3 的视觉传感器，启用人体识别算法。循环程序中，当检测到人体时，根据人体的横向坐标判断底盘的转动方向，再根据人体的纵向坐标判断脖子舵机的转动方向。

实验现象：烧录完成程序后打开 MoonBot 放在桌面上，站在机器人眼前，当视觉闪红灯时未检测到，当视觉闪蓝灯时则检测到人体。人上下左右移动时机器人也会移动和转动脖子，保持脸始终朝向人体。


初始化
机器人 头部 连接至 舵机 3
左手 连接至 舵机 4
右手 连接至 舵机 1
舵机 1 翻转方向
舵机 3 翻转方向
Serial3 波特率 115200
初始化 Mu00 端口 Serial3
设置 Mu00
启用 算法 人体
如果 Mu00 检测到 人体
执行
如果 获取 Mu00 算法 人体 横向坐标 值 < 40
执行
电机 1 写入值(±255) 100
电机 2 写入值(±255) -100
否则如果 获取 Mu00 算法 人体 横向坐标 值 > 60
执行
电机 1 写入值(±255) -100
电机 2 写入值(±255) 100
否则 底盘控制 停止
如果 获取 Mu00 算法 人体 纵向坐标 值 < 40
执行
舵机 3 角度(0~180°) 舵机 3 读取角度 + 3
否则如果 获取 Mu00 算法 人体 纵向坐标 值 > 60
执行
舵机 3 角度(0~180°) 舵机 3 读取角度 - 3

CHAPTER 11

MoonBot Kit MU Bot App 教程

本文介绍 MoonBot Kit 连接手机端 MU Bot App 的教程。

11.1 MoonBot Kit APP 固件升级向导

使用 MoonBot Kit 手机编程需要在主控内烧录特定的固件才可进行编程。

本文旨在指导用户如何升级 MoonBot Kit 主控模块烧录手机编程所需固件。

11.1.1 准备工作

硬件:

- MoonBot 开发者套件
- PC (Windows、Linux 或 Mac OS)

软件:

- Arduino 官方 IDE
- MoonBot 遥控器 Arduino 固件或源码

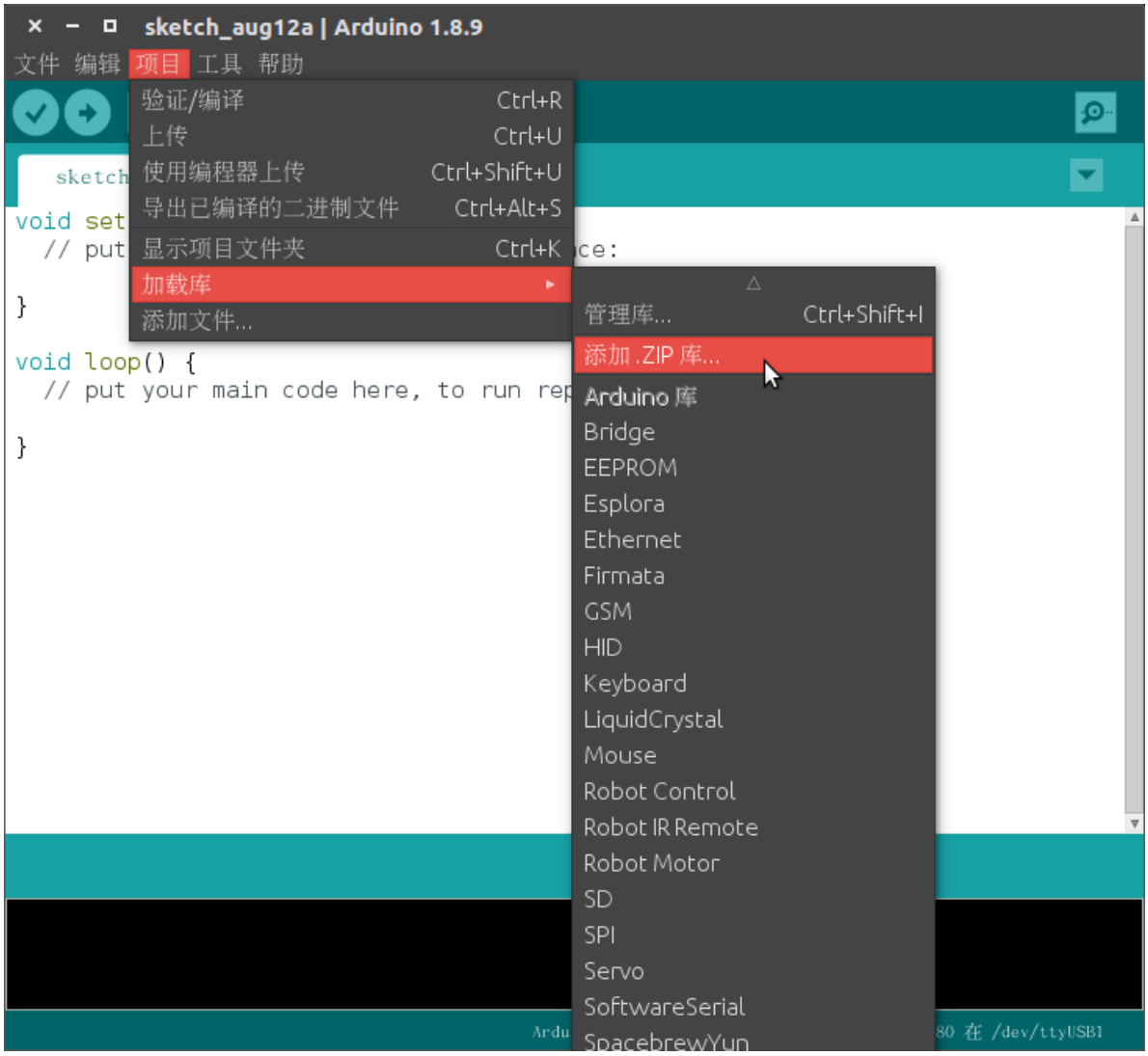
11.1.2 方法一通过烧录 HEX 文件进行升级

- 1. 下载MoonBot 主控遥控器固件(.hex 文件)
- 2. 下载Arduino Hex 烧录工具
- 3. 烧录 hex 固件
  - Windows

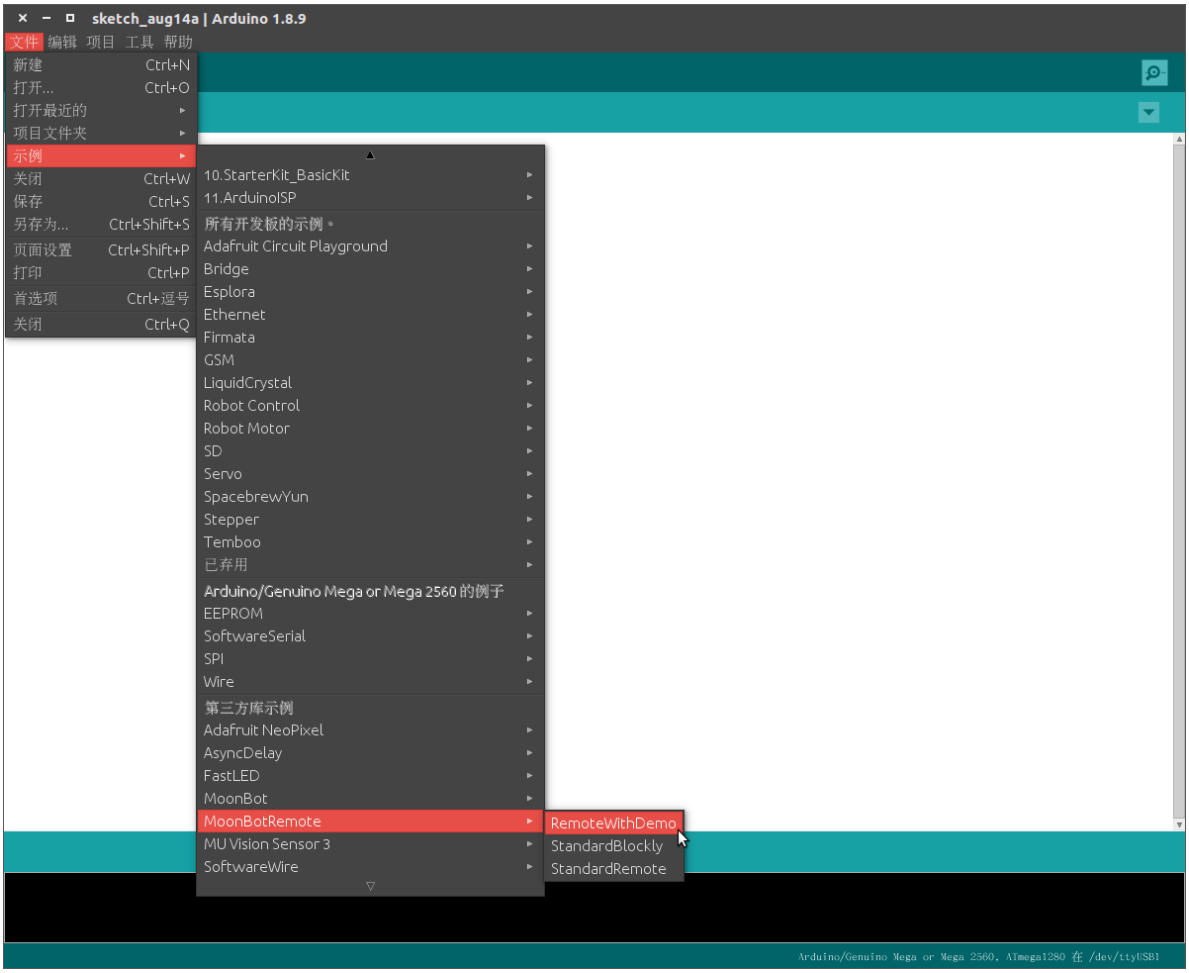
```
1）选择 MoonBot 对应的 COM 端口，硬件选择 `Arduino Mega (ATmega1280)`
2）选择加载 HEX 文件
3）点击下载，等待下载完成
```

11.1.3 方法二通过 Arduino IDE 编译 Arduino 源码进行升级

- 1. 搭建MoonBot Kit Arduino 开发环境
- 2. 下载MoonBot Kit 主控遥控器源码(Source.zip 文件)
- 3. 打开 Arduino IDE，点击项目-> 加载库-> 添加.ZIP 库，选择第 2 步下载的.zip 文件文件



- 4. 点击确定，完成 MoonBot Kit 主控遥控器源码的加载
- 5. 点击 Arduino 文件-> 示例->MoonBotRemote->RemoteWithDemo，打开源码



- 6. 连接 MoonBot Kit 主控至电脑，点击 Arduino 工具–> 端口，选择对应的 MoonBot 端口
- 7. 点击下载按钮，等待下载完成

11.1.4 测试

- 1. 开机重启后按下主控上按钮 A，靠近 A 键 LED 亮青色灯光并发出提示音
- 2. 开机重启后按下主控上按钮 B，靠近 B 键 LED 亮绿色灯光并发出提示音

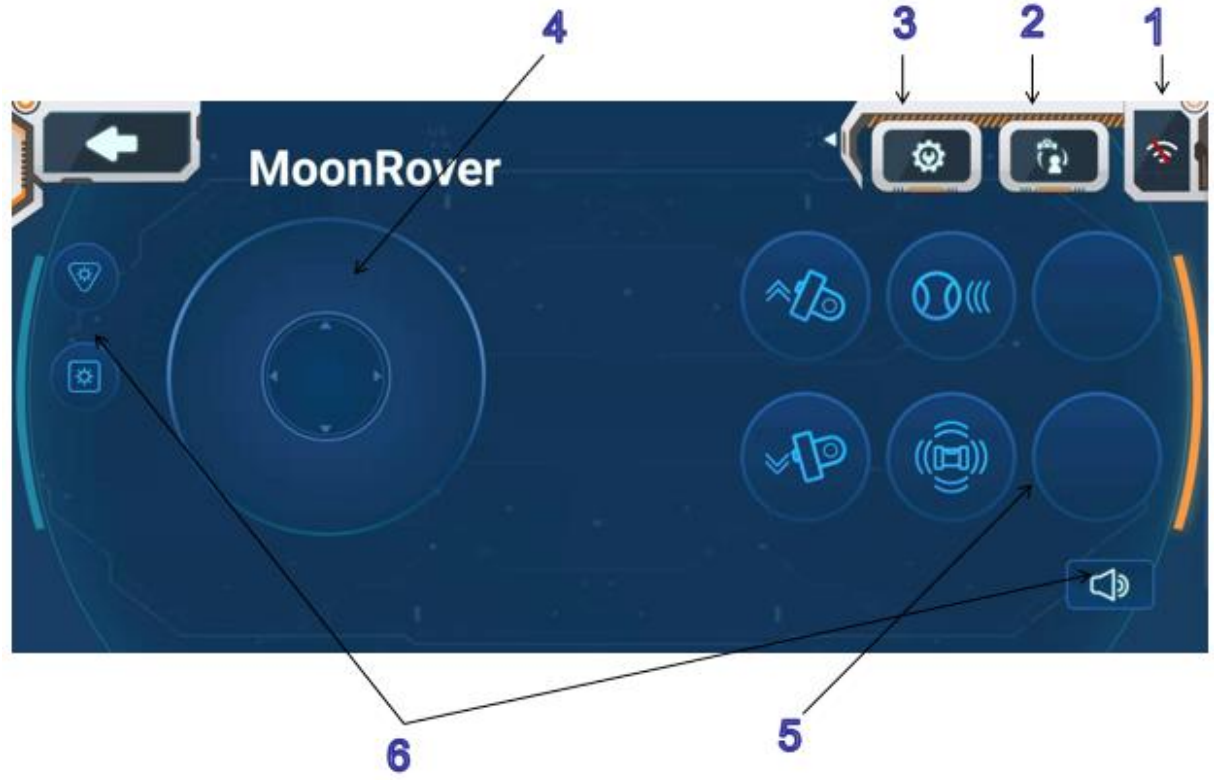
11.2 APP 控制器

MoonBot 的官方形态可以通过 Mu Bot app 直接控制，完成移动、视觉避障等基础功能。

APP 演示视频

11.2.1 控制器使用简介

首次进入 APP 控制器功能，选择对应搭建好的形态。



1.WiFi 连接

点选与视觉模块 LED 灯光颜色一致的颜色块组合开始连接。连接失败时，可按视觉模块复位键重新点选连接。

2. 形态选择

点击后在三种搭建形态中选择对应搭建好的形态。

3. 设置功能键

点击设置功能键，下方功能键圆形框呈虚线，点击虚线圆框可增加删除替换各个功能至圆形框内，点击设置功能键完成设置。

设置功能键 GIF 动图

4. 轮盘控制

连接好 wifi 后可通过轮盘控制机器人运动

5. 功能键

点击已设置的功能键使机器人做出相应的动作，某些功能可开启/关闭

车形态内置功能：视觉角度上、视觉角度下、跟踪球、避障开关、卡片识别

机械臂形态内置功能：主臂角度上/下、视觉角度上/下、机械爪开/闭、找球抓球、找篮板投篮

人形态内置功能：左臂角度上/下、右臂角度上/下、头角度上/下、舞蹈 1/2、卖萌

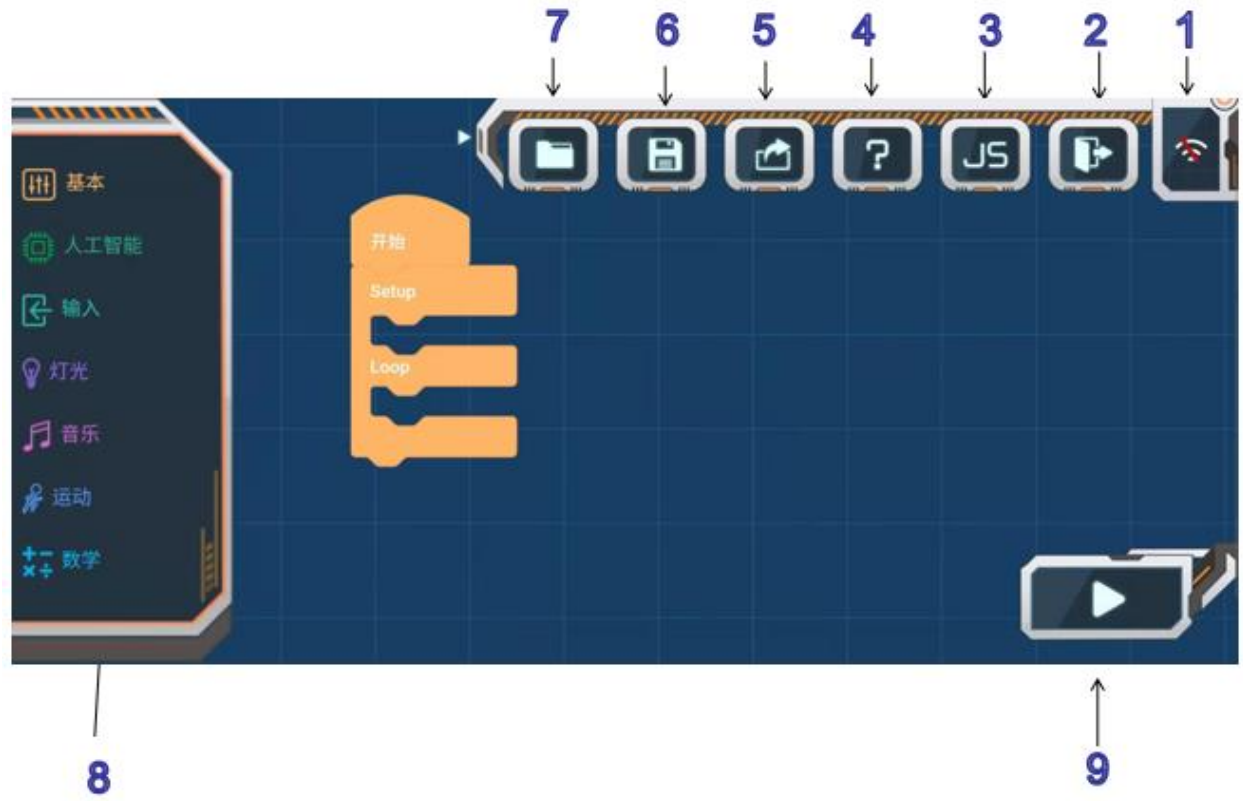
6. 灯光及声音键

点击按键使用会有灯光/声音。

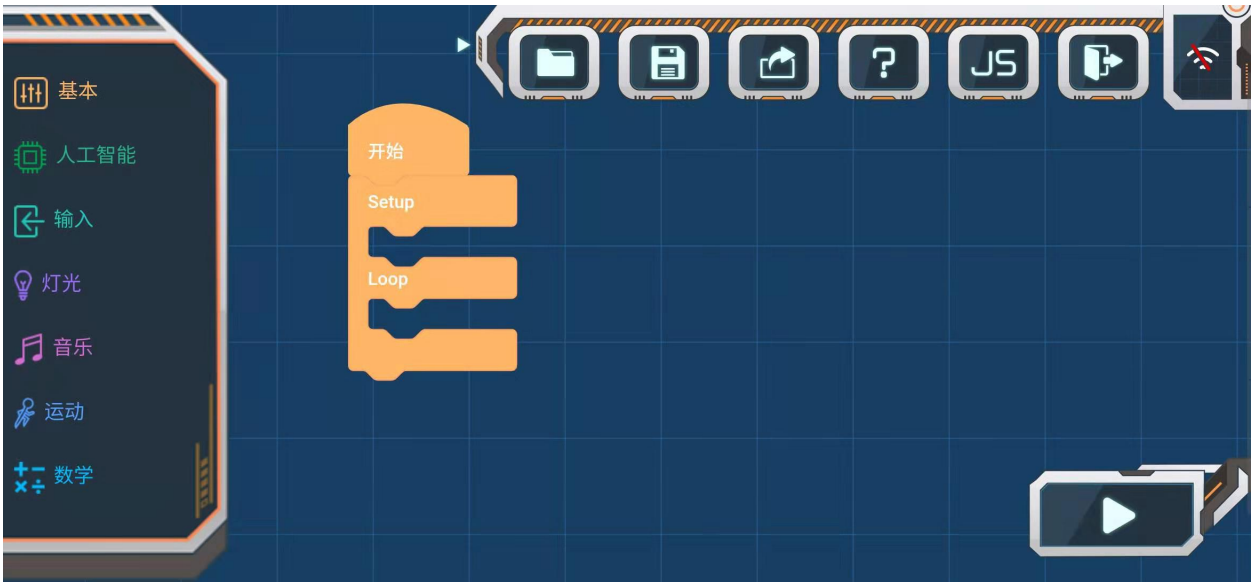
11.2.2 使用

控制器使用 GIF 动图

11.3 APP 自由编程



11.3.1 自由编程使用简介



1.Wifi 连接

点选与视觉模块 LED 灯光颜色一致的颜色块组合开始连接。连接失败时，可按视觉模块复位键重新点选连接。

2. 退出按钮

3. 此按键待更新功能

4. 帮助功能

点击会有各个按键说明

5. 分享功能

点击后可将程序分享至朋友/QQ/微信/微信朋友圈

6. 项目保存功能

点击后可将程序命名后保存至我的项目中。当程序已保存要更新时，可选择覆盖保存或另存为

7. 我的项目

可选择自己保存好的项目打开

8. 编程块

详见编程块描述，拖动至编程界面使用

9. 执行按钮

编写好程序后点击此按键执行传输

11.3.2 使用

编程 GIF 动图

11.4 APP 编程块 _ 人工智能

11.4.1 人工智能

算法开启关闭

算法：球体、人体、形状卡片、交通卡片、数字卡片、色块检测、颜色识别

参数：开启/关闭

使用说明

球体算法：识别橙色乒乓球 (Label：1) 和绿色网球（Label：2）

人体算法：检测人体的上半身特征

形状、交通、数字卡片：识别特定的卡片

色块检测：设定一种颜色，检测其色块区域

颜色识别：制定一块区域，检测其区域内的颜色

使用时可开启一种或同时开启多种算法

设置色块检测颜色

颜色：黑色、白色、红色、黄色、绿色、青色、蓝色、紫色

使用说明

色块检测算法默认检测红色，可通过此功能块改变检测的颜色

读取算法检测参数

算法：球体、人体、形状卡片、交通卡片、数字卡片、色块检测

设置参数：检测到、X 坐标、Y 坐标、宽度、高度、分类编号

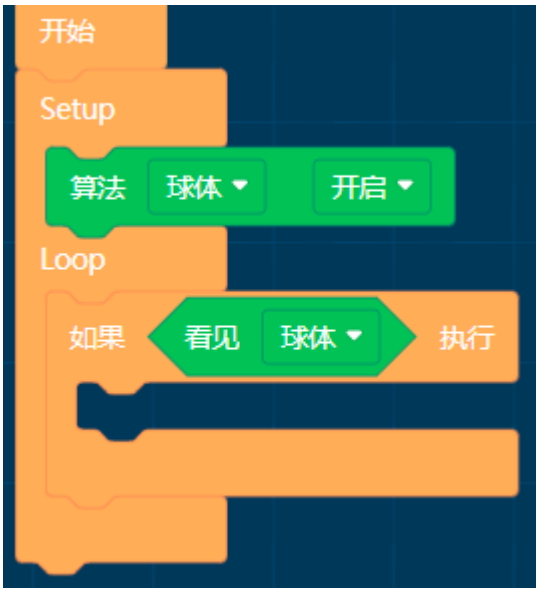
检测到：检测到时为真 True, 未检测到时为假 False

X 坐标、Y 坐标、宽度、高度: 量化到（0~100）

分类编号：Label 值

看见算法

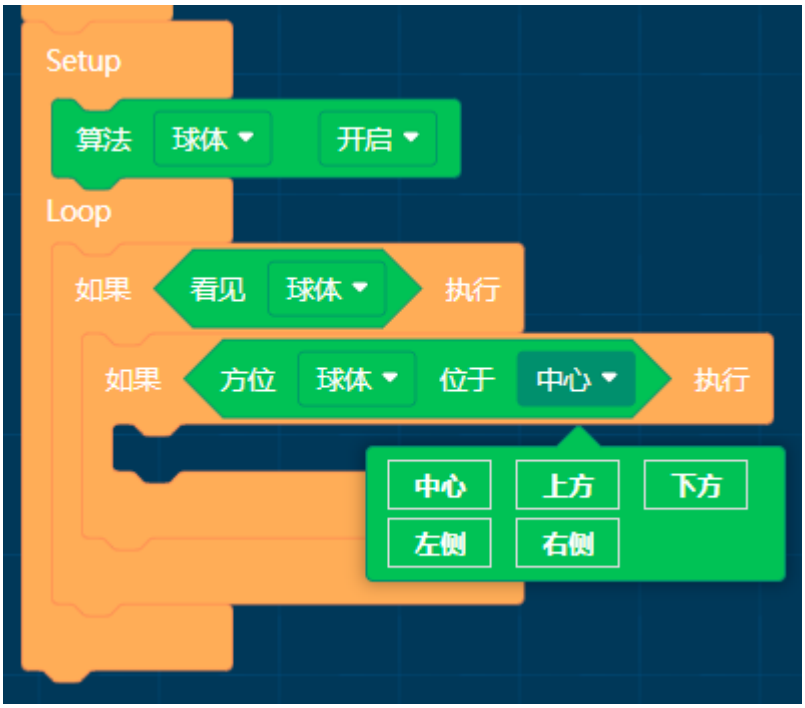
算法：球体、人体、形状卡片、交通卡片、数字卡片、色块检测、颜色识别



算法方位位置

算法：球体、人体、形状卡片、交通卡片、数字卡片、色块检测

参数：中心、上方、下方、左侧、右侧



算法尺寸大小

算法：球体、人体、形状卡片、交通卡片、数字卡片、色块检测

参数：大、中、小

算法区域位置

算法：球体、人体、形状卡片、交通卡片、数字卡片、色块检测

x：1/2/3/4/5

y：1/2/3/4/5

形状卡片

参数：对号、叉号、圆形、方形、三角形

交通卡片

参数：前进、左转、右转、掉头、停止

数字卡片

参数：0~9

球体识别

参数：乒乓球（橙色）、网球（绿色）

颜色识别

参数：黑色、白色、红色、黄色、绿色、青色、蓝色、紫色

检测到手势

参数：上划、下划、左划、右划、任意

有物体靠近小 MU

参数：任意、远、中、近

环境光亮度

参数：很暗、暗、良好、亮、很亮

摄像头缩放

参数：1/2/3/4/5

使用说明

Zoom 值小，则视野广，距离近。Zoom 值大，则视野窄，距离远。

白平衡模式

参数：自动、锁定、白光、黄光

使用说明

自动模式：适用于光照良好且对颜色要求不高的环境中使用；

锁定模式：适用于颜色要求较高的环境，让小 MU 面对白纸进行白平衡校准，然后锁定白平衡参数，锁定后颜色不会随环境的改变而变化；

白光模式：适合于白色灯光或者阴天环境中使用，该模式也属于自动白平衡模式；

黄光模式：适合于黄色灯光或者阳光环境中使用，该模式也属于自动白平衡模式。

小 MU 检测时灯光设置

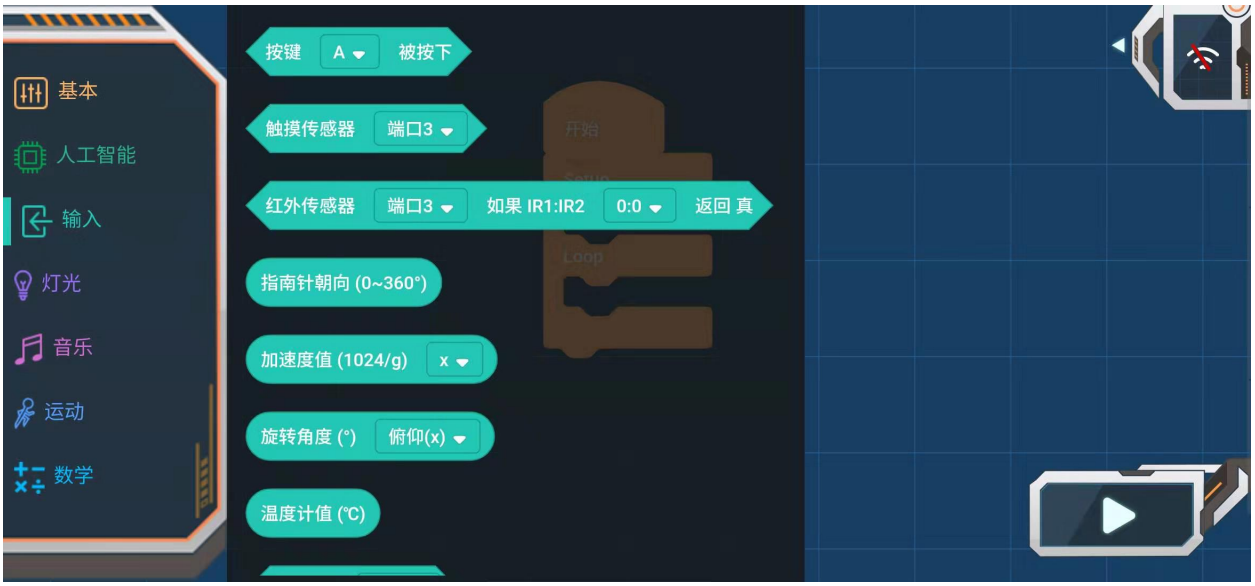
检测到时 LED 灯颜色：关闭、蓝色、绿色、青色、红色、紫色、黄色、白色、随机

未检测到时 LED 灯颜色：同上

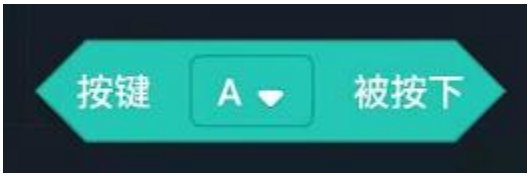
灯光亮度：1~10，数值越大越亮

11.5 APP 编程块 _ 输入

11.5.1 输入



读取按键被按下状态



参数：A、B、A+B

返回：按键被按下/未被按下

初始化触摸传感器端口

参数：端口 3、端口 5、端口 7、端口 8

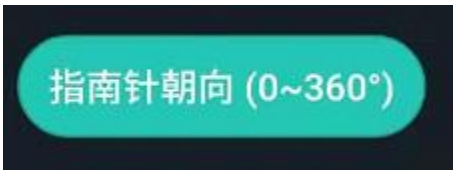
初始化红外传感器端口，读取两路红外传感器

端口参数：端口 3、端口 5、端口 7、端口 8

红外传感器 IR1:IR2 参数：0:0、0:1、1:0、1:1

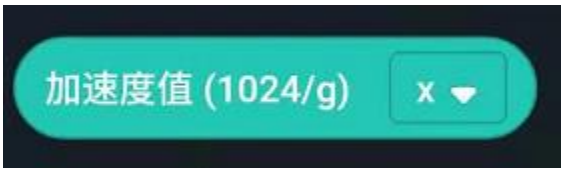
0 表示未检测到，1 表示检测到

读取指南针朝向 (0~360°)



返回：指南针朝向角度

读取加速度值（1024/g）



参数：x 方向、y 方向、z 方向、强度值

返回：加速度值

读取旋转角度（°）

参数：俯仰（x）、横滚（y），读取主控倾斜角

返回：角度值（-180°~+180°）

读取温度计值

返回：温度值

读取 IMU 动作

参数：震动、自由落体、X 轴向上、X 轴向下、Y 轴向上、Y 轴向下、Z 轴向上、Z 轴向下、3g、6g、8g

校准指南针

指南针校准模块，校准时主控需以”∞“字形翻转

11.6 APP 编程块 _ 灯光

11.6.1 灯光

眼睛灯光设置块

眼睛参数：所有、左眼、右眼

颜色参数：关闭、蓝色、绿色、青色、红色、紫色、黄色、白色、随机

亮度参数：1~10，数值越大越亮

表情块

参数：开心、悲伤、生气、眨眼、转圈、闪烁、彩虹、闭眼

返回: 显示表情

眼睛预设各 LED 灯 RGB

LED 灯：1~12 及所有灯

红色：0~255

绿色：0~255

蓝色：0~255

显示眼睛预设值

主控 LED 灯设置块

主控 LED 参数：所有、1、2

颜色参数：关闭、蓝色、绿色、青色、红色、紫色、黄色、白色、随机

亮度参数：1~10，数值越大越亮

小 MU LED 灯设置块

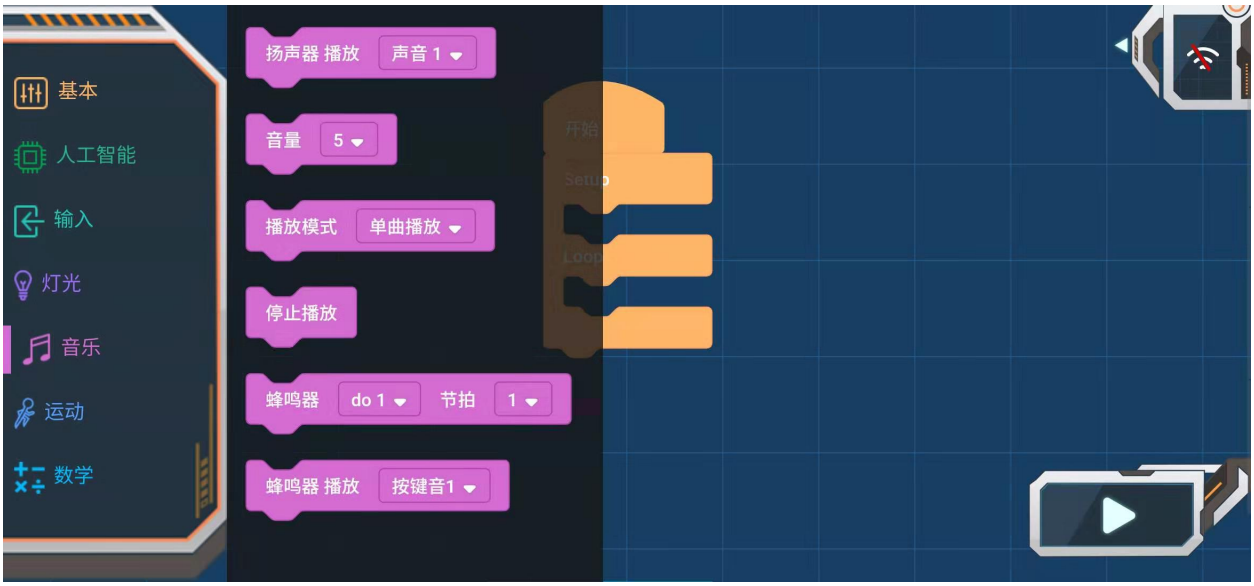
小 MU LED 灯参数：所有、1、2

颜色参数：关闭、蓝色、绿色、青色、红色、紫色、黄色、白色、随机

亮度参数：1~10，数值越大越亮

11.7 APP 编程块 _ 音乐

11.7.1 音乐



扬声器播放声音

播放指定的声音

可选声音：动物声、问候语、钢琴、城市、鼓、自定义声音

音量选择块

参数:0~10, 数值越大音量越大

播放模式

参数：单曲播放、单曲循环

停止播放

停止播放声音

蜂鸣器播放块

蜂鸣器以设置好的节拍播放音阶

音阶：空拍、do1-si7、Do1-Si7

节拍：1/8~4 拍

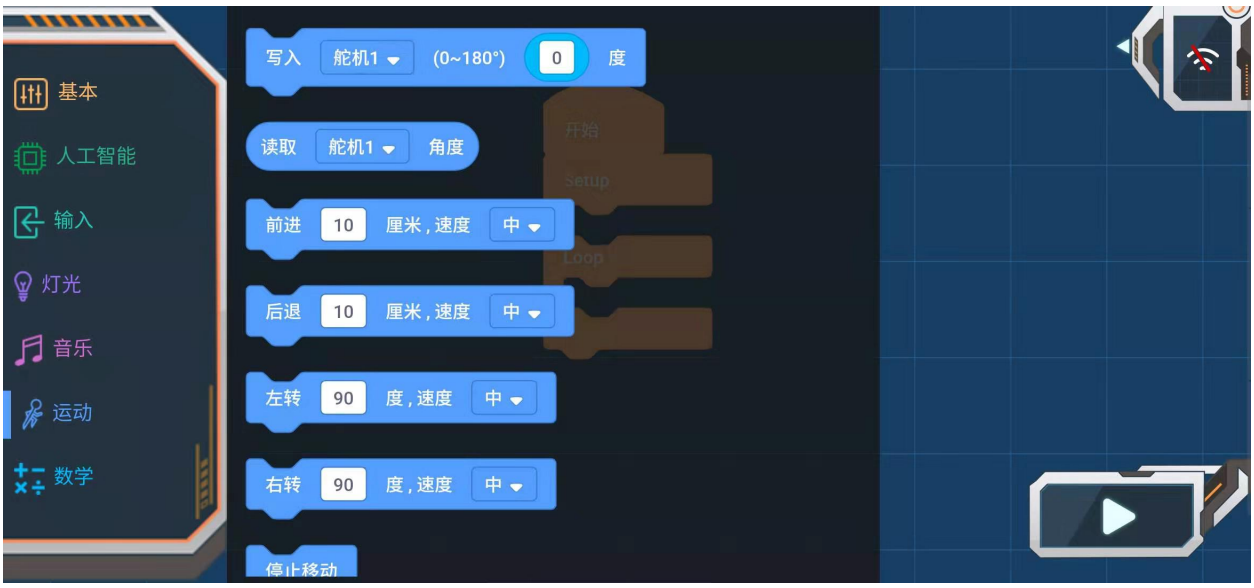
蜂鸣器播放块

播放指定的声音

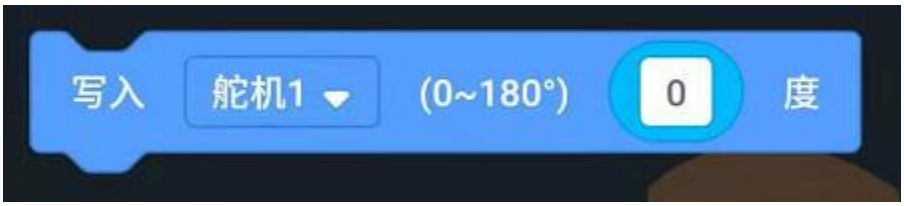
参数：按键音 1-4、警报 1-2、音效 1-4、救护声、警笛声

11.8 APP 编程块 _ 运动

11.8.1 运动



设置舵机角度



舵机口：舵机 1~ 舵机 4

角度：0~180°

读取舵机角度

读取指定舵机角度

参数：舵机 1~4

前进

以指定档位速度前进设置好的距离

执行距离：0~999cm

速度参数：很快、快、中、慢、很慢

后退

以指定档位速度后退设置好的距离

执行距离：0~999cm

速度参数：很快、快、中、慢、很慢

左转

以指定档位速度左转设置好的角度

执行距离：0~999°

速度参数：很快、快、中、慢、很慢

右转

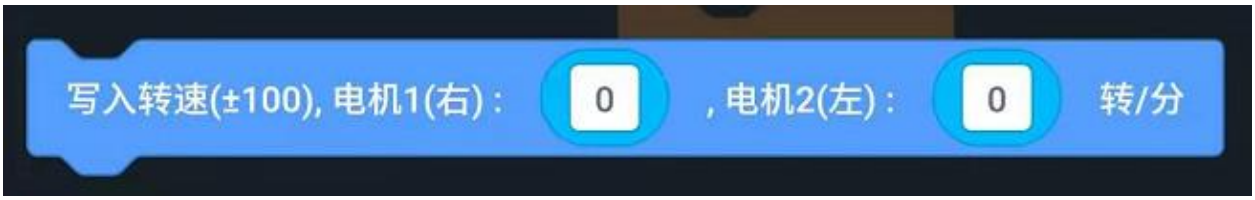
以指定档位速度右转设置好的角度

执行距离：0~999°

速度参数：很快、快、中、慢、很慢

停止运动

电机写入转速



向电机写入一定的转速（-100~+100 转/分）

参数：电机 1、电机 2

读取电机速度

参数：电机 1、电机 2

校准直行偏移

通过校准直行偏移使之不会往某一方向偏移

参数：0~200，>100 向右校准，<100 向左校准

校准直行距离

校准外部干扰引起的直行距离不准确的情况

参数：>100 增大距离，<100 减少距离

校准转弯角度

校准外部干扰引起的转弯角度不到位的情况

参数：>100 增大转弯角度，<100 减少转弯角度

校准舵机角度

通过舵机校准模块校准生产安装上的角度误差

参数：舵机 1~4

增加/减少（-90~+90°）

11.9 APP 示例程序

11.9.1 触摸招手

MoonBot 机器人手部装有舵机，头部装有触摸传感器，可通过编程实现触摸招手

简介：循环检测触摸传感器状态，当头部左侧被触摸时，机器人招左手。当头部右侧被触摸时，机器人招右手。

11.9.2 简单算法

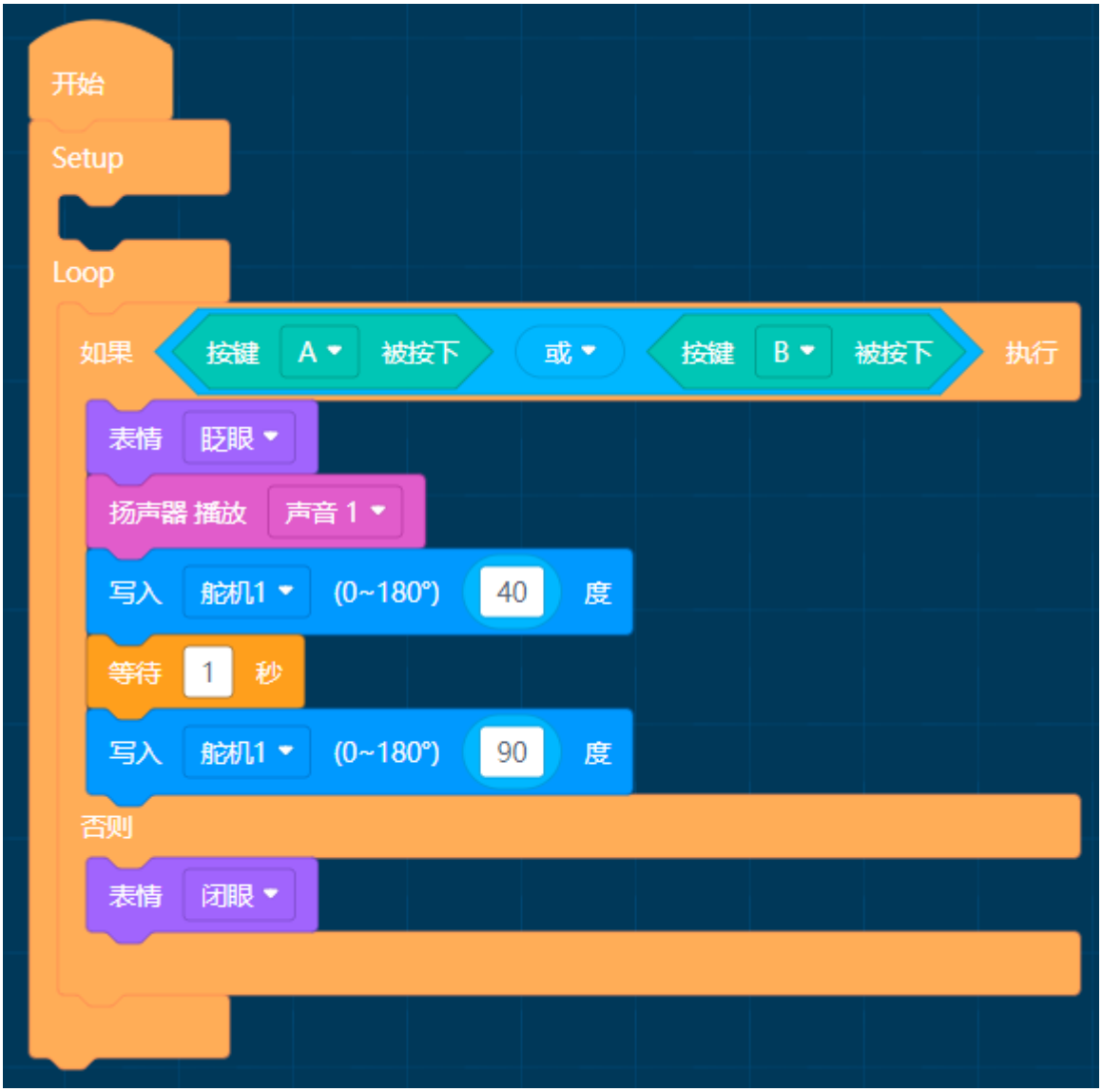
MoonBot 机器人使用视觉传感器与眼睛灯模块

简介：循环检测球算法，当检测到球时眼睛转圈表情，未检测到时闭眼。

11.9.3 功能模块示例

MoonBot 机器人使用按键扬声器 LED 灯舵机灯配合数学模块。

简介：循环检测按键 A/B 的状态，当按键被按下时，MoonBot 机器人做出声音灯光手臂动作。

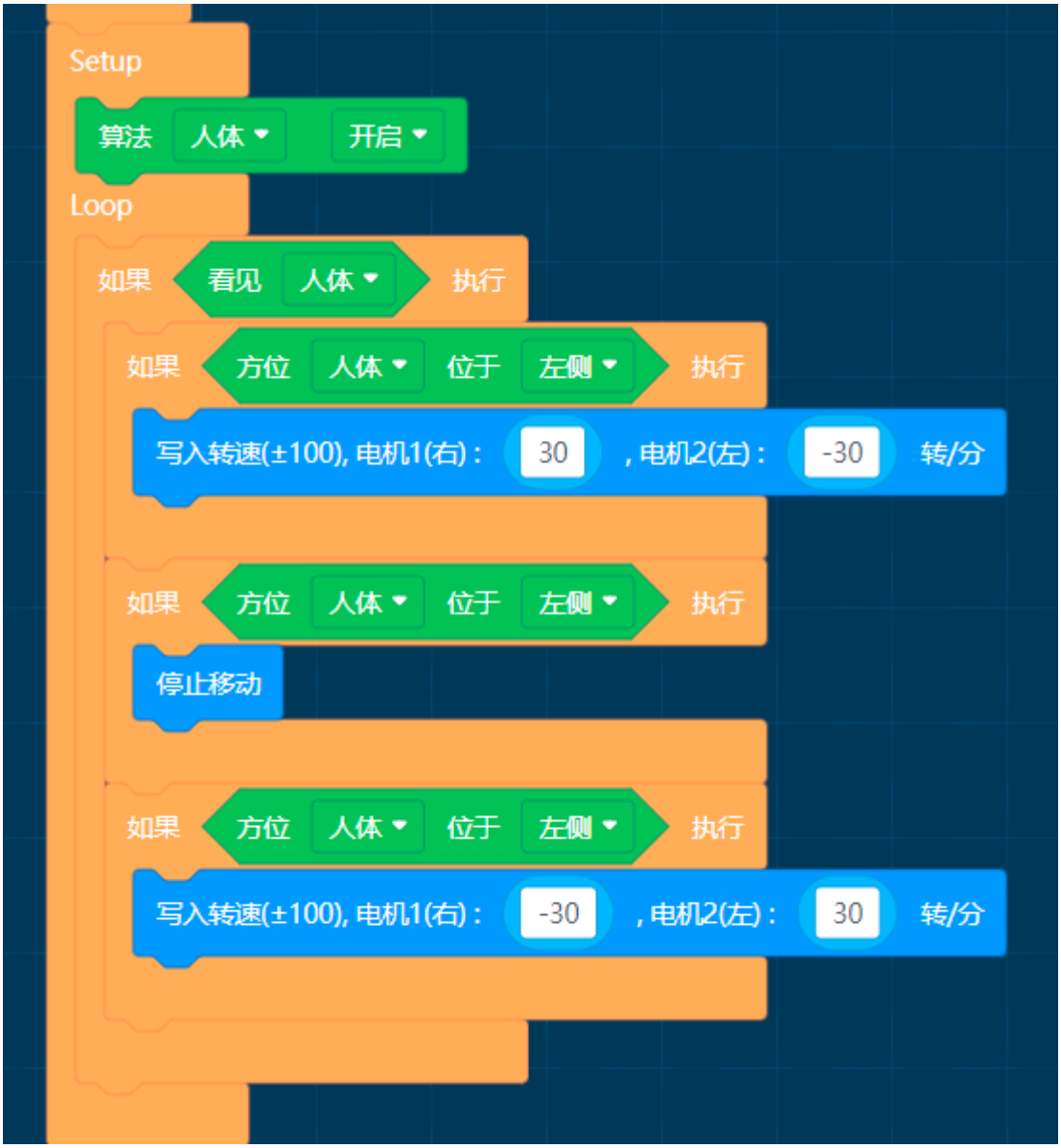


11.9.4 找人示例

MoonBot 机器人使用视觉传感器与运动模块

简介：开启人体算法，未检测到人体时视觉传感器 LED 闪烁红灯，检测到时亮蓝灯。

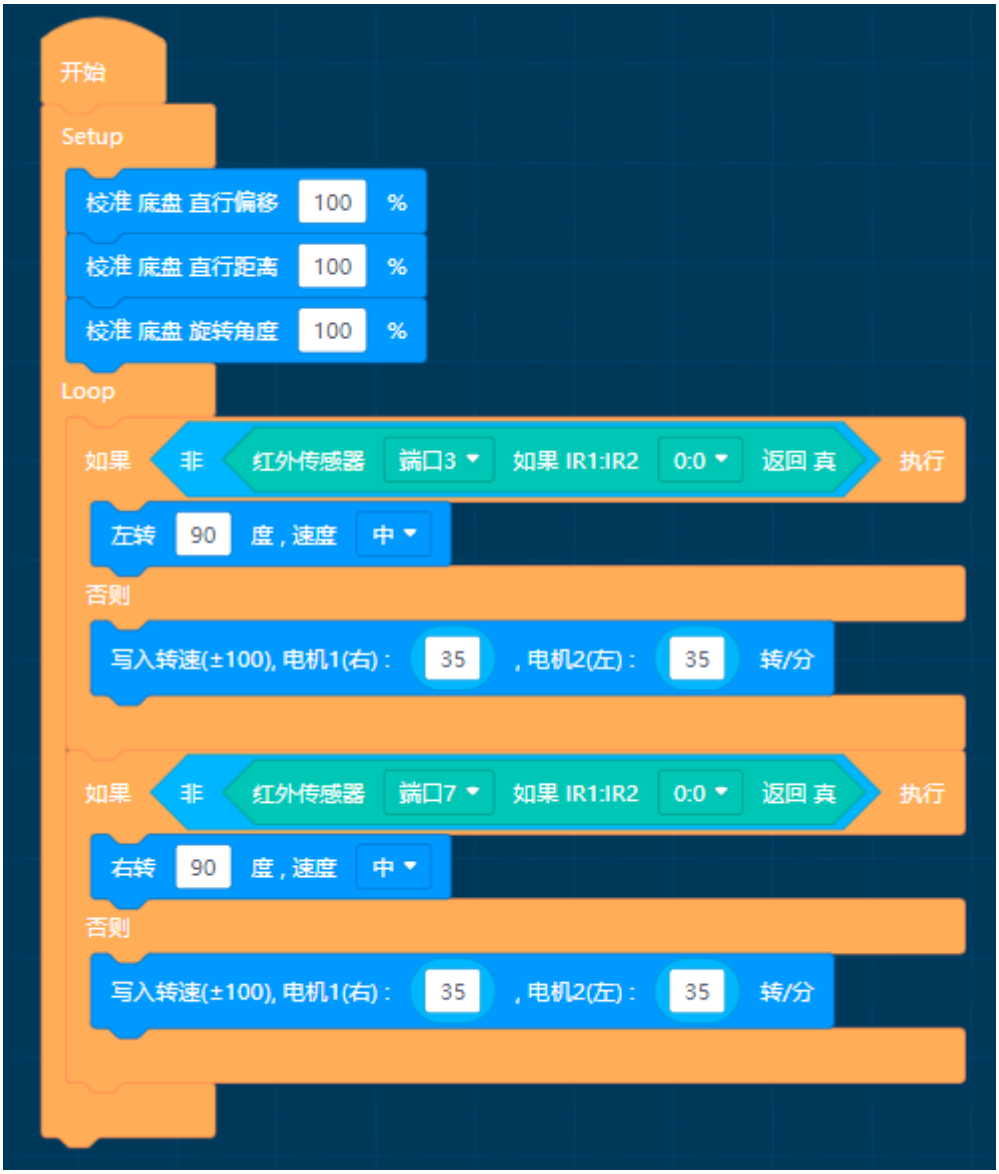
当检测到人体位于中心时机器人停止运动，否则向左/右转动。



11.9.5 避障智能车

在智能车左右两边装上红外传感器

简介：校准底盘，当智能车右侧红外传感器检测到障碍时左转，当左侧红外传感器检测到障碍时右转，均未检测到时直行



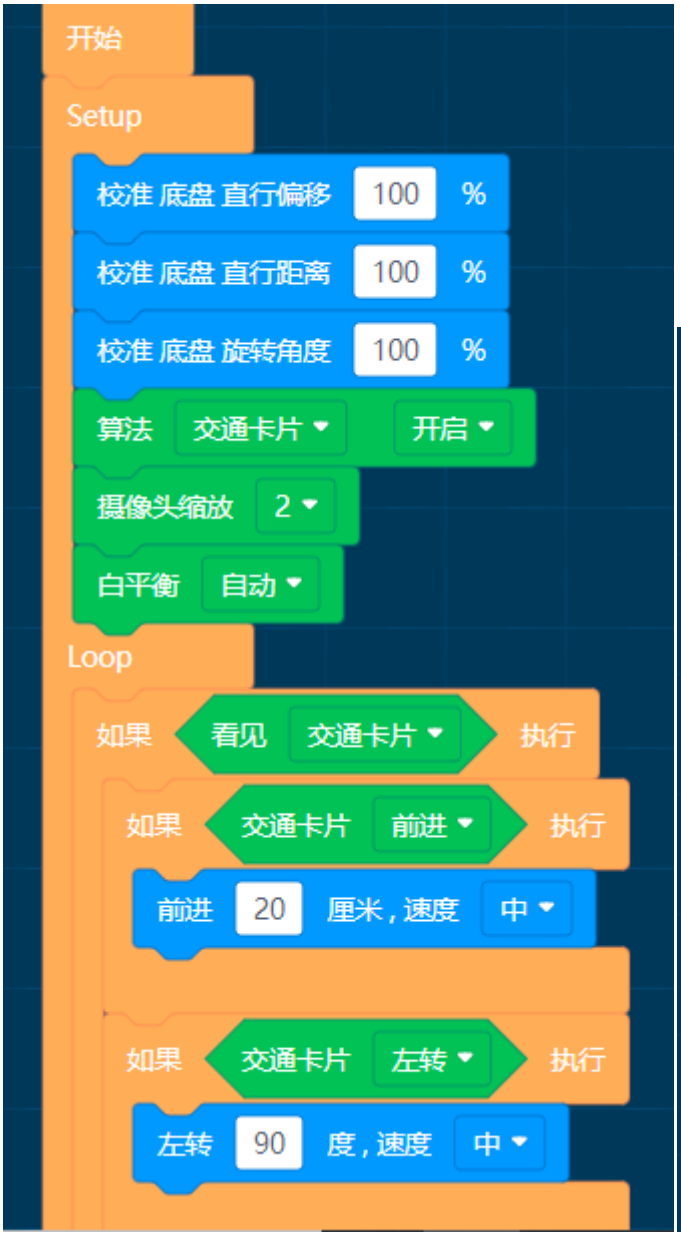
11.9.6 交通智能车

智能车配合交通卡片算法

简介：校准底盘，开启交通卡片算法，设置摄像头缩放等级，根据灯光设置白平衡参数。

未检测到交通卡片时，视觉传感器闪烁红灯，检测到卡片时亮蓝灯。

将卡片放在智能车前，识别不同交通卡片，智能车会做出不同的运动。


开始
Setup
校准 底盘 直行偏移 100 %
校准 底盘 直行距离 100 %
校准 底盘 旋转角度 100 %
算法 交通卡片 开启
摄像头缩放 2
白平衡 自动
Loop
如果 看见 交通卡片 执行
如果 交通卡片 前进 执行
前进 20 厘米，速度 中
如果 交通卡片 左转 执行
左转 90 度，速度 中


左转 90 度，速度 中
如果 交通卡片 右转 执行
右转 90 度，速度 中
如果 交通卡片 掉头 执行
左转 180 度，速度 中
如果 交通卡片 停止 执行
停止移动

CHAPTER 12

MoonBot Kit Mixly 教程

本文介绍 MoonBot Kit 使用米思奇（Mixly）进行开发的教程。

Mixly 基础教程请参考官方帮助文档 Mixly 帮助文档

12.1 MoonBot Mixly 编程搭建指南

12.1.1 完整安装包下载

对于未使用过 Mixly 的用户，可以下载完整的定制 Mixly 安装包，已包含 MU Vision Sensor 3 和 MoonBot Kit 的库。

Windows/Linux/Mac 完整版 MoonBot Mixly 安装包下载地址：https://pan.baidu.com/s/1h8Cuj8UYm99Mh3O1ppmMfg，提取码：`ksme`

12.1.2 独立库安装

导入、升级 Mixly-Arduino 库

对于已安装 Mixly 的用户，可通过导入 MU 的库来支持编程。此方法同样适用于其他第三方库的导入。

- 1. 启动 Mixly 下 Arduino
  - Windows

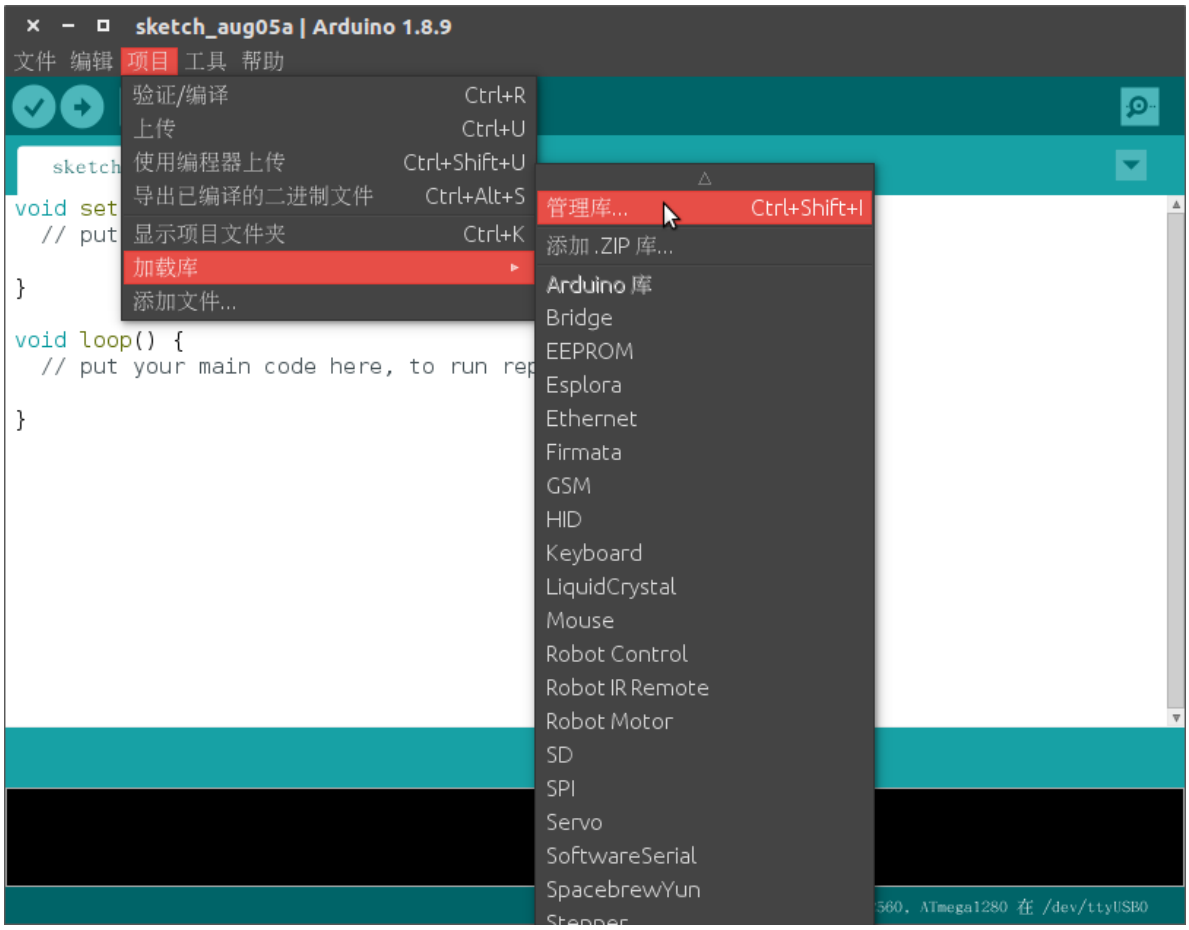
打开 Mixly 安装路径下 `{your_mixly_path}/arduino-1.8.5/arduino.exe` 文件，启动 Arduino

– Linux

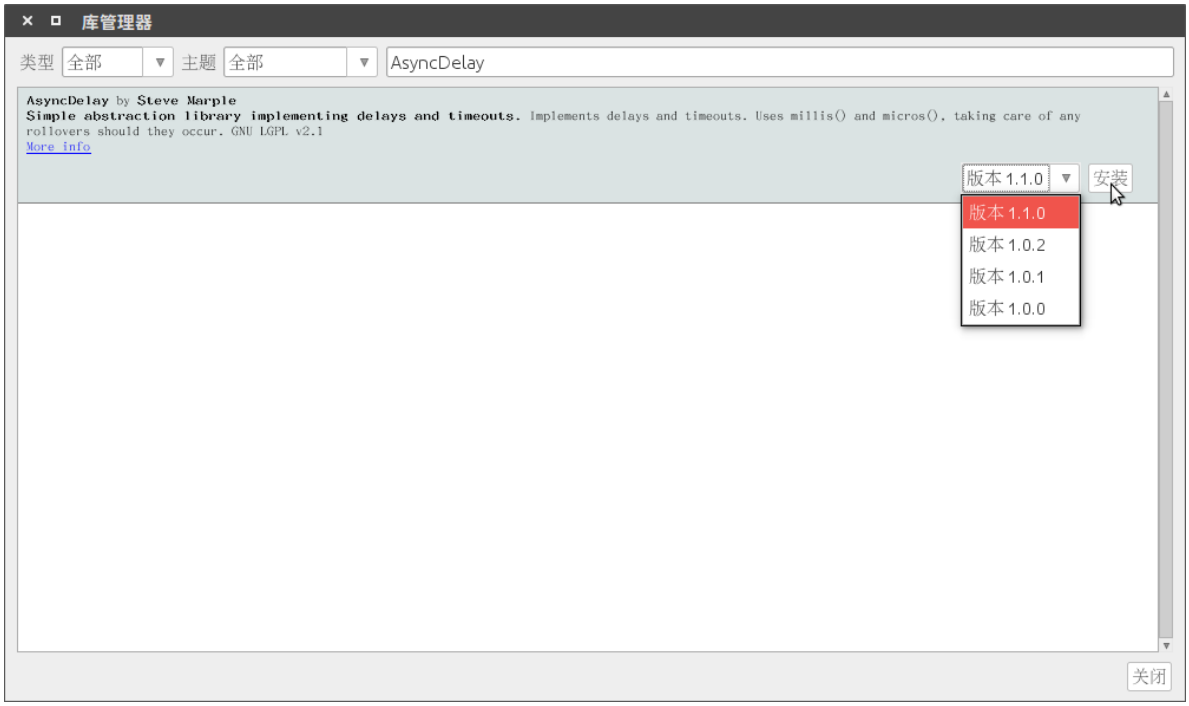
在终端运行路径 Mixly 安装路径下 Arduino 文件，启动 Arduino

```
$ cd {your_mixly_path}
$ ./arduino-1.8.2-linux64/arduino
```

- 2. 点击项目–> 加载库–> 管理库，打开库管理器



- 3. 搜索库 `AsyncDelay`，若没有安装则安装相关库，若库有更新，则进行更新



- 4. 按照第三步的安装方法安装库 `SoftwareWire` `Adafruit_NeoPixel` `Servo`，保证相关库安装到最新版
- 5. 关闭 Arduino，完成基础库安装

导入 Mixly 库

- 1. 点击下载MoonBot/MuVisionSensor3最新版 Mixly 库压缩包
- 2. 解压已下载的 `MoonBot/MuVisionSensor3` 压缩包
- 3. 打开 Mixly 界面，点击导入按钮，分别在刚解压出的 `MoonBot/MuVisionSensor3` 文件夹路径下找到`.xml` 结尾的文件
- 4. 至此，库安装完成

12.2 API 参考

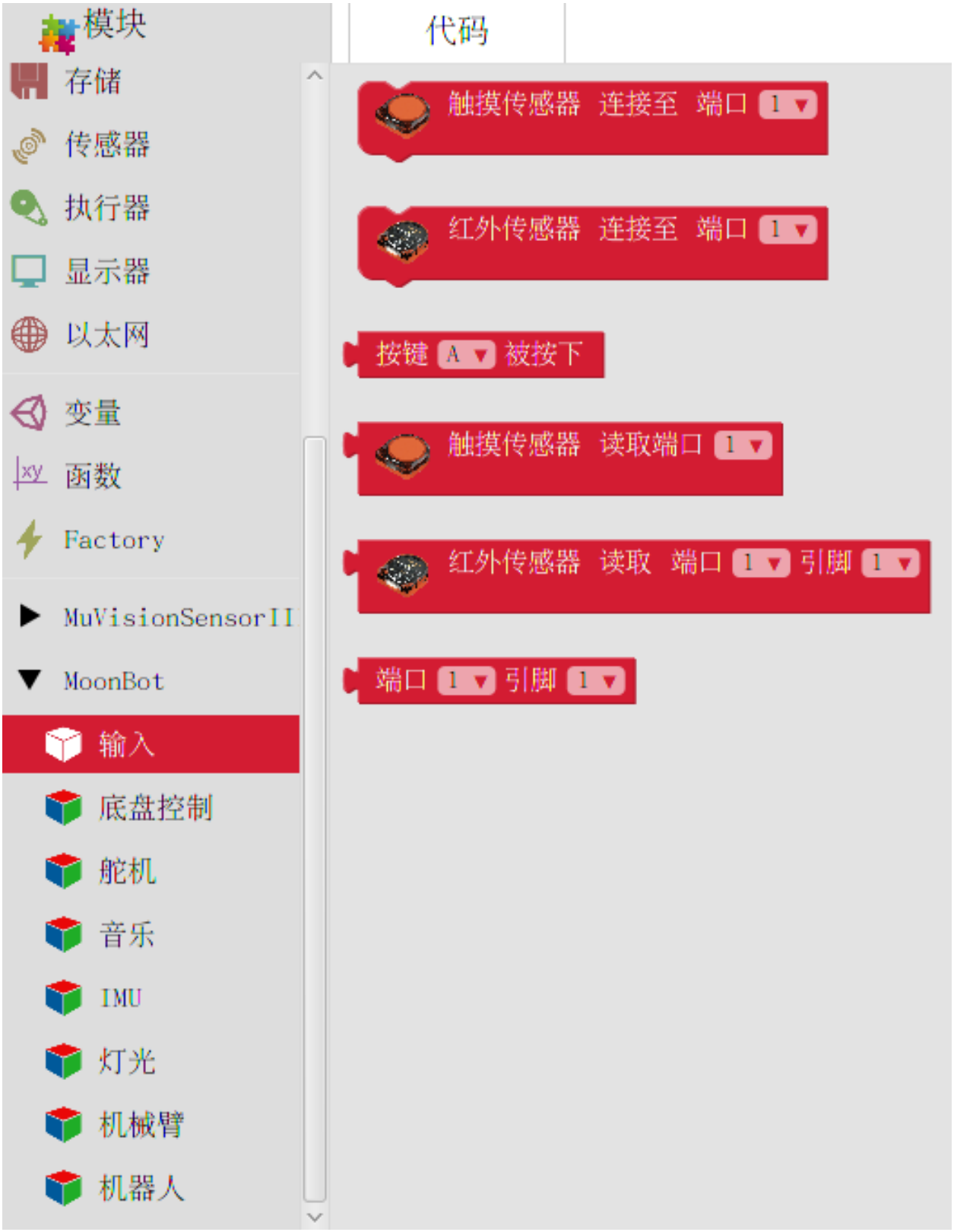
MoonBot Mixly 中定制了 MoonBot Kit 和 MU Vision Sensor 3 的编程块，本文将对程序块逐一说明，以及一些复杂的程序示例。可结合之前的硬件模块示例进行学习。

Mixly 基础教程见 Mixly 帮助文档，此处不再赘述。

MU Vision Sensor 3 教程见: MU Vision Sensor 3 Mixly 教程

12.2.1 输入

输入包含了 MoonBot Kit 触摸模块 红外模块 主控模块 的按键及引脚映射模块


模块
存储
传感器
执行器
显示器
以太网
变量
函数
Factory
MuVisionSensorII
MoonBot
输入
底盘控制
舵机
音乐
IMU
灯光
机械臂
机器人
代码
触摸传感器 连接至 端口 1
红外传感器 连接至 端口 1
按键 A 被按下
触摸传感器 读取端口 1
红外传感器 读取 端口 1 引脚 1
端口 1 引脚 1

触摸传感器初始化

描述 初始化触摸传感器至对应端口。

参数

端口

- 1~9

读取触摸传感器

描述 读取触摸传感器对应端口的值

参数

端口

- 1~9

返回

- HIGH： 有物体触摸到触摸传感器
- LOW： 无物体触摸到触摸传感器

红外传感器初始化

描述 初始化触摸传感器至对应端口

参数

端口

- 1~9

读取红外传感器

描述 读取红外传感器端口对应引脚的值

参数

端口

- 1~9

引脚

- 1~2

返回

- HIGH： 红外传感器被触发
- LOW： 红外传感器没有被触发

读取按键

描述 读取按键状态

参数

按键

- A： 按键 A
- B： 按键 B
- A&B： 按键 A 和 B

返回

- HIGH： 按键被按下
- LOW： 按键没被按下

端口引脚映射

描述 读取 MoonBot 端口对应的 Arduino 引脚号

参数

端口

- 1~9

引脚

- 1~2

返回

- 对应的 Arduino 引脚

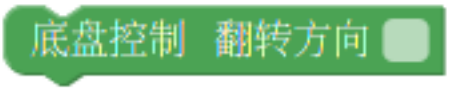
12.2.2 底盘控制

底盘控制包含了驱动 MoonBot Kit 电机模块 以及电机内编码器的驱动。

通过调用这些模块，可以让电机底盘运动起来。


模块
存储
传感器
执行器
显示器
以太网
变量
函数
Factory
MuVisionSensorII
MoonBot
输入
底盘控制
舵机
音乐
IMU
灯光
机械臂
机器人
代码
底盘控制 翻转方向
底盘控制 直行偏移校正(%) 100
底盘控制 直行距离校正(%) 100
底盘控制 转弯角度校正(%) 100
底盘控制 前进(cm) 0 转速(0~100RPM) 30
底盘控制 后退(cm) 0 转速(0~100RPM) 30
底盘控制 左转(°) 0 转速(0~100RPM) 30
底盘控制 右转(°) 0 转速(0~100RPM) 30
底盘控制 停止
电机 1 写入值(±255) 0

翻转方向



描述 翻转电机的运动方向。

参数

翻转方向

- `true`： 翻转方向
- `false`： 默认方向

直行偏移校正

描述 因为摩擦、安装偏差等扰动存在，底盘直行时会存在往某一方向偏移的情况。

通过 直行偏移校正模块，可以校正外部扰动引起的直行偏移。

参数

直行偏移校正（%）

- `0~200`： >100 向右侧校正，<100 向左侧校正

直行距离校正

描述 因为摩擦、安装偏差等扰动存在，底盘直行一定距离时会存在直行路程不到位的情况。

通过 直行距离校正模块，可以校正外部扰动引起的直行距离不到位的情况。

校正直行距离前，建议先进行 直行偏移校正。

参数

直行距离校正（%）

- `0~+∞`： >100 直行距离增大，<100 直行距离减小

转弯角度校正

描述 因为摩擦、安装偏差等扰动存在，底盘转动一定角度时会存在转弯角度不到位的情况。

通过 转弯角度校正模块，可以校正外部扰动引起的转弯角度不到位的情况。

校正转弯角度前，建议先进行 直行偏移校正和 直行距离校正。

参数

转弯角度校正（%）

- 0~+∞： >100 转弯角度增大，<100 转弯角度减小

前进

描述 控制底盘以给定速度向前直行至给定距离后停止。

该模块 **会调用编码器模块**，务必保证对应的编码器已连接至对应的端口。

参数

前进距离（cm）

- 距离值：给定直行距离，单位：厘米（cm）

转速

- 转速值：给定直行电机转速，单位：转/分（RPM）

后退

描述 控制底盘以给定速度向后直行至给定距离后停止。

该模块 **会调用编码器模块**，务必保证对应的编码器已连接至对应的端口。

参数

后退距离（cm）

- 距离值：给定直行距离，单位：厘米（cm）

转速

- 转速值：给定直行电机转速，单位：转/分（RPM）

左转

描述 控制底盘以给定速度左转至给定角度后停止。

该模块 **会调用编码器模块**，务必保证对应的编码器已连接至对应的端口。

参数

左转角度（°）

- 角度值：给定直行距离，单位：度（°）

转速

- 转速值：给定直行电机转速，单位：转/分（RPM）

右转

描述 控制底盘以给定速度右转至给定角度后停止。

该模块 **会调用编码器模块**，务必保证对应的编码器已连接至对应的端口。

参数

右转角度（°）

- 角度值：给定直行距离，单位：度（°）

转速

- 转速值：给定直行电机转速，单位：转/分（RPM）

停止

描述 底盘停止转动。

电机写入值

描述 向对应端口的电机写入模拟量。

参数

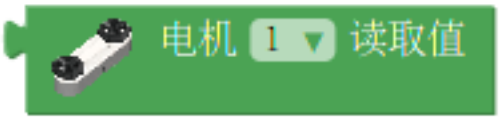
电机端口

- 1： 电机端口 1
- 2： 电机端口 2

值

- ±255： 写入模拟量的值，>0 向前转，<0 向后转，=0 停止转动

电机读取值



描述 读取对应电机端口模拟量的值。

参数

电机端口

- 1： 电机端口 1
- 2： 电机端口 2

返回

- 值: 电机模拟量的值

电机写入转速

描述 向对应端口的电机写入转速。

该模块 **会调用编码器模块**，务必保证对应的编码器已连接至对应的端口。

参数

电机端口

- `1`： 电机端口 1
- `2`： 电机端口 2

值

- `±60`： 写入模拟量的值，>0 向前转，<0 向后转，=0 停止转动，单位：转/分（RPM）

电机读取转速

描述 读取对应电机端口的转速。

参数

电机端口

- `1`： 电机端口 1
- `2`： 电机端口 2

返回

- 转速：电机的转速，单位：转/分（RPM）

12.2.3 舵机

舵机包含了 MoonBot Kit 舵机模块 驱动，可以用来驱动 MoonBot Kit 中 4 个舵机端口上所连接的舵机，来实现单个或多个口的舵机同时运动。

设置角度

描述 向给定舵机端口连接的舵机写入角度。

参数

舵机端口

- 1~4

角度

- 0~180°

读取角度

描述 读取给定舵机端口的当前角度值。

参数

舵机端口

- 1~4

预设角度

描述 预设给定舵机端口的舵机角度和舵机运行速度。

该模块需和 同步移动所有舵机至预设角度模块搭配使用。

参数

舵机端口

- 1~4

速度

- 快：设定舵机运行速度为快速（约 150°/s）
- 中：设定舵机运行速度为中速（约 100°/s）
- 慢：设定舵机运行速度为慢速（约 50°/s）

同步移动所有舵机至预设角度

描述 移动所有舵机至预设角度。

该模块需和 预设角度模块搭配使用。

翻转方向

描述 以 90° 为中间值，翻转舵机角度。

参数

- `false`： 默认运动方向
- `true`： 翻转舵机运动方向

校正

描述 因齿轮、舵机盘等生产、安装上会产生误差而导致舵机转不到给定角度。

通过舵机校准模块可以校正因以上原因而引起的角度误差。

参数

- `±90°`

12.2.4 音乐

音乐包含了 MoonBot Kit 主控模块 板载蜂鸣器的驱动和外部扬声器模块 的驱动。

通过调用这些模块可以让您控制 MoonBot Kit 播放动人的音乐。


模块
通信
存储
传感器
执行器
显示器
以太网
变量
函数
Factory
MuVisionSensorII
MoonBot
输入
底盘控制
舵机
音乐
IMU
灯光
机械臂
机器人
代码
扬声器 连接至 端口 2
扬声器 设置播放模式 单曲播放
扬声器 播放 大象
扬声器 播放 “ ”
扬声器 播放/暂停
扬声器 音量(0~32) 20
蜂鸣器 播放 中C' do' 1 拍
蜂鸣器 暂停 1/16 拍
蜂鸣器 设置节拍(BPM) 120
蜂鸣器 频率(Hz) 100 时间(ms) 0
蜂鸣器 停止播放

扬声器初始化

描述 初始化给定端口上连接的扬声器。

参数

端口

- 2,7,9

扬声器设置播放模式

描述 设置扬声器的播放模式。

参数

播放模式

- 单曲播放：播放指定音乐后停止播放
- 单曲魂环：循环播放指定音乐
- 播放所有：播放完指定音乐后自动播放音乐列表中下一首音乐
- 随机播放：播放完指定音乐后随机播放音乐列表中的一首音乐

扬声器播放音乐

描述 播放给定名称的音乐。

参数

音乐名

- ： 见模块对应的下拉菜单

扬声器播放自定义音乐

描述 播放给定音乐名称的音乐。

该操作前用户需往扬声器内存入对应的自定义音乐（如何存入音乐？），音乐名前四字需为字母或数字。

参数

音乐名

- ：自定义音乐名称的前 4 个字符，只支持 **英文**或 **数字**

扬声器播放设置

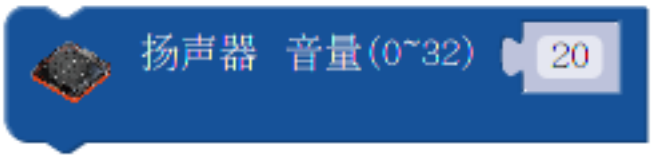
描述 设置当前扬声器播放状态。

参数

播放设置

- 播放/暂停：播放或暂停当前音乐
- 下一首：播放音乐列表中下一首音乐
- 上一首：播放音乐列表中上一首音乐
- 停止：停止播放音乐

扬声器设置音量



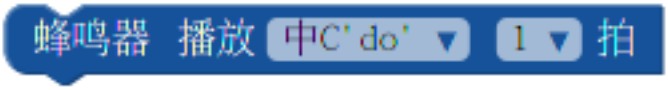
描述 设置扬声器音量。

参数

音量

- 0~32

蜂鸣器播放音阶



描述 蜂鸣器以给定音阶播放给定节拍的时间。

参数

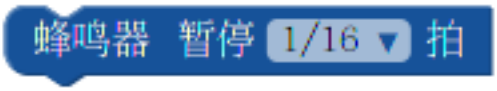
音阶

- 高 中 低三阶

节拍

- 1/16~4 拍：单节拍时间可通过 蜂鸣器设置播放节拍设置

蜂鸣器暂停播放



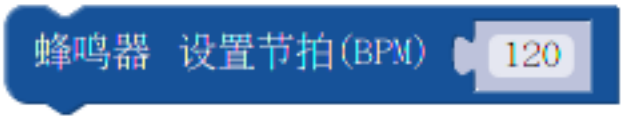
描述 蜂鸣器暂停播放给定节拍的时间。

参数

节拍

- 1/16~4 拍：单节拍时间可通过 蜂鸣器设置播放节拍设置

蜂鸣器设置播放节拍



描述 设置蜂鸣器每分钟节拍数（BPM）。

参数

每分钟节拍数

蜂鸣器播放频率

描述 设置蜂鸣器以给定频率播放给定时间的音乐。

参数

频率

- 0~65535： 频率推荐设置在人耳所能接受的频率范围（20 ~ 20000Hz）

时间

- 0： 持续播放
- 其他：播放指定长度的时间后停止

蜂鸣器停止播放

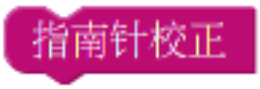
描述 蜂鸣器停止播放声音。

12.2.5 IMU

IMU 包含了 MoonBot Kit 主控模块 板载的三轴加速度，三轴磁力计和温度传感器的驱动。

通过调用这些模块，您可以获取 MoonBot Kit 主控当前方向、倾斜角度和状态等。

指南针校正



描述 校准指南针，校准时主控需以”∞“字形翻转。

具体可参考手机指南针校准方法。

获取指南针角度

描述 读取指南针 Y 轴当前方向与正北方向的夹角。

返回

- 0~360°

获取加速度值

描述 读取给定轴的加速度模拟量。

参数

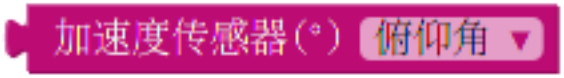
方向轴

- X,Y,Z

返回

- 加速度模拟量

获取加速度角度



描述 获取当前主控的倾斜角度

参数

夹角类型

- 俯仰角：主控坐标系 Y 轴与水平面的夹角，当主控向上倾斜时，俯仰角为正，反之为负
- 横滚角：主控坐标系 X 轴与水平面的夹角，当主控向右倾斜时，横滚角为正，反之为负

返回

- ±180°

读取温度

描述 读取当前温度值

参数

温度单位

- ℃： 返回值以摄氏度为单位
- ℉： 返回值以华氏度为单位

返回

- 温度值

读取当前状态

描述 读取当前主控状态。

参数

状态

- 震动：主控是否处于震动状态
- 自由落体：主控是否处于自由落体状态

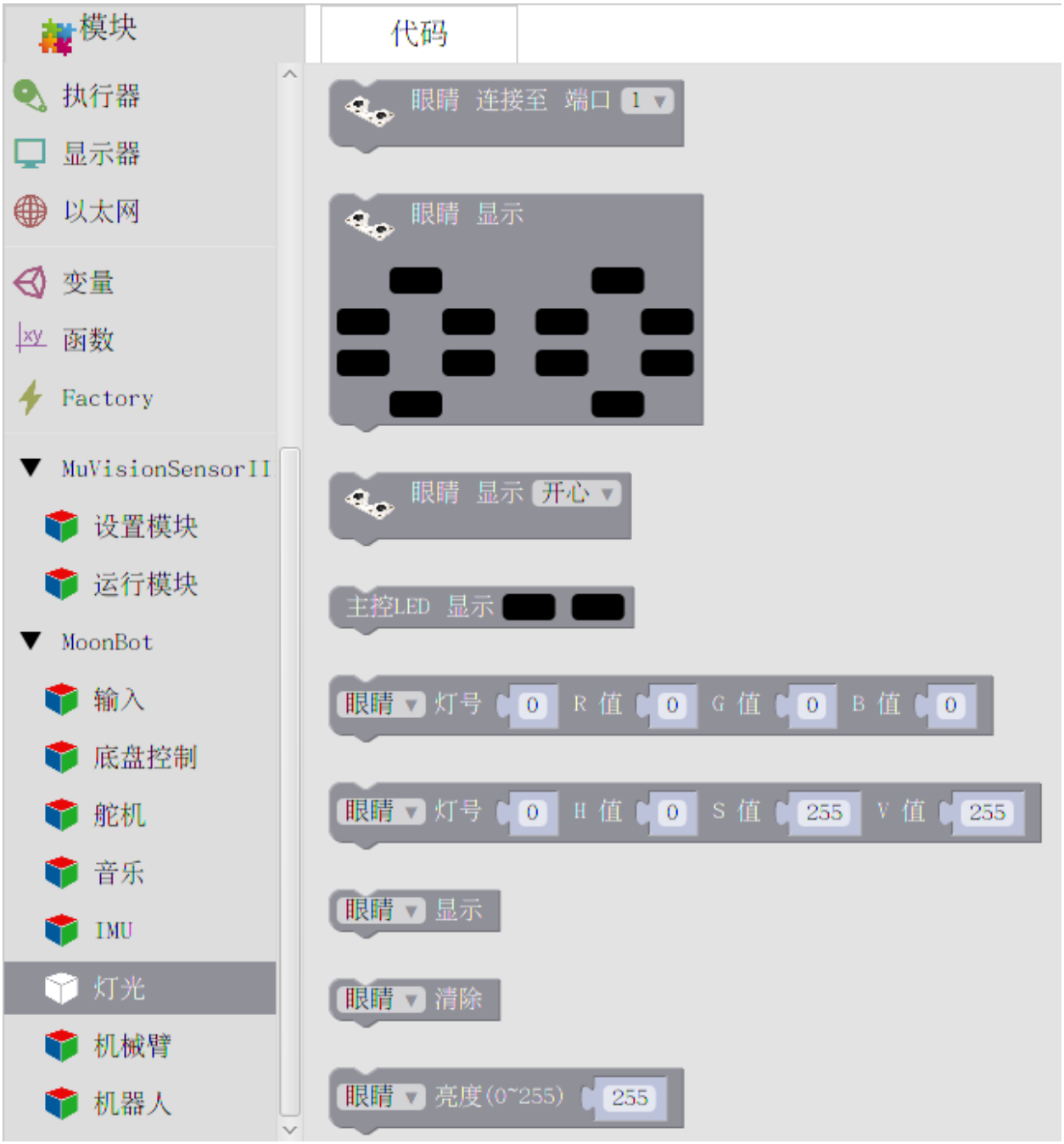
返回

- `true`： 主控当前处于该状态
- `false`： 触控当前不处于该状态

12.2.6 灯光

灯光模块包含了 MoonBot Kit 主控模块 板载的两颗 LED 和外接 12 颗 LED 眼睛模块 的驱动。

通过这些模块，您可以轻松地设置 LED 颜色和亮度。



眼睛初始化

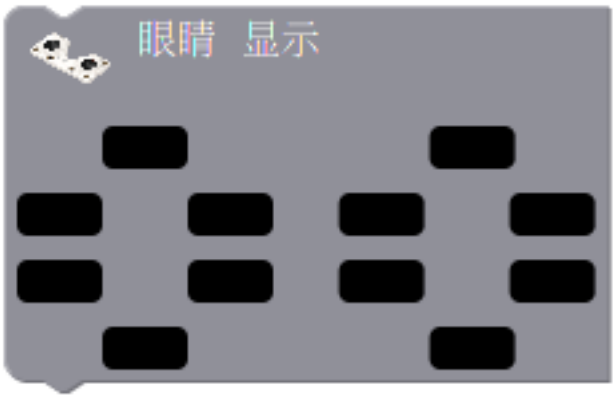
描述 初始化眼睛模块至指定端口。

参数

端口

- 1~9

眼睛显示



描述 将眼睛 LED 颜色数值写入缓存，并显示。

参数

颜色

- 

眼睛显示表情

描述 眼睛 LED 显示表情动作。

参数

表情

- 见模块下拉菜单

主控 LED 显示

描述 将主控 LED 颜色数值写入缓存，并显示。

参数

颜色

- 

LED 设置 RGB 值

描述 将 LED 给定灯号的 RGB 颜色数值写入缓存。

参数

LED 类型

- 眼睛：眼睛 LED
- 主控 LED：主控 LED

灯号

- 眼睛：0~11，主控 LED：0~1

R 值

- 0~255：红色通道模拟量

G 值

- 0~255：绿色通道模拟量

B 值

- 0~255：蓝色通道模拟量

LED 设置 HSV 值

描述 将 LED 给定灯号的 HSV 颜色数值写入缓存。

参数

LED 类型

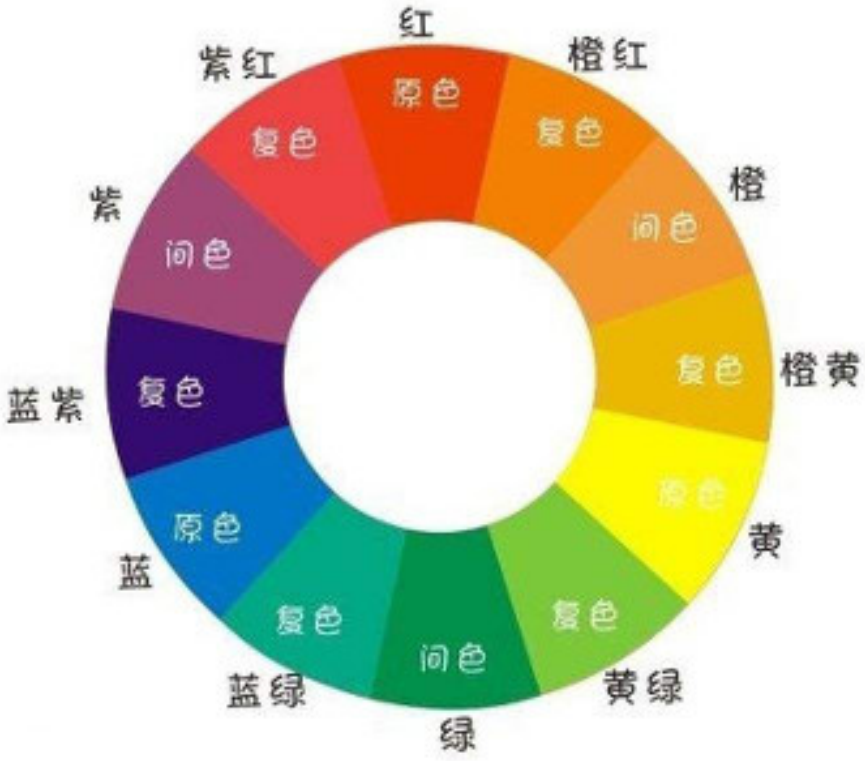
- 眼睛： 眼睛 LED
- 主控 LED： 主控 LED

灯号

- 眼睛: 0~11，主控 LED： 0~1

H 值

- 0~360°： 色调值



S 值

- 0~255： 饱和度值模拟量

V 值

- 0~255： 亮度值模拟量

LED 显示

描述 将给定 LED 缓存内的颜色值显示出来。

参数

LED 类型

- 眼睛： 眼睛 LED
- 主控 LED： 主控 LED

LED 清除

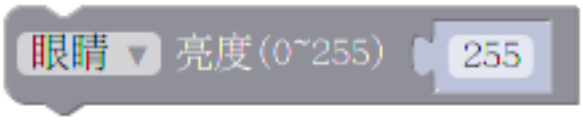
描述 清除指定 LED 的缓存。

参数

LED 类型

- 眼睛： 眼睛 LED
- 主控 LED： 主控 LED

LED 亮度



描述 设置给定 LED 的亮度。

参数

LED 类型

- 眼睛： 眼睛 LED
- 主控 LED： 主控 LED

亮度

- 0~255： 0 为最暗，255 为最亮

12.2.7 机械臂

机械臂包含了机械臂 形态下的集成动作。

通过调用这些模块，您可以轻松控制机械臂去抓球等。

初始化

描述 初始化 MoonMech 机械臂端口。

参数

MU 地址

- `MU00`： MU 地址 0x60
- `MU01`： MU 地址 0x61
- `MU10`： MU 地址 0x62
- `MU11`： MU 地址 0x63

MU 端口

- `2,7,9`

机械爪舵机端口

- `1~4`

上臂舵机端口

- `1~4`

下臂舵机端口

- `1~4`

设置抓球位置

描述 设置 MoonMech 机械臂抓球位置，可以通过调整识别球的 X Y 值来让机械爪抓到球。

当球处于给定的 X Y 值范围内时，机械爪会闭合抓取球。

参数

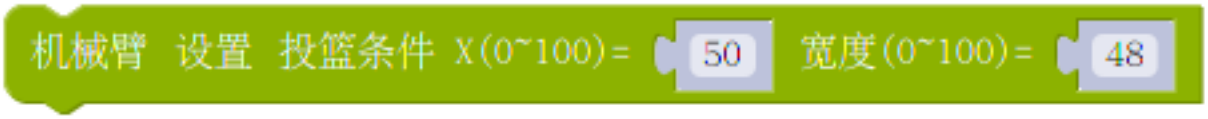
X

- 0~100：机械爪抓球时的水平位置，可通过修改此值调整机械爪抓球时相对于球的水平位置

Y

- 0~100：机械爪抓球时的垂直位置，可通过修改此值调整机械爪抓球时的垂直高度

设置投篮条件



描述 设置 MoonMech 机械爪投篮的条件，可以通过调整识别到卡片的水平位置 X 和宽度来让机械臂准确的投进篮筐。

当卡片处于给定的 X 宽度值范围内时，机械臂会触发投篮动作进行投篮。

参数

X

- 0~100：机械爪投篮时，相对于卡片横向坐标 X 的水平位置，可通过修改此值调整机械爪相对于卡片的水平位置

宽度

- 0~100：机械爪投篮时卡片的大小，可通过修改此值调 MoonMECH 机械臂投篮时与篮筐（卡片）的距离。

爪子动作

描述 设置机械爪动作。可通过此模块控制机械爪水平平移或上下平移。

参数

动作

- 张开：将机械爪张开（110°）
- 关闭：将机械爪关闭（90°）
- 前进：机械爪水平前进一个单位
- 后退：机械爪水平后退一个单位

- 向上：机械爪垂直向上一个单位
- 向下：机械爪垂直向下一个单位

找到球

描述 控制 MoonMech 机械臂执行找球动作。

返回

- `true`： 找到球
- `false`： 没有找打球

抓到球

描述 控制 MoonMech 机械臂执行抓球动作。

诺执行此块时没有找到球机械臂会原地不动，且返回 `false`。

返回

- `true`： 抓到球
- `false`： 没有找打球

找到卡片

描述 控制 MoonMech 机械臂执行搜索篮筐（卡片）动作。

参数

卡片类型

- 形状卡片
- 交通卡片
- 数字卡片

返回

- `true`： 找到给定卡片
- `false`： 没有找给定卡片

投篮

描述 控制 MoonMech 机械臂执行投篮动作。

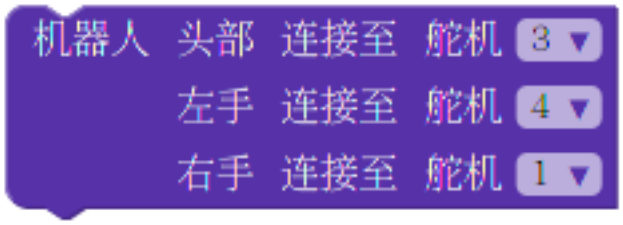
诺执行此块时没有找到卡片机械臂会原地不动。

12.2.8 机器人

机器人包含了*MoonBot 机器人* 中集成的动作。

通过调用这些模块，您可以控制机器人执行点头、招手等动作。

初始化



描述 初始化 MoonBot 机器人各个端口。

参数

头部舵机

- 1~4

左手舵机

- 1~4

右手舵机

- 1~4

招手

描述 驱动机器人的手臂进行招手。

参数

手臂

- 左手
- 右手
- 双手

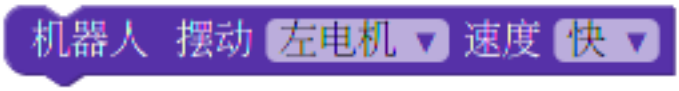
偏移

- 0~90： 机器人上下招手偏移角度

速度

- 快
- 中
- 慢

摆动



描述 同时摆动机器人的头和脚。

参数

电机

- 左电机
- 右电机
- 双电机

速度

- 快
- 中
- 慢

左右晃动身体

描述 控制机器人电机左右晃动。

参数

速度

- 快
- 中
- 慢

时间

- 0 ~ +∞： 电机单次晃动时间

前进迈步

描述 控制机器人向前迈出一步。

参数

速度

- 快
- 中
- 慢

时间

- 0 ~ +∞： 电机向前迈一步的时间，时间越长，步子越大

点头

描述 控制机器人点一次头。

参数

偏移

- 0~90°： 点头幅度

速度

- 快
- 中
- 慢

CHAPTER 13

MoonBot Kit Arduino 教程

本文介绍 MoonBot Kit 使用 Arduino IDE 进行开发的教程。

13.1 MoonBot Kit Arduino 开发环境搭建指南

MoonBot Kit（以下简称 MoonBot ）提供了 Arduino 库函数，支持在 Arduino(ATmega1280) 上进行开发编程。

本文档旨在指导用户基于 Arduino 官方 IDE 进行 MoonBot 硬件开发环境搭建。

13.1.1 准备工作

硬件：

- MoonBot 开发者套件
- PC (Windows、Linux 或 Mac OS)

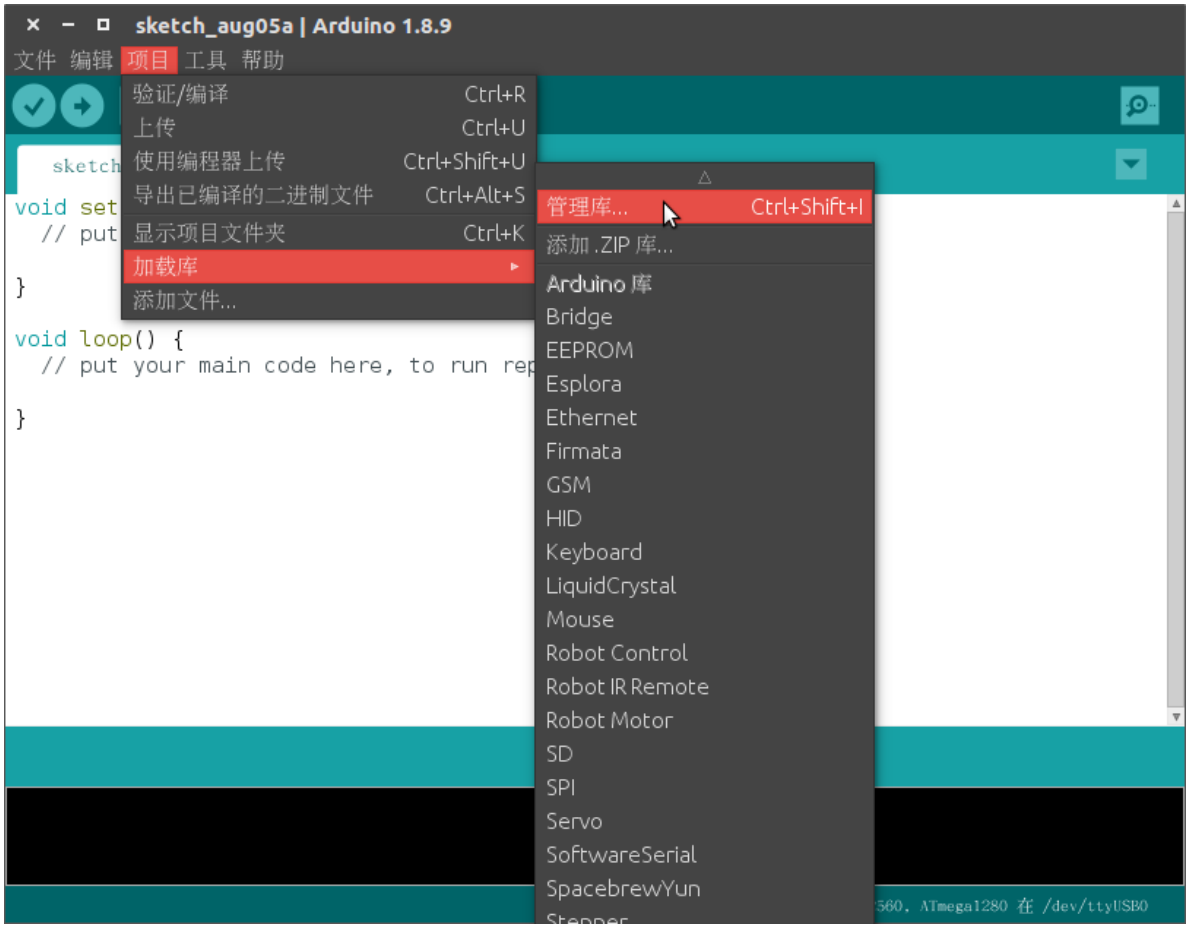
软件：

- Arduino 官方 IDE
- MoonBot Arduino 库

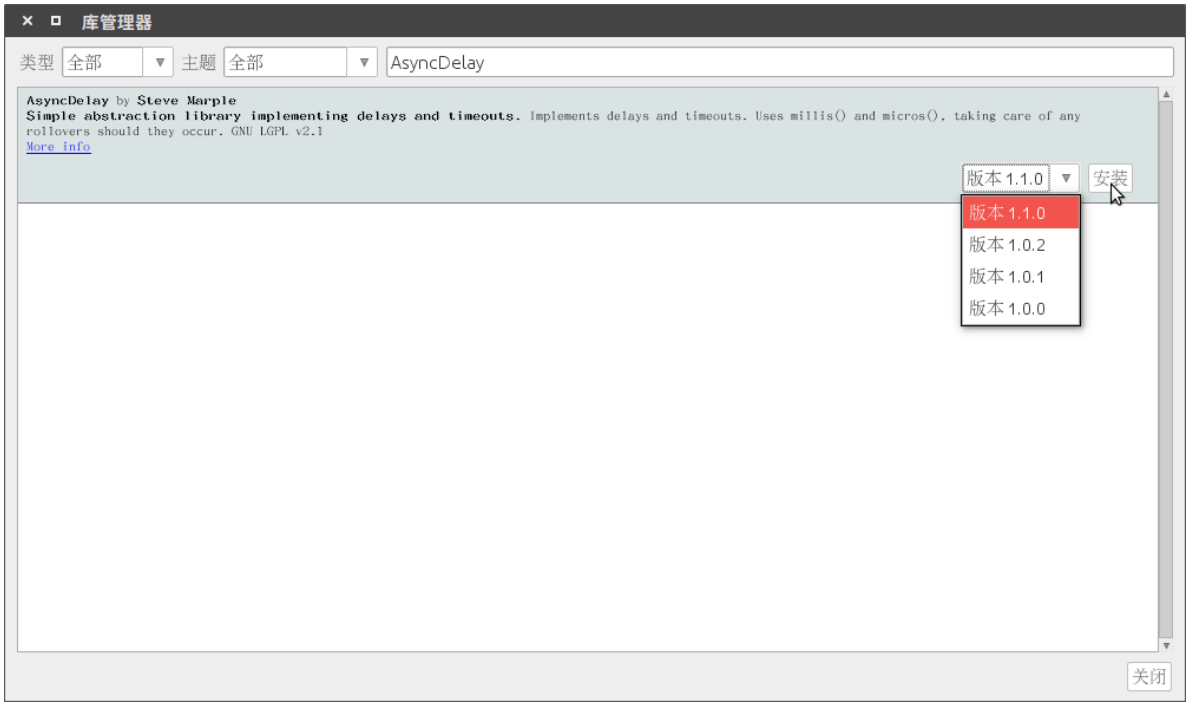
13.1.2 详细安装步骤

第一步：MoonBot Arduino 外部依赖库导入

- 1. 启动 Arduino 官方 IDE
- 2. 点击项目-> 加载库-> 管理库，打开库管理器



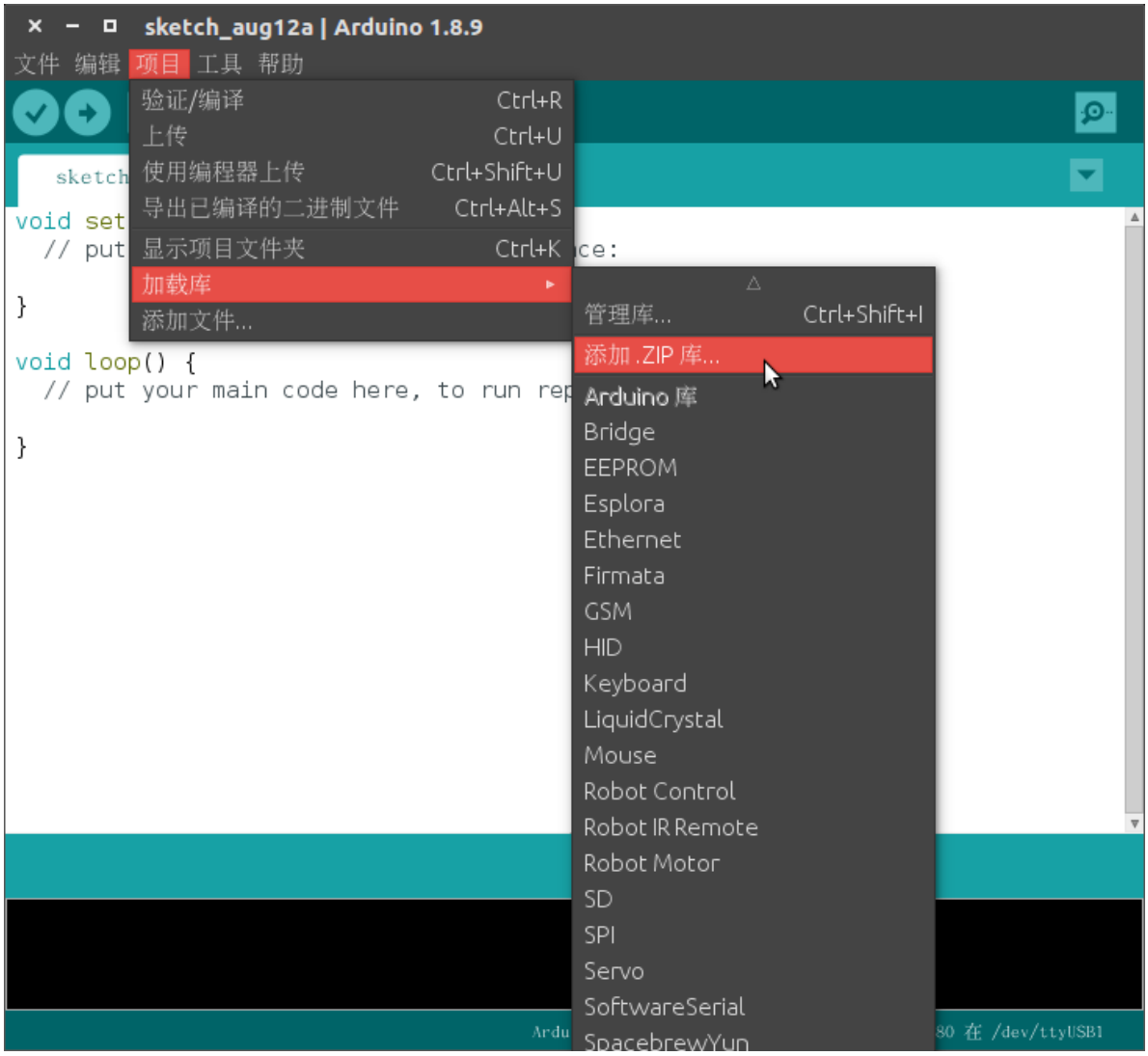
- 3. 搜索库 `AsyncDelay`，若没有安装则安装相关库，若库有更新，则进行更新



- 4. 按照第三步的安装方法安装库 `SoftwareWire` `Adafruit_NeoPixel` `Servo`，保证相关库安装到最新版

第二步：MoonBot Arduino 库导入

- 1. 在 github 下载最新的 MuVisionSensor3 的 Arduino 库和 MoonBot Arduino 库（Source code(zip)）
- 2. 点击项目-> 加载库-> 添加.zip 库，选中第一步下载的 MoonBot Arduino 库，完成库的导入



- 3. 重复上一步，导入 MuVisionSensor3 Arduino 库，完成库导入

第三步：连接设备

现在连接您的 MoonBot 连接到 PC，进行设备连接和端口配置

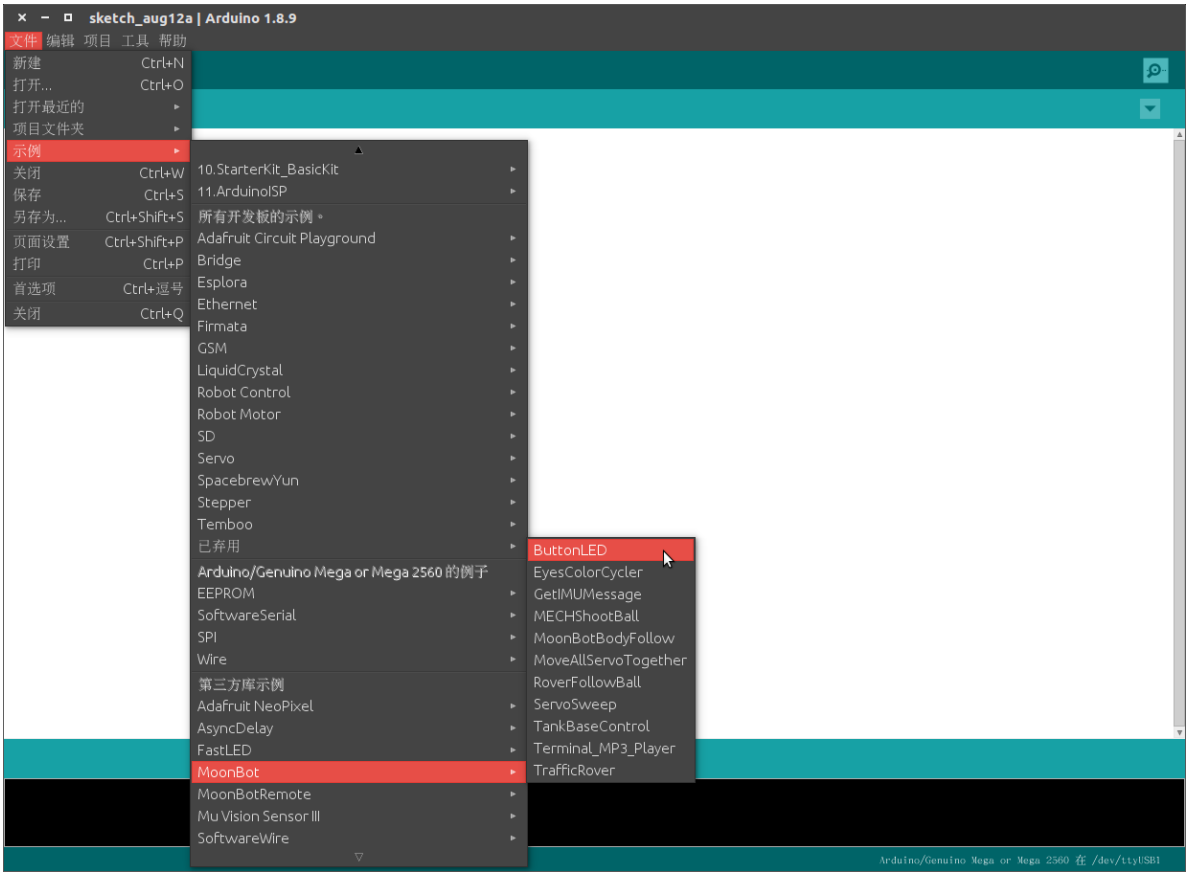
- 1. 点击工具-> 开发板，选择 `Arduino/Genuino Mega or Mega 2569`。
- 2. 点击工具-> 处理器，选择 `ATmega1280`
- 3. 点击工具-> 端口，选择对应的 MoonBot 端口

  通常，串口在不同操作系统下显示的名称有所不同：

  - **Windows 操作系统：** `COM1` 等
  - **Linux 操作系统：** 以 `/dev/tty` 开始
  - **MacOS 操作系统：** 以 `/dev/cu.` 开始

第四步：编译例程

- 1. 点击文件-> 示例->MoonBot，选择其中一个例程



- 2. 点击上传按钮，如果一切顺利，烧录完成后，开发板将会复位，对应例程会开始运行。

13.2 API 参考

MoonBot Arduino 中定制了 MoonBot Kit 和 MU Vision Sensor 3 的编程块，本文将对程序块逐一说明，以及一些复杂的程序示例。可结合之前的硬件模块示例进行学习。

MU Vision Sensor 3 教程见: MU Vision Sensor 3 教程

13.2.1 引脚映射

概览

MoonBot Kit 主控模块 包含了 9 个通用端口、4 个舵机端口、2 个电机端口、2 个按键、1 个蜂鸣器，对应的，在 Arduino 库中我们也提供了引脚映射 库和相关引脚的宏定义来方便用户获取对应引脚号。

通过这些函数和宏，我们可以方便的获取 MoonBot Kit 主控上的端口、舵机等对应引脚号。

按键状态获取

比如，通过获取板载按键的状态来点亮板载的 LED：

```
#include <MoonBot.h>

int button_a = MOONBOT_PIN_BUTTON_A;          // 获取板载按键 A 的引脚号
int button_b = MOONBOT_PIN_BUTTON_B;          // 获取板载按键 B 的引脚号

void setup()
{
    LED.begin();          // 板载 LED 初始化
}

void loop()
{
    if ((!digitalRead(button_a) && !digitalRead(button_b))) {
        // 如果按键 A 和 B 同时被按下，板载 LED 0 1 同时亮青色灯光
        LED.setPixelColor(0, 0x00ffff);
        LED.setPixelColor(1, 0x00ffff);
        LED.show();
    } else if ((!digitalRead(button_a))) {
        // 如果按键 A 和 B 同时被按下，板载 LED 0 亮绿色灯光
        LED.setPixelColor(0, 0x00ff00);
        LED.setPixelColor(1, 0x000000);
        LED.show();
    } else if ((!digitalRead(button_b))) {
        // 如果按键 A 和 B 同时被按下，板载 LED 1 亮蓝色灯光
        LED.setPixelColor(0, 0x000000);
        LED.setPixelColor(1, 0x0000ff);
        LED.show();
    } else {
        // 如果按键 A 和 B 同都没被按下，板载 LED 关闭
        LED.setPixelColor(0, 0x000000);
        LED.setPixelColor(1, 0x000000);
```

(下页继续)

(续上页)

```
        LED.show();
    }
}
```

获取 LED 眼睛引脚

再或者，我们希望获取连接在端口 3 上的眼睛模块 引脚来初始化 LED：

```
moonbot_eyes.setPin(moonbotPortToPin(kPort3, kPortPin1));      // 设置 LED 眼睛模块的引脚为
端口 3 的第一个引脚
moonbot_eyes.begin();          // LED 眼睛初始化
```

触摸模块状态获取

以及，读取连接在各个通用端口上触摸模块 的状态：

```
#include <MoonBot.h>

// connect touch sensor 1 on port 1
uint8_t touch1 = moonbotPortToPin(kPort1, kPortPin1);
// connect touch sensor 2 on port 2
uint8_t touch2 = moonbotPortToPin(kPort2, kPortPin1);

void setup()
{
  // initialize touch sensor 1/2 as INPUT_PULLUP
  pinMode(touch1, INPUT);
  pinMode(touch2, INPUT);
}

void loop()
{
  Serial.println("=======================");
  Serial.print("touch sensor1: ");
  // read touch sensor 1 state
  if (digitalRead(touch1)) {
    Serial.println("on touch");
  } else {
    Serial.println("not touch");
  }
  Serial.print("touch sensor2: ");
```

(下页继续)

(续上页)

```
  // read touch sensor 2 state
  if (digitalRead(touch2)) {
    Serial.println("on touch");
  } else {
    Serial.println("not touch");
  }
}
```

红外模块状态获取

同样的，我们可以用相同的方法读取红外模块 的状态：

```
#include <MoonBot.h>

// connect ir sensor 1 on port 1
uint8_t ir1[2] = {
    moonbotPortToPin(kPort1, kPortPin1),
    moonbotPortToPin(kPort1, kPortPin2)
};
// connect ir sensor 1 on port 1
uint8_t ir2[2] = {
    moonbotPortToPin(kPort2, kPortPin1),
    moonbotPortToPin(kPort2, kPortPin2)
};

void setup()
{
  // initialize ir sensor 1/2 as INPUT_PULLUP
  pinMode(ir1[0], INPUT);
  pinMode(ir1[1], INPUT);
  pinMode(ir2[0], INPUT);
  pinMode(ir2[1], INPUT);
}

void loop()
{
  Serial.println("======================");
  Serial.print("ir sensor1: ");
  // read ir sensor 1 state
  if (!digitalRead(ir1[0]) || !digitalRead(ir1[1])) {
    Serial.println("triggered");
  } else {
```

(下页继续)

(续上页)

```
    Serial.println("not triggered");
  }
  Serial.print("ir sensor2: ");
  // read ir sensor 2 state
  if (!digitalRead(ir2[0]) || !digitalRead(ir2[1])) {
    Serial.println("triggered");
  } else {
    Serial.println("not triggered");
  }
}
```

注意: 红外传感器为低电平触发，当对应引脚电平为 LOW 时，红外传感器被触发；反之，读取引脚电平为 HIGH 时，红外传感器不被触发。

API 参考 - 引脚映射

头文件

- src/pins_moonbot.h

枚举

enum moonbot_servo_t

- MoonBot Kit 舵机端口

值:

kServo1=0

kServo2

kServo3

kServo4

kServoNum

- 舵机端口数量

enum servo_pin_t

- 舵机端口引脚类型

值:

kSignal

- 舵机信号引脚

kShutDown

- 舵机供电引脚

kState

- 舵机状态引脚

enum moonbot_motor_t

- MoonBot Kit 电机端口

值:

kMotor1=0

kMotor2

kMotorNum

- 电机端口数量

enum motor_pin_t

- 电机端口引脚类型

值:

kDirection

- 电机方向引脚

kSpeed

- 电机速度引脚

enum moonbot_port_t

- MoonBot Kit 通用端口

值:

kPort1=0

kPort2

kPort3

kPort4

kPort5

kPort6

kPort7

kPort8

kPort9

kPortNum

- 通用端口数量

enum port_pin_t

- 通用端口引脚类型

值:

kPortPin1=0

kPortPin2

kPortPinNum

- 端口引脚数量

宏

MOONBOT_PIN_LED

- MoonBot Kit 主控板载 LED 引脚

MOONBOT_PIN_BUZZER_SIG

- MoonBot Kit 主控板载蜂鸣器信号引脚

MOONBOT_PIN_BUZZER_SHDW

- MoonBot Kit 主控板载蜂鸣器供电引脚

MOONBOT_PIN_BUTTON_A

- MoonBot Kit 主控板载按钮 A 引脚

MOONBOT_PIN_BUTTON_B

- MoonBot Kit 主控板载按钮 B 引脚

函数

uint8_t moonbotPortToPin(moonbot_port_t port_num, port_pin_t pin_num);

- 获取 MoonBot Kit 给定端口的引脚对应的 Arduino 引脚

  参数

  - `port_num`： 通用端口号
  - `pin_num`： 端口引脚号

返回

- 对应的 Arduino 引脚

uint8_t moonbotMotorToPin(moonbot_motor_t motor_num, motor_pin_t pin_type);

- 获取 MoonBot Kit 给定电机端口的引脚类型对应的 Arduino 引脚

参数

- `motor_num`： 电机端口号
- `pin_type`： 电机端口引脚类型

返回

- 对应的 Arduino 引脚

uint8_t moonbotServoToPin(moonbot_servo_t servo_num, servo_pin_t pin_type);

- 获取 MoonBot Kit 给定舵机端口的引脚类型对应的 Arduino 引脚

参数

- `servo_num`： 舵机端口号
- `pin_type`： 舵机端口引脚类型

返回

- 对应的 Arduino 引脚

13.2.2 电机

概览

MoonBot Kit 电机模块 包含了基础的电机及编码器。在对应的 Arduino 库中提供了电机 库以用来驱动单电机，以及底盘控制 库以用来驱动双电机。

我们可以通过在程序中包含 `MoonBot.h` 头文件来调用 `TankBase` 以驱动双电机底盘，或者通过调用 `Motor1` `Motor2` 以单独驱动电机。

控制双电机底盘

如何让底盘前后左右运动起来？先来看一个简单的例程：

```
#include <MoonBot.h>

void setup()
{
  TankBase.begin();    // enable TankBase, use default setting
```

(下页继续)

(续上页)

```
}

void loop()
{
  // forward 1s
  TankBase.write(100, 100);
  delay(1000);
  // backward 1s
  TankBase.write(-100, -100);
  delay(1000);
  // turn right 1s
  TankBase.write(100, -100);
  delay(1000);
  // turn left 1s
  TankBase.write(-100, 100);
  delay(1000);
}
```

通过调用 TankBase.write() 函数，分别写入左右电机速度，可以快速地让底盘运动起来。

如果你想精准地控制电机转速，推荐使用 TankBase.writeRpm() 函数来设置电机转速。

```
// 底盘左转，左电机速度 30 转/分，右电机速度-30 转/分
TankBase.writeRpm(30, -30);
```

注解: 使用某些可以精准的控制电机转速的函数（如：TankBase.write() TankBase.writeDistance() TankBase.writeAngle() 等），在硬件上需要连接编码器模块至对应端口，软件上需要在底盘初始化函数 TankBase.begin() 内，将参数 enc_enable 设为 true （该参数默认即为 true ）

你甚至可以让电机前行特定的距离或转动特定的角度：

```
void loop() {
    TankBase.writeDistance(30, 20);      // 底盘以 30 转/分的速度前行 20cm
    while(TankBase.read(kLeftMotor) || TankBase.read(kRightMotor));      // 等待底盘停止
    delay(100);
    TankBase.writeAngle(30, 180);        // 底盘以 30 转/分的速度右转 180°
    while(TankBase.read(kLeftMotor) || TankBase.read(kRightMotor));      // 等待底盘停止
    delay(100);
}
```

注解: 因为存在这安装误差、表面摩擦力等外部扰动，你可能会发现底盘无法走直，直行距离不到位或转动

角度不到位，没关系，通过以下几个函数校准底盘就可以将误差控制在 ±1% 以内：

```
void setup() {
    TankBase.rpmCorrection(82);            // 调节双轮速度一致%，校准转速误差
    TankBase.distanceCorrection(120);      // 校准直行距离误差%
    TankBase.wheelSpacingSet(100);         // 校准转弯角度误差%
}
```

控制单个电机

如果你只希望控制单个电机，可以通过调用 `Motor1` `Motor2` 来实现。

```
Motor1.write(100);       // 设置电机 1 模拟量为 100
Motor2.write(100);       // 设置电机 2 模拟量为 100
Motor1.writeRpm(30);     // 设置电机 1 转速速 30 转/分
Motor2.writeRpm(30);     // 设置电机 2 转速速 30 转/分
```

API 参考 - 电机

头文件

- src/MoonBot_Motor.h

枚举类型

enum moonbot_motor_t

- 电机端口类型

值:

kMotor1=0

kMotor2

kMotorNum

类

class Motor

- 基础单电机驱动

  成员函数

  Motor(moonbot_motor_t motor_type);

  - 构造函数, 指定端口类型。

    参数

    - `motor_type`： 电机端口类型

  int begin(const bool reverse_dir = false, const bool enc_enable = true);

  - 初始化给定端口的电机。

    参数

    - `reverse_dir`： 翻转电机转动方向，默认为 `false`
    - `enc_enable`： 开启电机编码器功能，默认为 `true`

    返回

    - `0`： 初始化成功
    - `-1`： 找不到给定的电机端口

  void write(int vol);

  - 写入模拟量。

    参数

    - `vol`： 电压的模拟量，取值 **±255** ，>0 正转，<0 反转

  int read(void);

  - 读取模拟量。

    返回

    - `±255`： 对应模拟量的值

  void writeStep(uint32_t step, int rpm = 30);

  - 驱动电机以给定转速转动给定步数后停止。
  - 该函数会调用编码器功能，必须在 `begin()` 函数内开启编码器功能后才可使用。

    参数

    - `step`： 转动步数，一圈对应 240 步

- rpm： 电机转动速度，默认转速 30 转/分（RPM）

void writeRpm(int rpm = 30);

- 写入电机转速，单位：转/分（RPM）
- 该函数会调用编码器功能，必须在 begin() 函数内开启编码器功能后才可使用。

  参数

  - rpm： 电机转动速度，默认转速 30 转/分（RPM）

int readRpm(void);

- 读取电机转速，单位：转/分（RPM）
- 该函数会调用编码器功能，必须在 begin() 函数内开启编码器功能后才可使用。

  返回

  - 电机转动速度

void writeDistance(int rpm, uint32_t distance_cm);

- 驱动电机以给定转速转动给定距离后停止，因电机在安装和表面摩擦会引起误差，使用该函数前建议先使用函数 distanceCorrection() 进行校准
- 该函数会调用编码器功能，必须在 begin() 函数内开启编码器功能后才可使用。

  参数

  - rpm： 电机转动速度
  - distance_cm： 转动距离，单位：厘米（cm）

uint32_t readEncoderPulse(void);

- 读取编码器计数值。
- 该函数会调用编码器功能，必须在 begin() 函数内开启编码器功能后才可使用。

  返回

  - 当前编码器计数值

void rpmCorrection(uint8_t percent);

- 电机转速校准

  参数

  - percent： 校正百分比，>100 转速增加，<100 转速降低

void distanceCorrection(uint8_t percent);

- 电机直行距离校准

  参数

  - `percent`： 校正百分比，>100 距离增加，<100 距离降低

API 参考 - 底盘控制

头文件

- src/MoonBot_TankBase.h

枚举类型

enum motor_type_t

- 底盘电机类型

值:

kLeftMotor=0

- 左侧电机

kRightMotor

- 右侧电机

类

class MoonBotTankBase

- 底盘双电机驱动程序

  成员函数

  MoonBotTankBase(Motor& left_motor, Motor& right_motor);

  - 构造函数，指定左右电机至指定端口。

    参数

    - `left_motor`： 左电机
    - `right_motor`： 右电机

  int begin(const bool reverse_dir = false, const bool enc_enable = true);

  - 初始化坦克底盘的电机。

参数

- reverse_dir： 翻转电机转动方向，默认为 false
- enc_enable： 开启电机编码器功能，默认为 true

返回

- 0： 初始化成功
- -1： 找不到给定的电机端口

int begin(const bool left_reverse_dir, const bool right_reverse_dir, const bool enc_enable);

- 初始化坦克底盘的电机。

  参数

  - left_reverse_dir： 翻转左电机转动方向
  - right_reverse_dir： 翻转右电机转动方向
  - enc_enable： 开启电机编码器功能

  返回

  - 0： 初始化成功
  - -1： 找不到给定的电机端口

void write(int left_vol, int right_vol);

- 向左右电机写入模拟量。

  参数

  - left_vol： 左电压的模拟量，取值 **±255** ，>0 正转，<0 反转
  - right_vol： 右电压的模拟量，取值 **±255** ，>0 正转，<0 反转

int read(motor_type_t motor_type);

- 读取指定电机的模拟量。

  参数

  - motor_type： 电机类型

  返回

  - ±255： 对应模拟量的值

uint32_t readEncoderPulse(motor_type_t motor_type);

- 读取对应电机编码器计数值。

- 该函数会调用编码器功能，必须在 `begin()` 函数内开启编码器功能后才可使用。

  参数

  - `motor_type`： 电机类型

  返回

  - 当前编码器计数值

void writeRpm(int left_rpm, int right_rpm);

- 向左右电机写入转速，单位：转/分（RPM）
- 该函数会调用编码器功能，必须在 `begin()` 函数内开启编码器功能后才可使用。

  参数

  - `left_rpm`： 左电机转动速度
  - `right_rpm`： 右电机转动速度

int readRpm(motor_type_t motor_type);

- 读取对应电机转速，单位：转/分（RPM）
- 该函数会调用编码器功能，必须在 `begin()` 函数内开启编码器功能后才可使用。

  参数

  - `motor_type`： 电机类型

  返回

  - 电机转动速度

void writeDistance(int rpm, uint32_t distance_cm);

- 驱动左右电机以给定转速直行给定距离后停止，因电机在安装和表面摩擦会引起误差，使用该函数前建议先使用函数 `rpmCorrection()` 和 `distanceCorrection()` 进行校准
- 该函数会调用编码器功能，必须在 `begin()` 函数内开启编码器功能后才可使用。

  参数

  - `rpm`： 电机转动速度
  - `distance_cm`： 转动距离，单位：厘米（cm）

void writeAngle(int rpm, uint32_t angle);

- 驱动左右电机以给定转速转弯给定角度后停止，因电机在安装和表面摩擦会引起误差，使用该函数前建议先使用函数 `rpmCorrection()` 和 `wheelSpacingSet()` 进行校准
- 该函数会调用编码器功能，必须在 `begin()` 函数内开启编码器功能后才可使用。

参数

- `rpm`： 电机转动速度
- `angle`： 转动角度，单位：度（°）

void wheelSpacingSet(int correct, float space_cm = 0);

- 设定轮子间距，校正转弯角度，该函数可以校准轮子转弯不到位的情况

参数

- `correct`： 校正值，>100 转弯角度变大，小于 100 转弯角度减小
- `space_cm`： 电机间距

void rpmCorrection(int percent);

- 校准左右电机转速

参数

- `percent`： 校正值，>100 向右校正，<100 向左校正

void distanceCorrection(int percent);

- 底盘直行距离校准

参数

- `percent`： 校正百分比，>100 距离增加，<100 距离降低

void forward(unsigned int step, unsigned int rpm = 30);

- 底盘向前直行一定距离后停止。

参数

- `step`： 前行距离，单位：厘米（cm）
- `rpm`： 电机转动速度，默认转速 30 转/分（RPM）

void backward(unsigned int step, unsigned int rpm = 30);

- 底盘向后直行一定距离后停止。

参数

- `step`： 后退距离，单位：厘米（cm）

- `rpm`： 电机转动速度，默认转速 30 转/分（RPM）

void turnLeft(unsigned int step, unsigned int rpm = 30);

- 底盘左转一定角度后停止。

  参数

  - `step`： 左转角度，单位：度（°）
  - `rpm`： 电机转动速度，默认转速 30 转/分（RPM）

void turnRight(unsigned int step, unsigned int rpm = 30);

- 底盘右转一定角度后停止。

  参数

  - `step`： 右转角度，单位：度（°）
  - `rpm`： 电机转动速度，默认转速 30 转/分（RPM）

void stop(void);

- 底盘停止运行。

13.2.3 音乐

概览

MoonBot Kit 提供了两类发声设备，分别是主控模块 板载的蜂鸣器模块和外接的扬声器模块 。相对应的，我们可以通过调用 Arduino 的基础函数 tone() noTone() 函数来控制蜂鸣器发声，通过扬声器 库来控制扬声器发声。

我们可以通过在程序中包含 `MoonBot.h` 头文件来调用 `speaker` 来驱动外部扬声器。

板载蜂鸣器驱动

我们可以通过调用宏定义 `MOONBOT_PIN_BUZZER_SIG` 来获取对应的 Arduino 引脚，通过控制引脚 `MOONBOT_PIN_BUZZER_SHDW` 电平的高低来开启或关闭板载的蜂鸣器功能。

```
:emphasize-lines: 7

#include <MoonBot.h>

void setup()
{
    pinMode(MOONBOT_PIN_BUZZER_SIG, OUTPUT);         // 初始化蜂鸣器信号引脚为输出模式
    pinMode(MOONBOT_PIN_BUZZER_SHDW, OUTPUT);        // 初始化蜂鸣器供电引脚为输出模式
```

(下页继续)

(续上页)

```
    digitalWrite(MOONBOT_PIN_BUZZER_SHDW, LOW);      // 将蜂鸣器供电引脚电平拉低以开启蜂鸣器
    tone(MOONBOT_PIN_BUZZER_SIG, 1000, 2000);        // 让蜂鸣器以 1000Hz 的频率开始播放声音，
播放 2000ms 后停止
}
```

注意： 如上述例程的第七行所述：

```
:lineno-start: 7

digitalWrite(MOONBOT_PIN_BUZZER_SHDW, LOW);      // 将蜂鸣器供电引脚拉低以开启蜂鸣器
```

将蜂鸣器供电引脚 MOONBOT_PIN_BUZZER_SHDW 电平设为 LOW 即为开启蜂鸣器，设为 HIGH 即为关闭蜂鸣器功能。该引脚电平默认为 LOW 。

外接扬声器驱动

MoonBot Kit 提供的扬声器模块 使用了 WT2003S MP3 解码芯片，通过调用 speaker 你可以轻松自制一台 MP3 播放器！

先来看一个简单的 MP3 播放器的例程：

```
#include <MoonBot.h>

void setup()
{
    speaker.begin(Serial2);      // 初始化 speaker 至 Arduino 的串口 2（MoonBot 对应的端口 2）

    speaker.setPlayMode(0);      // 播放模式设置为单曲播放播放模式，播放完一首曲子后停止播放
    speaker.setVolume(20);       // 音量设置为 20，最大为 32
}

void loop()
{
    if ((!digitalRead(MOONBOT_PIN_BUTTON_A))) {
        // 若按键 A 被按下
        speaker.playNext();      // 播放下一首歌
    } else if ((!digitalRead(MOONBOT_PIN_BUTTON_B))) {
        // 若按键 B 被按下
        speaker.playPrevious(); // 播放上一首歌
    }
}
```

也可以查看 官方串口 MP3 播放器 的例程以获取更全面的信息。

API 参考 - 扬声器

头文件

- src/MoonBot_WT2003S_MP3_Decoder.h

类

class WT2003S

- WT2003S MP3 播放器驱动。

  成员函数

  void begin(SoftwareSerial &serialPort);

  - 以软串口作为端口初始化扬声器。

    参数

    - `serialPort`： 软串口

  void begin(HardwareSerial &serialPort = Serial);

  - 以硬件串口作为端口初始化扬声器。

    参数

    - `serialPort`： 硬件串口，默认为 Serial

  uint8_t play(char* fileName);

  - 播放给定文件名的音乐。

    参数

    - `fileName`： 音乐文件名前四个字节

    返回

    - `0` 命令正常执行，其他命令出错

  uint8_t setVolume(uint8_t volumeLevel);

  - 设置扬声器音量

    参数

    - `volumeLevel`： 扬声器音量，取值范围 `0~32`

    返回

    - `0` 命令正常执行，其他命令出错

  uint8_t stop(void);

- 停止播放当前正在播放的音乐。

  返回

  - 0 命令正常执行，其他命令出错

void pause(void);

- 播放时调用此函数，则暂停当前播放。没有播放时调用此函数，则播放当前音乐。

uint8_t playPrevious(void);

- 播放上一曲音乐，在播放第一曲音乐时，发送该指令可触发播放最后一曲音乐。

  返回

  - 0 命令正常执行，其他命令出错

uint8_t playNext(void);

- 播放下一曲音乐，在播放最后一曲音乐时，发送该指令可触发播放第一曲音乐。

  返回

  - 0 命令正常执行，其他命令出错

uint8_t setPlayMode(uint8_t mode);

- 设置扬声器播放模式。

  参数

  - mode：

0	单曲播放
1	单曲循环
2	列表循环
3	随机播放

  返回

  - 0 命令正常执行，其他命令出错

uint16_t getSongCount(void);

- 获取当前音乐在列表中的序号。

  返回

  - 当前音乐在列表中的序号

void getSongName();

- 获取当前音乐的乐曲名前 9 个字节。执行此函数后可以通过读取 `WT2003S::songName[MP3_NUM_NAME_BYTES]` 来获取乐曲名。

uint8_t playTrackNumber(uint8_t trackNumber);

- 播放给定序号的音乐。

  参数

  - `trackNumber`： 音乐在列表中的序号

  返回

  - `0` 命令正常执行，其他命令出错

uint8_t getVolume(void);

- 获取当前扬声器的音量值。

  返回

  - `0~32`： 扬声器音量值。

uint8_t getPlayStatus(void);

- 获取当前播放状态。

  返回

1	播放
2	停止
3	暂停

13.2.4 IMU

概览

MoonBot Kit 主控模块 集成了及三轴磁力计、三轴加速度、温度传感器三种功能于一体的 IMU 模组。对应的，在 Arduino 库中我们也提供了*IMU* 库来方便用户获取主控当前姿态、方向、温度等状态。

通过调用 `IMU` 我们可以快速的获取当前的指南针角度、俯仰角、横滚角、重力加速度、温度等状态值。

读取主控当前方向

通过读取指南针的角度，我们可以知道当前主控所处方向：

```
#include <MoonBot.h>

void setup()
{
  IMU.enable();          // IMU 使能
  IMU.calibrateMag();    // IMU 磁力计校准，校准时以"∞" 字形晃动主控
}

void loop()
{
  Serial.print("compass:");
  // 获取指南针角度（0~360°），指向正北时值为 0 或 360
  Serial.println(IMU.getMagAngle());
}
```

注解： 主控平放时，返回数值为 Y 轴（见主控正面丝印标识）与正北方向夹角；主控竖放时，返回数值为 Z 轴与正北方向夹角。

获取取俯仰角、横滚角

```
// 获取俯仰角（±180°），主控向上角度为正，向下角度为负
int pitch = IMU.getAccAngle(kAccPitch);
// 获取横滚角（±180°），主控右倾为正，左倾为正
int roll = IMU.getAccAngle(kAccRoll);
```

注解： MoonBot Kit 主控正方向为 Y 轴（见主控正面丝印标识），姿态角度都是以此为前提进行计算的。

获取当前加速度

```
// 获取加速度，单位：g，静止时数值为 1.0 左右
float acceleration = IMU.getAcceleration();
```

获取当前运动状态

```
void loop()
{
    if (IMU.on(kIMUShake)) {
        // 如果当前主控在晃动
        // 亮红灯
        LED.setPixelColor(0, 0xff0000);
        LED.setPixelColor(1, 0xff0000);
        LED.show();
    } else if (IMU.on(kIMUFreeFall)) {
        // 如果当前主控在自由落体
        // 亮绿灯
        LED.setPixelColor(0, 0x00ff00);
        LED.setPixelColor(1, 0x00ff00);
        LED.show();
    } else {
        // 如果主控静置
        // 关闭 LED
        LED.setPixelColor(0, 0x000000);
        LED.setPixelColor(1, 0x000000);
        LED.show();
    }
}
```

API 参考 - IMU

头文件

- src/LSM303AGR_IMU_Sensor.h

枚举

enum lsm303_axes_t

- IMU 方向轴类型。

值:

kDirX

kDirY

kDirZ

enum lsm303_acc_angle_t

- IMU 姿态角度类型。

值:

kAccRoll

kAccPitch

enum imu_state_t

- IMU 特殊状态类型。

值:

kIMUShake

- IMU 是否处于晃动状态

kIMUFreeFall

- IMU 是否处于自由落体状态

类

class LSM303AGR_IMU_Sensor

- IMU 驱动。

成员函数

int enable(void);

- 使能 IMU 单元。

返回

- `0` 使能成功，否则失败

int advGetMagAngle(lsm303_axes_t main_axes, lsm303_axes_t sub_axes);

- 获取给定主轴与副轴所在平面，主轴与正北方夹角

参数

- `main_axes`： 主轴
- `sub_axes`： 副轴

返回

- 主轴与正北方夹角

int getMagAngle(void);

- 获取指南针角度，主控平放时，返回 Y 轴正方向与正北方夹角；
  主控竖放时，返回 Z 轴正方向与正北方夹角。

返回

- 主轴与正北方夹角

int getAccAngle(lsm303_acc_angle_t angle_type);

- 获取主控姿态角度。

  参数

  - `angle_type`： 姿态角度类型

  返回

  - 姿态角度

float getAcceleration(void);

- 获取加速度值。

  返回

  - 加速度值，单位：g

bool on(imu_state_t imu_state);

- 获取主控知否处于某些状态。

  参数

  - `imu_state`： IMU 状态

  返回

  - `true` IMU 处于该状态，否则不处于该状态

bool calibrateMag(void);

- 磁力计校准

  返回

  - 是否校准完成

int16_t temperature(void);

- 获取温度原始数值

  返回

  - 温度原始数值

float temperatureC(void);

- 获取当前温度，单位：摄氏度（℃）

  返回

  - 当前温度，单位：摄氏度（℃）

float temperatureF(void);

- 获取当前温度，单位：华氏度（℉）

返回

- 当前温度，单位：华氏度（℉）

成员变量

LSM303AGR_ACC_Sensor Acc;

- 加速度驱动

LSM303AGR_MAG_Sensor Mag;

- 磁力计驱动

13.2.5 舵机

概览

MoonBot Kit 主控模块 最多可连接四个舵机模块 。在 Arduino 库中，我们也提供了舵机 库，通过这个库，您可以控制一个或多个舵机同时运动。

舵机库 继承了 Arduino 基础的舵机驱动 `class Servo` 类，除了基础的 `Servo` 类成员函数外，我们还提供了舵机校准、多个舵机同时运行等功能的函数。在 `MoonBot.h` 头文件中，我们提供了四个变量 `m_servo[kServo1]` `m_servo[kServo2]` `m_servo[kServo3]` `m_servo[kServo4]` ，以分别驱动主控上对应的四个舵机接口。

基础应用

先来看一个舵机的基础应用：

```
#include <MoonBot.h>

int pos;

void setup() {
    m_servo[kServo1].attach(kServo1, true);          // attaches servo on servo port␣
↪1, and reverse directions
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {     // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    m_servo[kServo1].write(pos);                     // tell servo to go to position␣
↪in variable 'pos'
```

(下页继续)

(续上页)

```
    delay(15);                                  // waits 15ms for the servo to reach the␣
↪position
  }
  for (pos = 180; pos >= 0; pos -= 1) {      // goes from 180 degrees to 0 degrees
    m_servo[kServo1].write(pos);                        // tell servo to go to position␣
↪in variable 'pos'
    delay(15);                                  // waits 15ms for the servo to reach the␣
↪position
  }
}
```

注解: 舵机的初始化函数已改为 `attach(moonbot_servo_t servo_port, bool reverse)`，基础的舵机初始化函数 `uint8_t attach(int pin)` 已不受支持。

多个舵机同时运动

我们提供了 `void MoonBotServo::setTargetAngle()` 和 `MoonBotServo::moveAllServoToTarget()` 函数来让多个舵机同时运动。

```
#include <MoonBot.h>

void setup() {
  for (int i = 0; i < kServoNum; ++i) {
    m_servo[i].attach((moonbot_servo_t)i);  // attaches servo
  }
}
void loop() {
  // in steps of 1 degree
  for (int i = 0; i < kServoNum; ++i) {
    m_servo[i].setTargetAngle(180, 1);      // set all servo to go to position in␣
↪variable '180', speed 1 degree per pulse(20ms)
  }
  MoonBotServo::moveAllServoToTarget();     // move all servo to target angle
  for (int i = 0; i < kServoNum; ++i) {
    m_servo[i].setTargetAngle(0, 1);        // set all servo to go to position in␣
↪variable '0', speed 1 degree per pulse(20ms)
  }
  MoonBotServo::moveAllServoToTarget();     // move all servo to target angle
}
```

注解： 当使用 `MoonBotServo::moveAllServoToTarget();` 函数的默认参数时，该函数会等待所有舵机动作运行完成后退出; 当参数不为 0 时，会在参数指定时间内退出，并返回动作是否完成。我们可以通过调用 `bool isMoving(void);` 来每隔一段时间检查一次当前的运行状态：

```
while (!MoonBotServo::moveAllServoToTarget(0)) {
    // 当检测到当前有舵机在进行运动时
    for (int i = 0; i < kServoNum; ++i) {
        if (!m_servo[i].isMoving()) {
            // 如果检查到当前端口舵机停止运行，打印舵机状态
            Serial.print("Servo");
            Serial.print(i);
            Serial.println(" Stopped.");
        }
    }
}
Serial.println("All Servo Stopped.");
```

改为以上形式后，使用串口调试软件即会接收到类似以下的信息

```
Servo1 Stopped.
...
Servo1 Stopped.
Servo2 Stopped.
...
Servo2 Stopped.
Servo3 Stopped.
...
Servo3 Stopped.
All Servo Stopped.
```

舵机校准

MoonBot Kit 舵机库 中提供了舵机校准函数，可在初始化过程中校准舵机的偏差。

```
m_servo[kServo1].correction(-2);          //将舵机 1 向下校准 2°
```

API 参考 - 舵机

头文件

- src/MoonBot_Servo.h

枚举

enum moonbot_servo_t

- 舵机端口类型

值:

kServo1

kServo2

kServo3

kServo4

kServoNum

- 舵机端口数量

类

class MoonBotServo

- MoonBot Kit 舵机驱动库

成员函数

uint8_t attach(moonbot_servo_t servo_port, bool reverse = MOONBOT_SERVO_REVERSE);

- 初始化舵机至对应的舵机端口

参数

- `servo_port`： 舵机端口
- `reverse`： 翻转舵机角度

返回

- `NOT_A_PORT` 舵机端口无效，其他初始化正确

uint8_t attach(moonbot_servo_t servo_port, int min, int max, bool reverse = MOONBOT_SERVO_REV

- 初始化舵机至对应的舵机端口，并指定舵机的运行范围

参数

- servo_port： 舵机端口
- min： 舵机最小运行角度
- max： 舵机最大运行角度
- reverse： 翻转舵机角度

返回

- NOT_A_PORT 舵机端口无效，其他初始化正确

void detach(void);

- 断开舵机与对应端口的连接

void write(int value);

- 向舵机写入角度

  参数

  - value： 角度值，取值范围 0~180°

int read(void);

- 读取舵机当前角度

  返回

  - 当前角度值

void reverse(bool state);

- 以 90° 为中心点，翻转舵机角度。

  参数

  - state：状态，true 为翻转角度，否则为默认方向

void setTargetAngle(int angle, unsigned int speed = 1);

- 预设舵机角度，需与函数 static bool moveAllServoToTarget() 配合使用。

  参数

  - angle： 预设角度
  - speed： 每个脉冲角度的变化量

void stop(void);

- 停止舵机动作

void power(bool state);

- 开启或断开舵机供电。

  参数

  - `state`： 舵机供电状态，`true` 为开启供电

void correction(int angle_offset);

- 舵机校准

  参数

  - `angle_offset`：舵机校准角度值，取值范围：`±90°`

bool isMoving(void);

- 读取舵机当前运动状态。

  返回

  - `true` 为当前正在运动，反之为当前没有运动

bool isPowerOverload(void);

- 检测当前是否电流过载。

  返回

  - `true` 为当前电流过载，反之为正常

静态成员函数

static bool moveAllServoToTarget(unsigned long timeToWait_ms = 0xFFFFFFFF);

- 移动所有舵机值预设角度。

  参数

  - `timeToWait_ms`： 等待时间，默认为直至舵机移动到预设角度（无限长）

  返回

  - `true` 完成移动到预设角度动作，反之未完成动作

static void stopAllServo(void);

- 停止所有舵机的动作。

13.2.6 灯光

概览

MoonBot Kit 中包含了两组灯光模块，分别位于主控模块 和眼睛模块 。我们可以使用 Adafruit_NeoPixel 库去驱动这两组灯光模块。

我们可以通过调用 LED 来驱动板载的两颗 LED 灯，通过调用 moonbot_eyes 来驱动 12 颗 LED 眼睛。同时通过调用*LED 眼睛动作* 让眼睛做出丰富的表情。

LED 基础驱动

```
#include <MoonBot.h>

void setup() {
    // 使能主控 LED
    LED.begin();
    moonbot_eyes.begin();
    //  clear LED color
    LED.clear();
    LED.show();
    moonbot_eyes.clear();
    moonbot_eyes.show();
}

void loop() {
    if (digitalRead(MOONBOT_PIN_BUTTON_A) == LOW
        && digitalRead(MOONBOT_PIN_BUTTON_B) == LOW) {
        // 如果 A&B 同时被按下
        // LED 和眼睛灯光显示青色
        LED.fill(0x003030);
        LED.show();
        moonbot_eyes.fill(0x003030);
        moonbot_eyes.show();
    } else if (digitalRead(MOONBOT_PIN_BUTTON_A) == LOW) {
        // 如果按键 A 被按下
        // LED0, 眼睛右眼显示绿色
        LED.setPixelColor(0, 0x003000);
        LED.setPixelColor(1, 0);
        LED.show();
        moonbot_eyes.clear();
        moonbot_eyes.fill(0x003000, 0, 6);
        moonbot_eyes.show();
    } else if (digitalRead(MOONBOT_PIN_BUTTON_B) == LOW) {
```

(下页继续)

(续上页)

```
        // 如果按键 B 被按下
        // LED1, 眼睛左眼显示蓝色
        LED.setPixelColor(0, 0);
        LED.setPixelColor(1, 0x000030);
        LED.show();
        moonbot_eyes.clear();
        moonbot_eyes.fill(0x000030, 6, 6);
        moonbot_eyes.show();
    } else {
        // LED 眼睛灯光关闭
        LED.clear();
        LED.show();
        moonbot_eyes.clear();
        moonbot_eyes.show();
    }
}
```

LED 眼睛灯光动作

MoonBot Kit 提供了丰富的眼睛动作库以供调用：

```
colorWipe(moonbot_eyes, 0x40, 200);          // 眼睛灯珠逐个变成绿色
theaterChase(moonbot_eyes, 0xFF00, 50);      // 眼睛转圈
MoonBotEyesCircle(moonbot_eyes, 50);         // 眼睛渐变转圈
rainbow(moonbot_eyes, 5);                    // 眼睛彩虹色渐变
rainbowCycle(moonbot_eyes, 5);               // 眼睛灯珠逐个以不同颜色开始彩虹色转圈渐变
```

API 参考 - Adafruit_NeoPixel

详情见 Adafruit 官网介绍：https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use

API 参考 - LED 眼睛动作

头文件

- src/MoonBot_Eyes.h

枚举类型

enum moonbot_eyes_t

- 眼睛类型

值:

kEyesLeft

- 左眼

kEyesRight

- 右眼

kEyesBoth

- 双眼

enum moonbot_look_t

- 眼睛方向类型

值:

kEyesLookUp

- 眼睛向上看

kEyesLookDown

- 眼睛向下看

kEyesLookLeft

- 眼睛向左看

kEyesLookRight

- 眼睛向右看

enum moonbot_eyes_scroll_t

- 眼睛滚动方向类型

值:

kEyesScrollUp

- 眼睛向上滚动

kEyesScrollDown

- 眼睛向下滚动

kEyesScrollLeft

- 眼睛向左滚动

kEyesScrollRight

- 眼睛向右滚动

函数

void colorFade(Adafruit_NeoPixel& led, uint8_t r, uint8_t g, uint8_t b, uint8_t wait);

- LED 颜色从当前颜色渐变到指定颜色。

  参数

  - `led`： LED 类型
  - `r`： 红色通道值
  - `g`： 绿色通道值
  - `b`： 蓝色通道值
  - `wait`： 变化时间

void colorWipe(Adafruit_NeoPixel& led, uint32_t c, uint8_t wait);

- LED 灯光颜色逐个变化。

  参数

  - `led`： LED 类型
  - `c`： LED 灯光 RGB 颜色
  - `wait`： 灯光动作等待时间

void rainbow(Adafruit_NeoPixel& led, uint8_t wait);

- LED 灯光从彩虹色依次渐变。

  参数

  - `led`： LED 类型
  - `wait`： 渐变时间

void rainbowCycle(Adafruit_NeoPixel& led, uint8_t wait) ;

- LED 每颗灯珠以不同颜色灯光从彩虹色依次渐变。

  参数

  - `led`： LED 类型
  - `wait`： 渐变时间

void theaterChase(Adafruit_NeoPixel& led, uint32_t c, uint8_t wait);

- LED 以指定颜色转圈。

参数

- `led`： LED 类型
- `c`： LED 灯光 RGB 颜色
- `wait`： 灯光动作等待时间

void MoonBotEyesLook(Adafruit_NeoPixel& led, moonbot_look_t look_tpye, uint32_t color);

- LED 眼睛向某个方向看。

参数

- `led`： LED 类型
- `look_tpye`： 眼睛方向类型
- `color`： 眼睛颜色

void MoonBotEyesScroll(Adafruit_NeoPixel& led, moonbot_eyes_scroll_t scroll_tpye, uint32_t color, uint8_t wait = 50);

- LED 眼睛向某个方向滚动。

参数

- `led`： LED 类型
- `scroll_tpye`： 眼睛滚动方向类型
- `color`： 眼睛颜色
- `wait`： 滚动时间，默认为 50ms

void MoonBotEyesCircle(Adafruit_NeoPixel& led, uint32_t color, moonbot_eyes_t eyes_type = kEyesBoth, uint8_t wait = 50);

- LED 眼睛渐变转圈滚动。

参数

- `led`： LED 类型
- `color`： 眼睛颜色
- `eyes_type`： 眼睛类型
- `wait`： 滚动时间，默认为 50ms

CHAPTER 14

MoonBot Kit 固件升级向导

本文档旨在指导用户更新 MoonBot Kit 主控模块 及视觉模块 的相关固件。

查看视频教程：固件更新视频教程

14.1 准备工作

硬件：

- MoonBot 开发者套件
- PC（Windows）

软件：

- Arduino 官方 IDE
- MoonBot Arduino 库
- MU 视觉传感器固件更新软件
- MU(for MoonBot) 最新版固件 (.bin 文件)

14.2 详细操作步骤

14.2.1 步骤一：更新 MoonBot Kit 主控固件

主控固件升级可参考：*MoonBot Kit APP 固件升级向导*

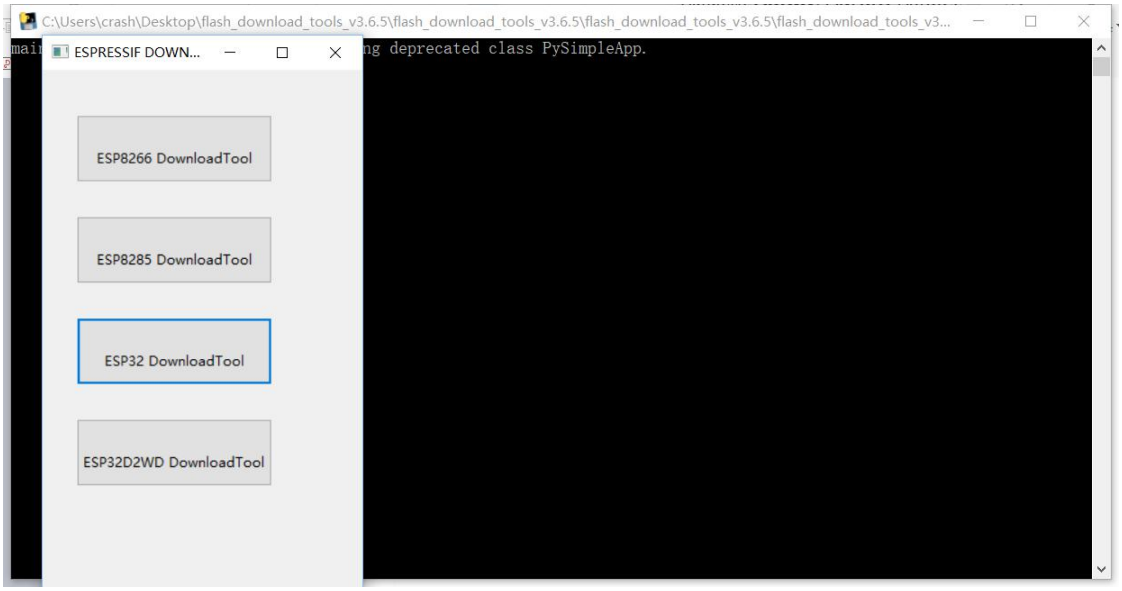
14.2.2 步骤二：更新 MU 视觉传感器 3 固件

1. 将视觉模块 连接至 MoonBot Kit 主控模块 的端口 P9，并将主控连接至电脑

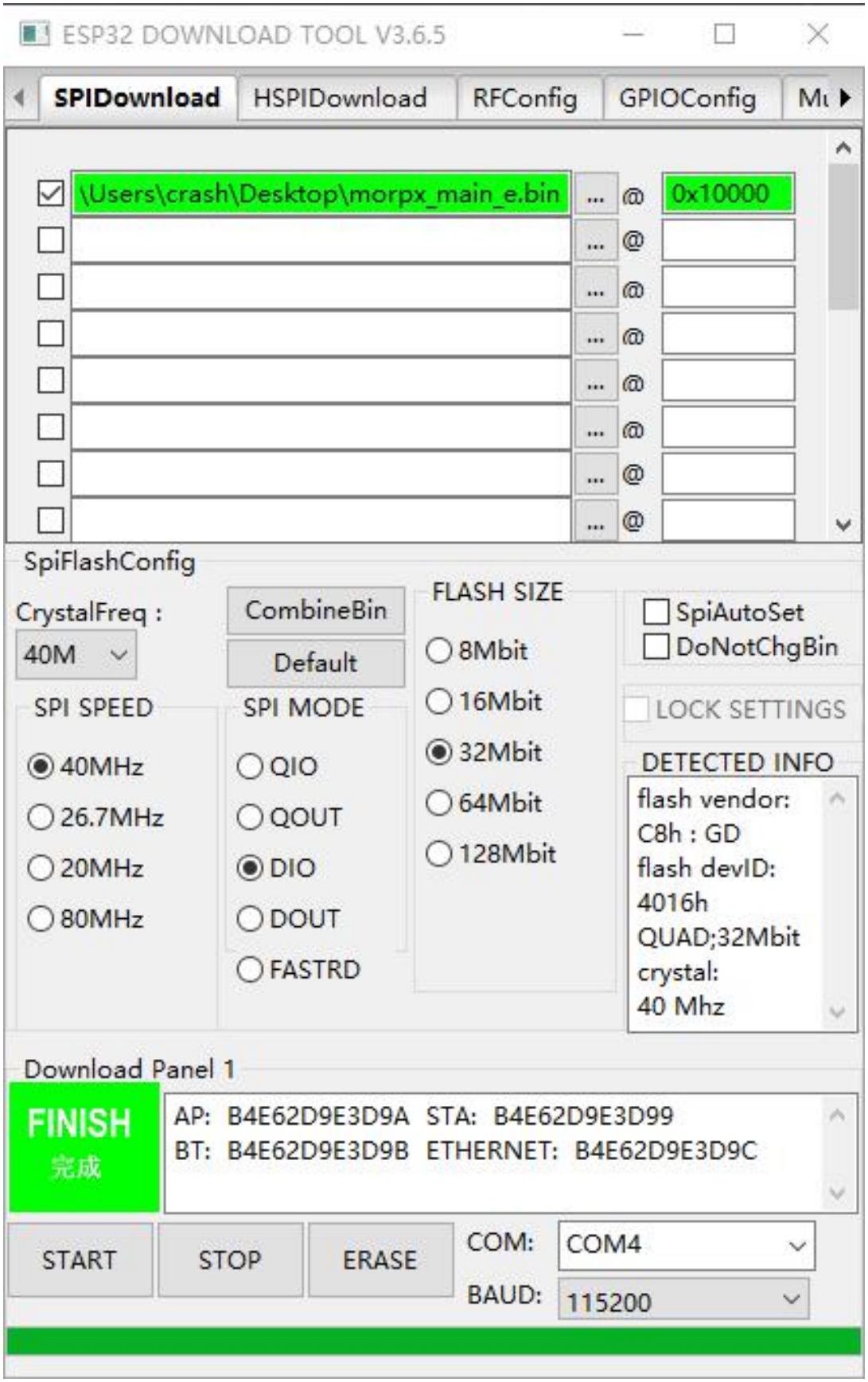
2. 按住视觉模块 左侧的 Mode 键, 再短按一次右侧 Reset 键重启, 即可进入烧录模式，此时可以松开 Mode 键

3. 双击打开 MU 视觉传感器固件更新软件 `flash_download_tools_vx.x.x.exe`

4. 点击选择 `ESP32 DownloadTool`

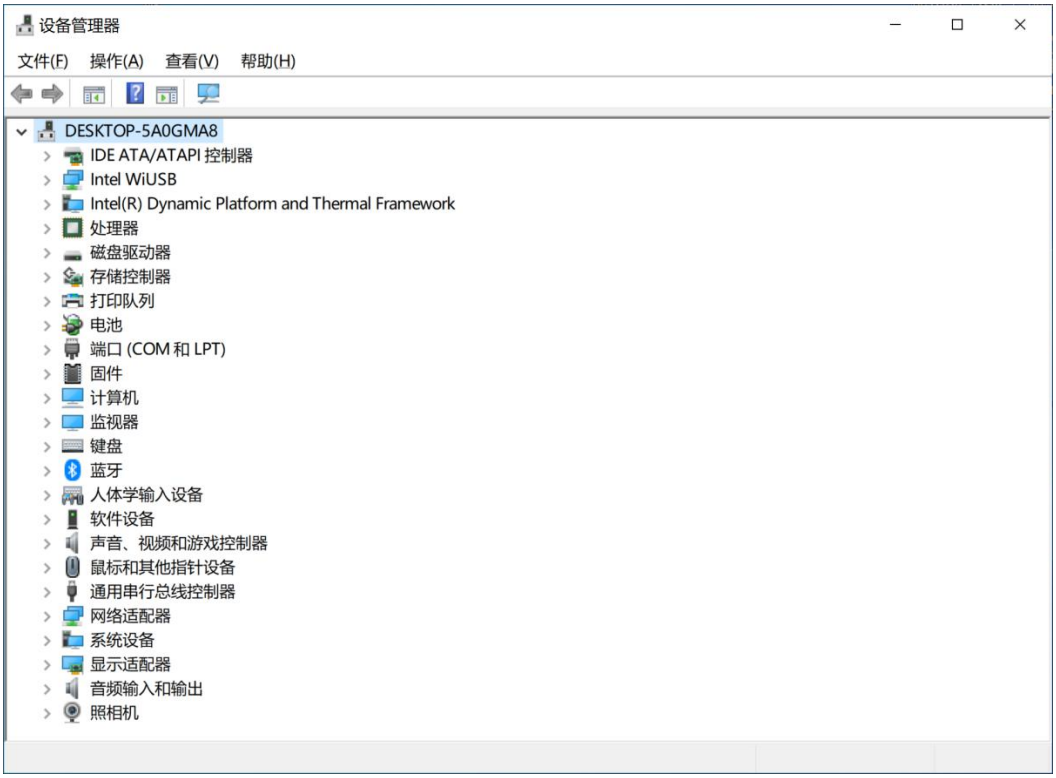


5. 设置参数


ESP32 DOWNLOAD TOOL V3.6.5
SPIDownload
HSPIDownload
RFConfig
GPIOConfig
\Users\crash\Desktop\morpx_main_e.bin
0x10000
SpiFlashConfig
CrystalFreq :
40M
CombineBin
Default
SPI SPEED
40MHz
26.7MHz
20MHz
80MHz
SPI MODE
QIO
QOUT
DIO
DOUT
FASTRD
FLASH SIZE
8Mbit
16Mbit
32Mbit
64Mbit
128Mbit
SpiAutoSet
DoNotChgBin
LOCK SETTINGS
DETECTED INFO
flash vendor:
C8h : GD
flash devID:
4016h
QUAD;32Mbit
crystal:
40 Mhz
Download Panel 1
FINISH
完成
AP: B4E62D9E3D9A STA: B4E62D9E3D99
BT: B4E62D9E3D9B ETHERNET: B4E62D9E3D9C
START
STOP
ERASE
COM:
COM4
BAUD:
115200

注解:

- SPI SPEED:40MHz
- SPI MODE:DIO
- FLASH SIZE:32Mbit
- BAUD:115200
- COM：连接模块的对应 COM 端口, 可以通过 Windows 设备管理器查看主控模块所连接的 COM 端口



6. 点击 `...` 按钮，选择所要更新的固件文件路径，并在前面方框内勾选 √

7. 在 `@` 符号后面输入地址：`0x10000`

注意：不要输入错误地址，也不要按 ERASE 擦除芯片所有内容，否则可能导致传感器芯片内置固件损坏。如有发生，请联系摩图科技售后技术支持进行解决

电话：(0571)8195 8588

邮箱：support@morpx.com

8. 点击左下角 `START` 按钮开始，电脑端显示“等待上电同步”，此时 **连续点击** MoonBot Kit 主控模块 上按钮 B 直至软件开始烧录，烧录时主控右侧 LED 绿灯常亮

9. 等待完成, 当窗口最下方的绿色进度条至最右端，并显示 FINISH 完成字样，则下载完成

CHAPTER 15

MoonBot Kit 扩展形态

除了官方形态之外，用户可以使用 MoonBot Kit 的零件模块配合其他材料搭建各具创意的机器人。

15.1 自定义外壳

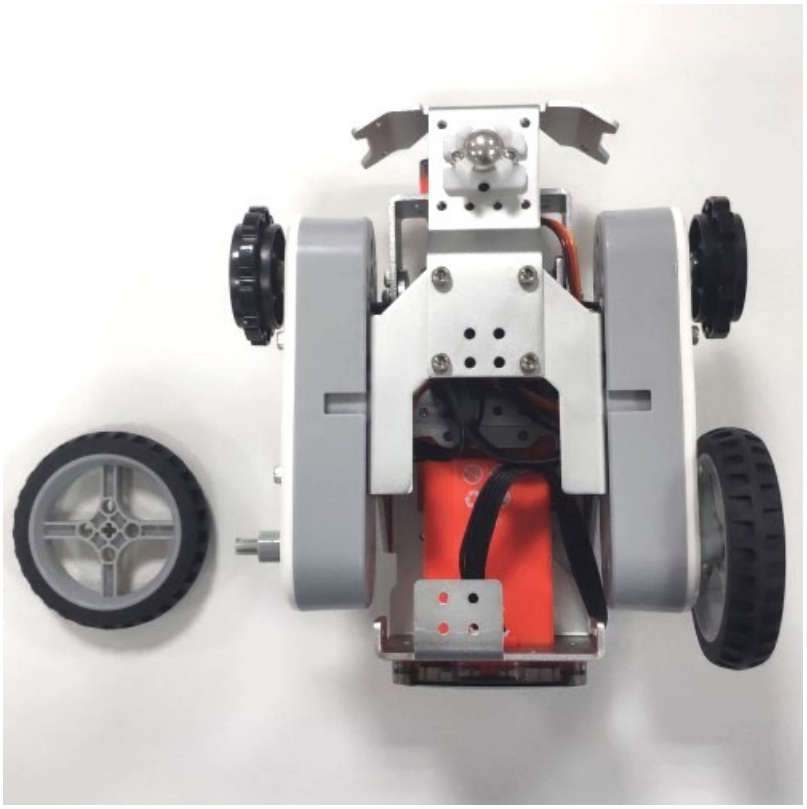
MoonBot 的外形可以自己制作，使用 3D 打印、厚纸板、木板等材料工具制作机器人外观或做内部改造。例如下图创客马老师制作的帅气龙。

15.2 积木小车

MoonBot 套件的硬件模块可以连接常见的乐高积木，替代钣金作为结构件搭建积木机器人。

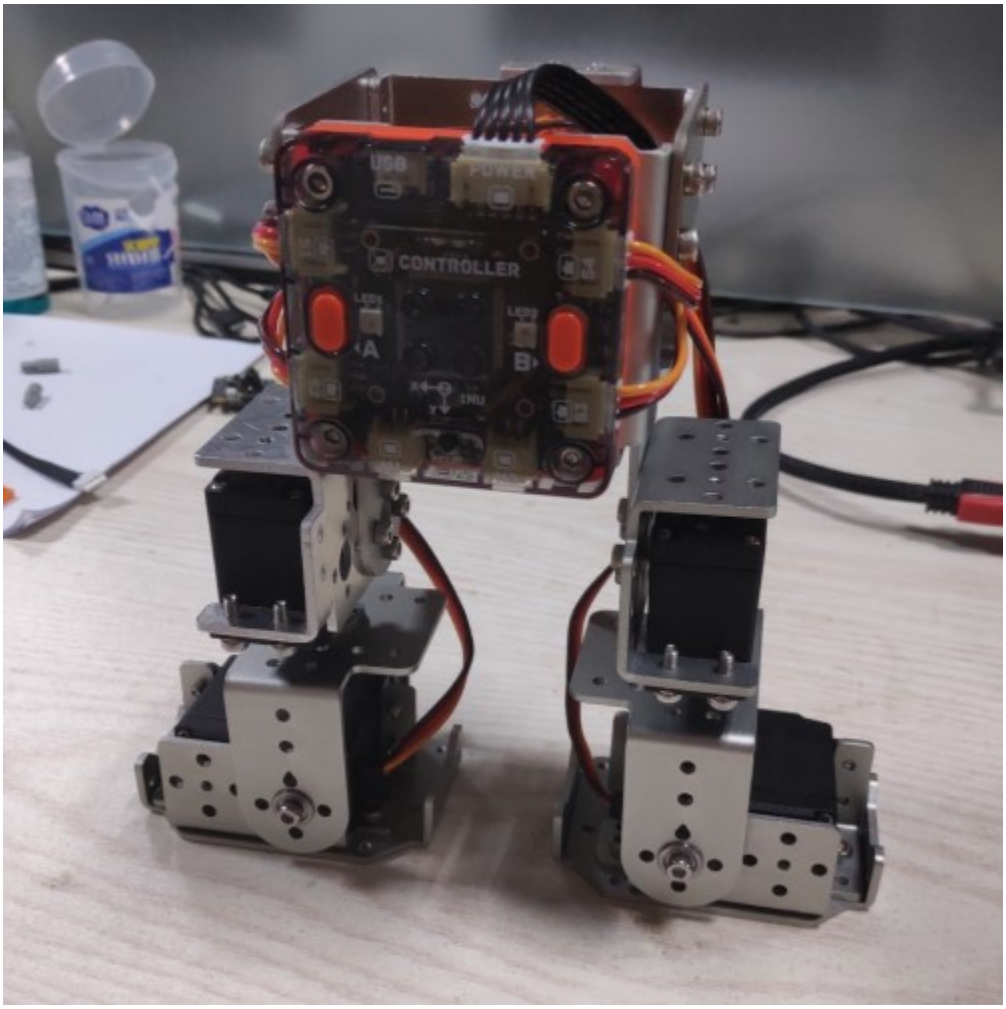
基础连接方式：乐高积木分为传统系列和插销式的 technic 系列，本套件中横向放置的模块使用传统接口，纵向放置的模块使用 techinc 接口。传统接口的模块需转接一块 2X2 的乐高块，就可以和连接到其他积木块；technic 接口的模块则使用插销来连接到其他 technic 块, 如图所示。

将电机模块主动轮的螺丝卸下，移除主动轮，换上一个 TT 马达转乐高的十字转接头，就可以使用乐高轮制作一个传统的两轮小车了。在前面安装一个万向轮以支撑小车，维持水平，如图所示。换成两轮小车后速度可以达到原本履带小车的两倍以上。



15.3 双足机器人

MoonBot 主控最多可以连接 4 个舵机，配合钣金件可搭建一台双足跳舞机器人。


USB
POWER
CONTROLLER
LED1
LED2
A
B
IMU

CHAPTER 16

MoonBot Kit 资源

16.1 技术资料

在以下网址可获取最新的 MoonBot Kit 技术资料：

官网技术支持：http://mai.morpx.com/page.php?a=moonbot-kit

GitHub：https://github.com/mu-opensource/

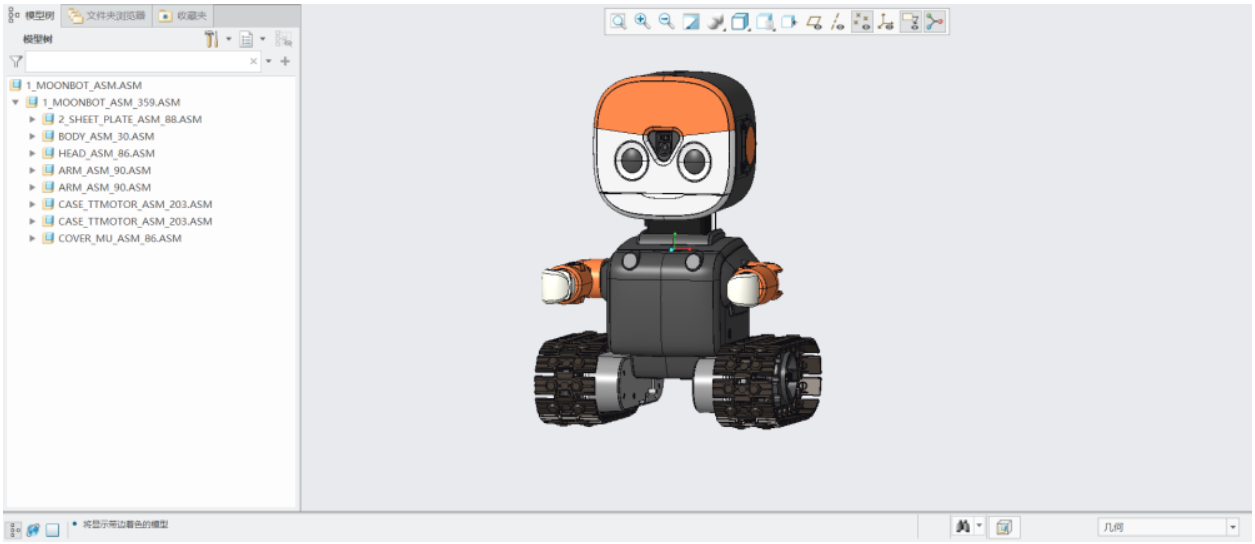
16.2 MoonBot 课件

在网盘下载 MoonBot 的课件：链接，提取码：wq8n

16.3 3D 装配图

在此下载 MoonBot Kit 的三形态 3D 图：

MoonBot Kit 3D 装配图



STP 文件为通用 3D 格式，可用主流 3D 软件如 solidworks，CREO 等打开。模型可用于查看 MoonBot Kit 的装配细节，测量相应尺寸，Keyshot 模型渲染等。注意该模型内塑料件和钣金件均针对开模设计，不建议用于 FDM 3D 打印。

16.4 平台链接

MoonBot Kit 兼容 Arduino 开源平台，查看其相关网站来学习基础知识。

Mixly:

Mixly 官网 http://mixly.org/

Mixly 帮助文档 https://mixly.readthedocs.io/zh_CN/latest/contents.html

Arduino：

Arduino 官网 https://www.arduino.cc/

Arduino 中文社区 https://www.arduino.cn/forum.php

DF 创客社区 http://mc.dfrobot.com.cn/

极客工坊 https://www.geek-workshop.com/forum.php

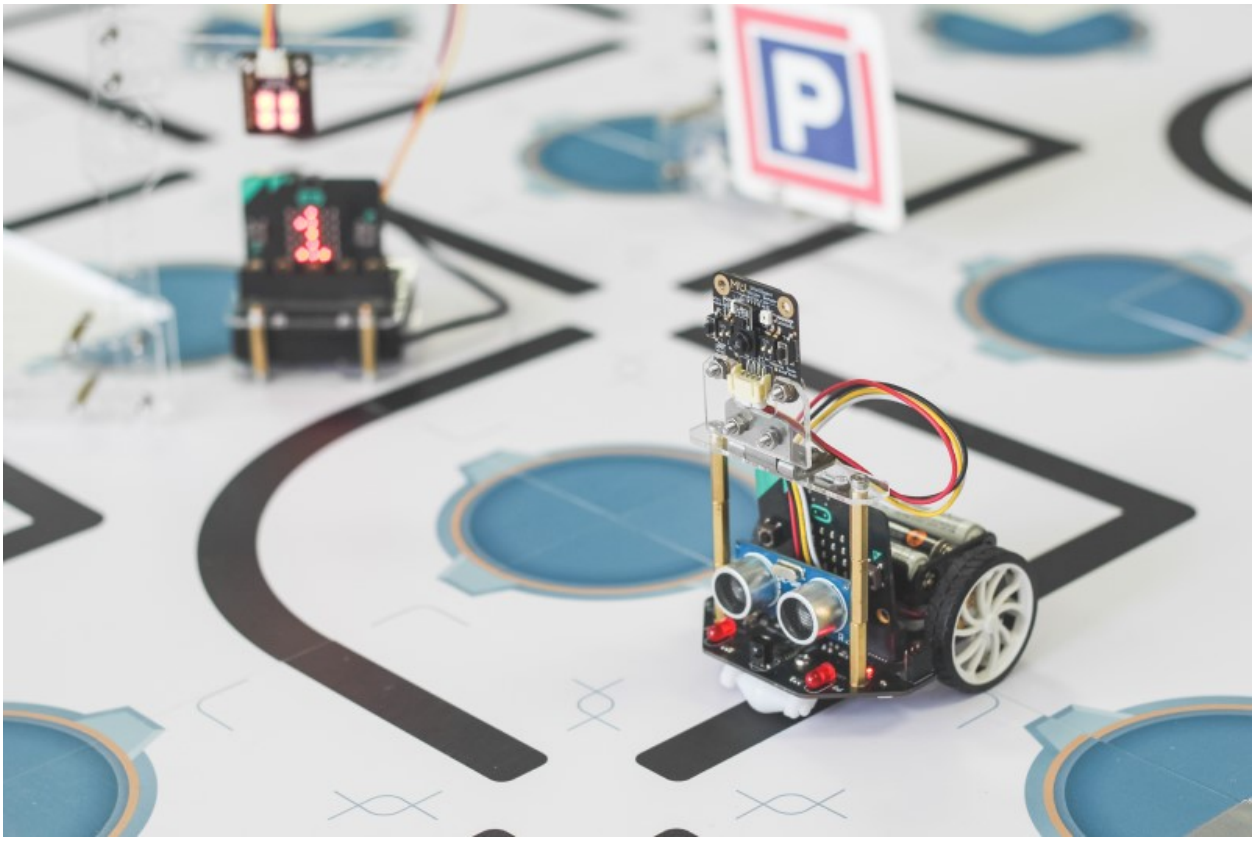
CHAPTER 17

MU 无人驾驶套件简介

MU 无人驾驶套件是基于 MU 视觉识别技术而延伸的学习套件。套件包含了一台自动驾驶小车和模块化的路障、地图。

自动驾驶小车以 Micro:bit 作为主控，MU 作为眼睛，通过红外、视觉、超声波等反馈引导小车自动驾驶；另一块 Micro:bit 作为路障主控，通过交通灯、舵机控制等实现动态的交通指示，也可和小车进行无线通讯；模块化的地图可拼搭成不同路线形式，模拟各类道路情况。

无人驾驶创客先行，快用 MU 无人驾驶套件体验未来交通吧。



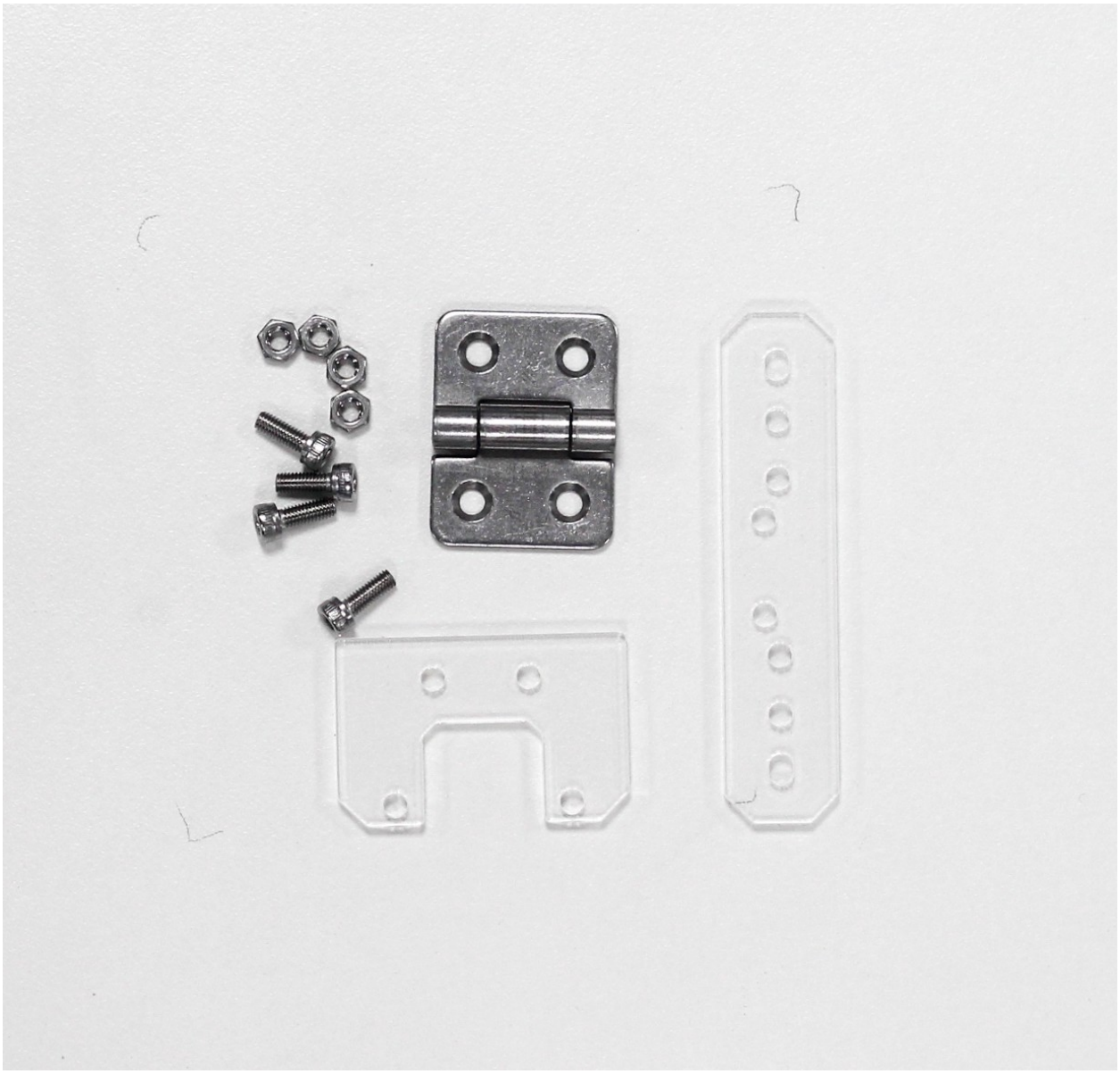
CHAPTER 18

MU 无人驾驶套件拼装

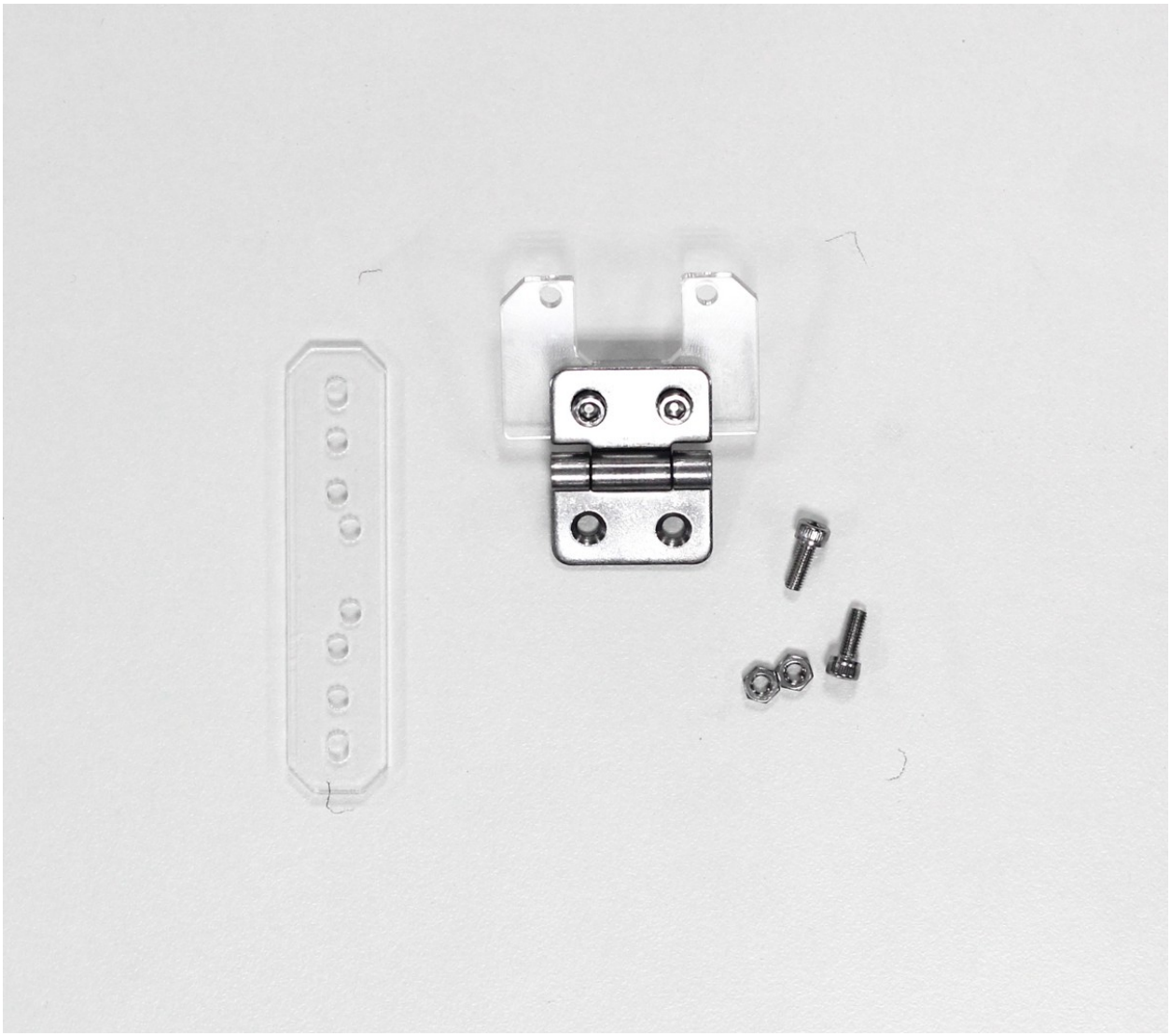
18.1 麦昆小车扩展

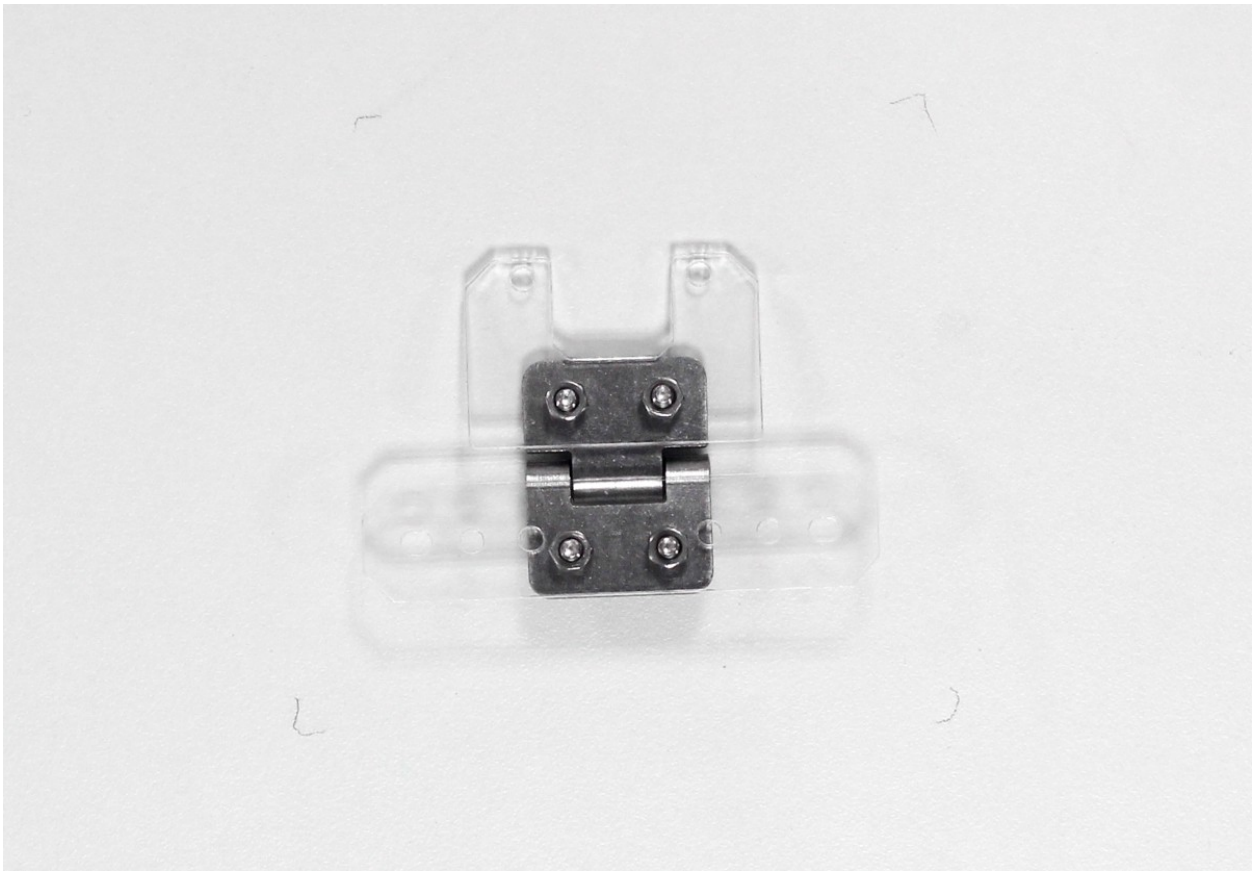
无人驾驶小车需要在麦昆小车上安装 MU3 视觉传感器，用于识别线和交通指示牌。以下为参考安装教程，用户也可根据自身需要发挥创意。

1. 准备金属转轴组件的相关零件。

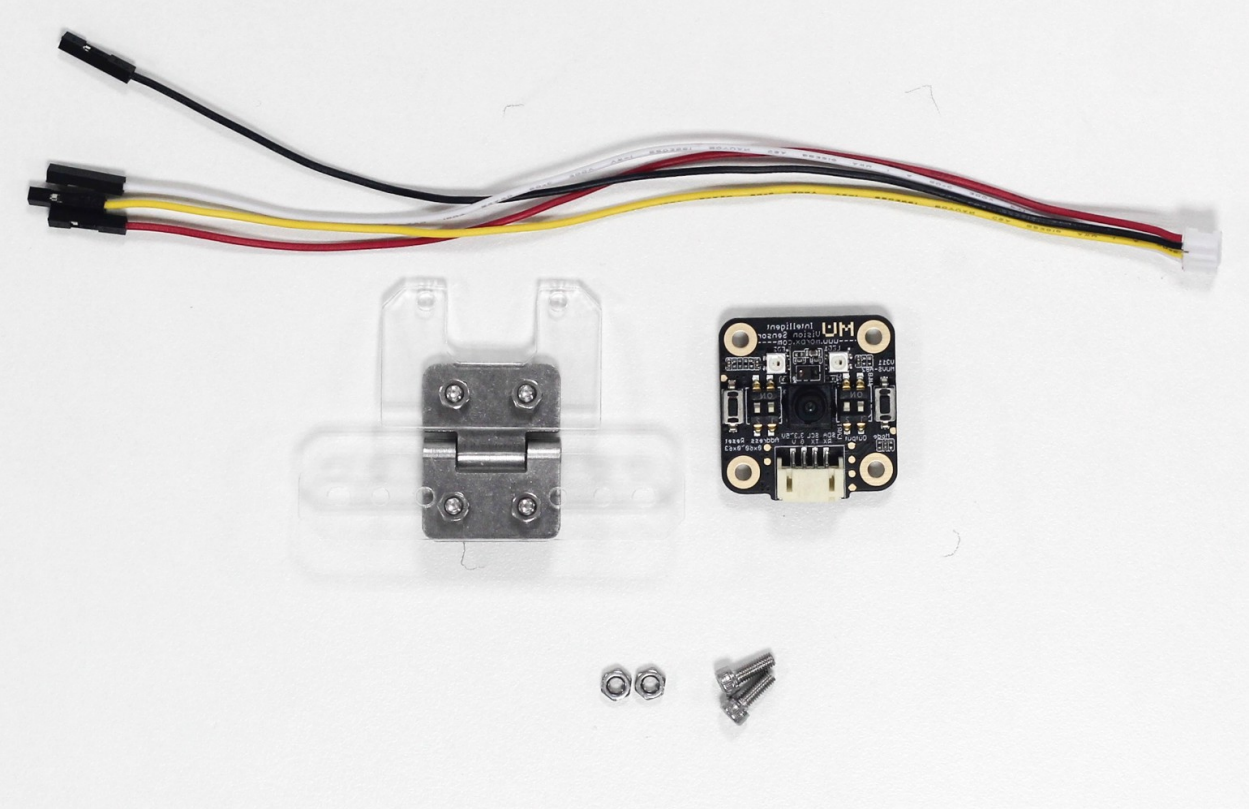


2. 连接金属折页和亚克力支架。

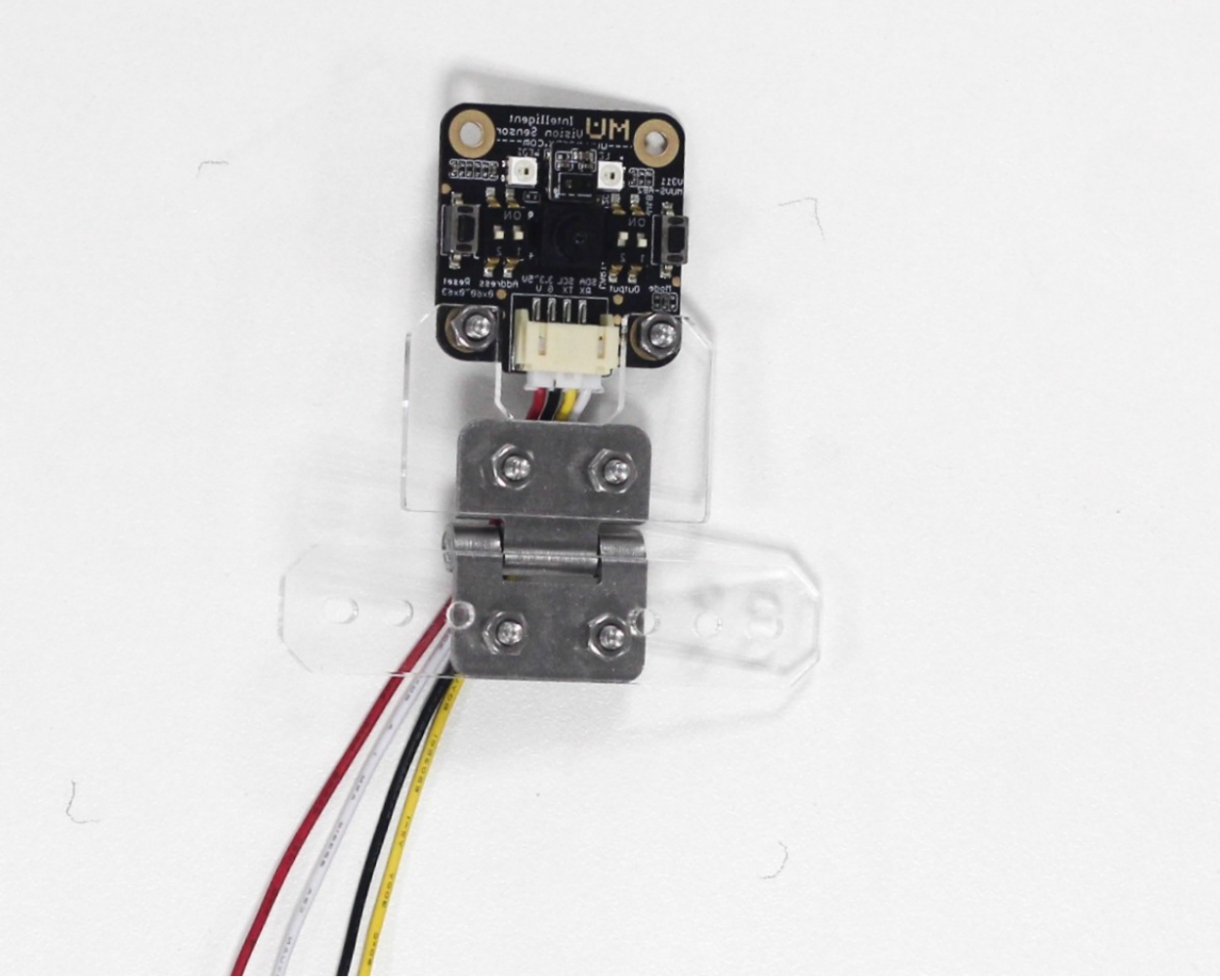




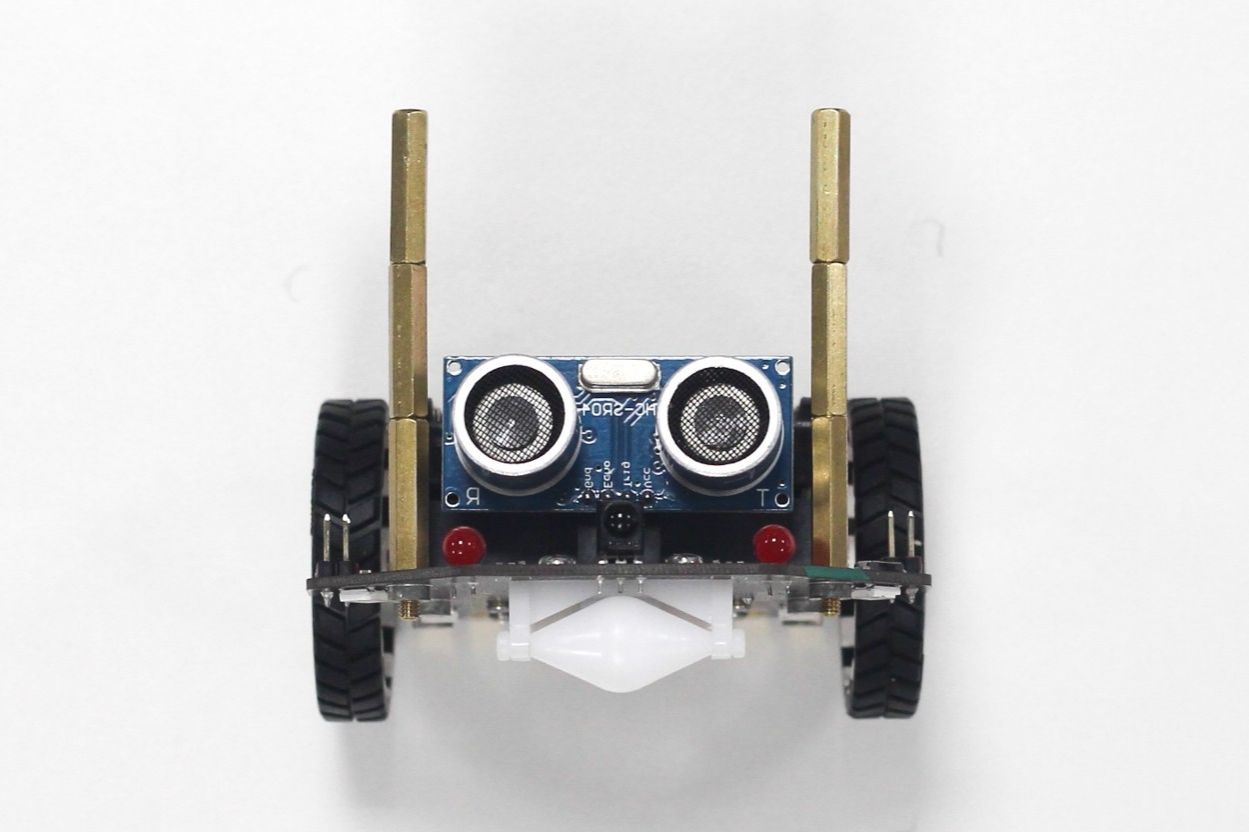
3. 准备 MU 视觉传感器和金属转轴组件。



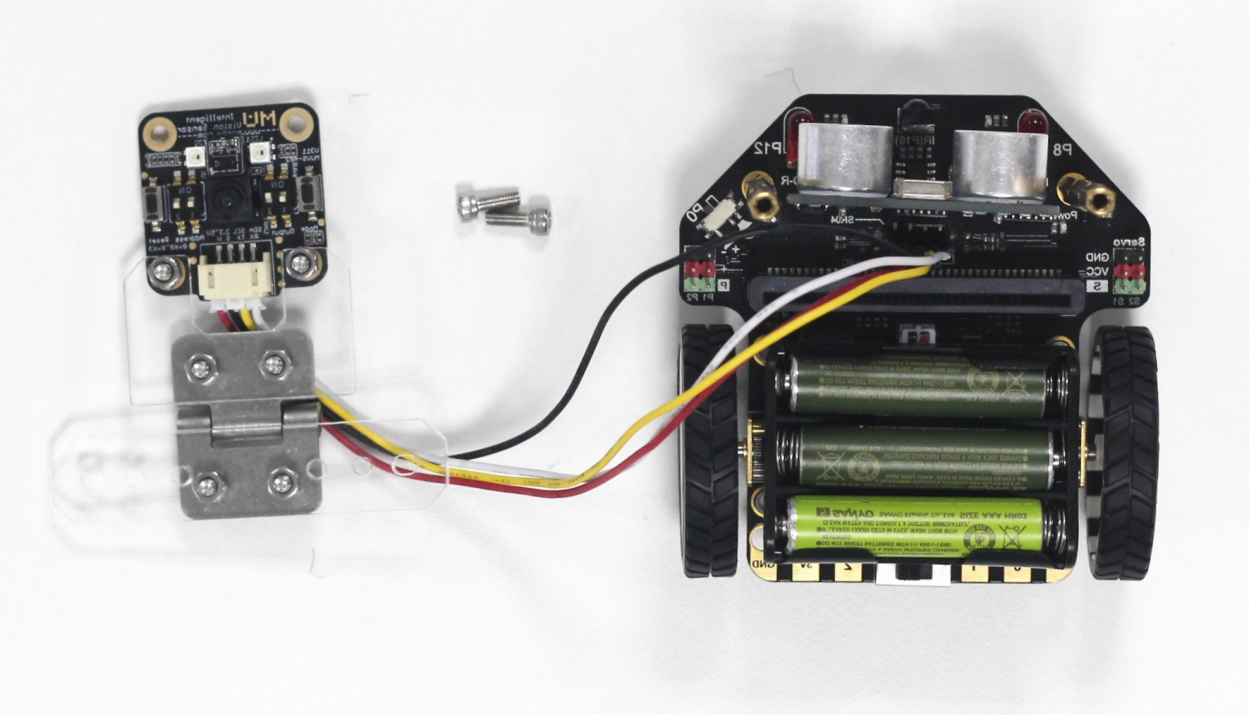
4. 将 MU 视觉传感器连接杜邦线的白色插口，再安装到金属转轴上。注意视觉传感器安装方向。



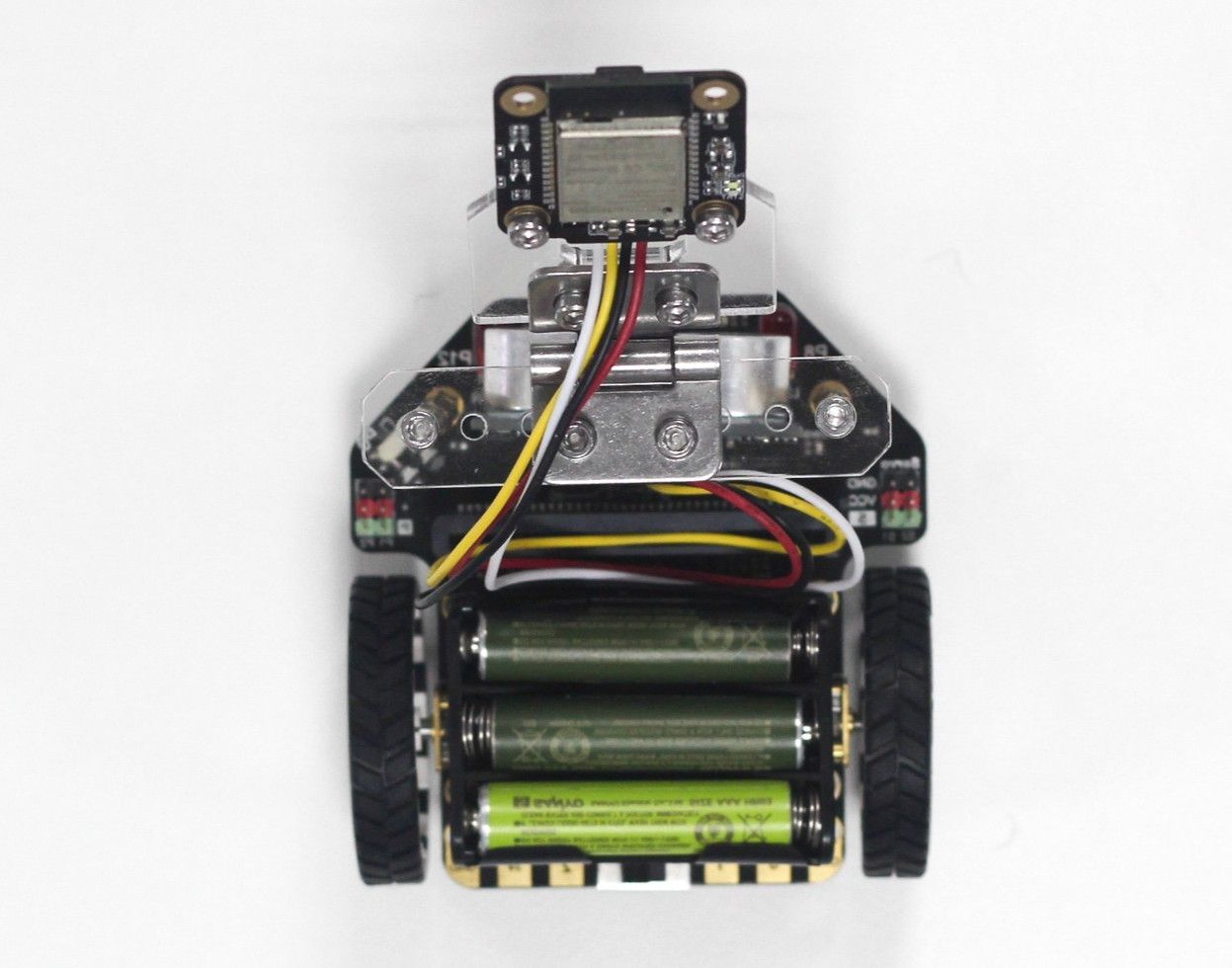
5. 麦昆小车两侧螺纹孔安装铜柱。通过铜柱数量堆叠可调节传感器高度，适用于不同场景。如 2 节铜柱可用于前方交通卡片识别，3 节铜柱可用于地面路识别。



6. 麦昆小车与 MU 视觉传感器连线，顺序依次为 +(红色)，-(黑色)，C(黄色)，D(白色)。



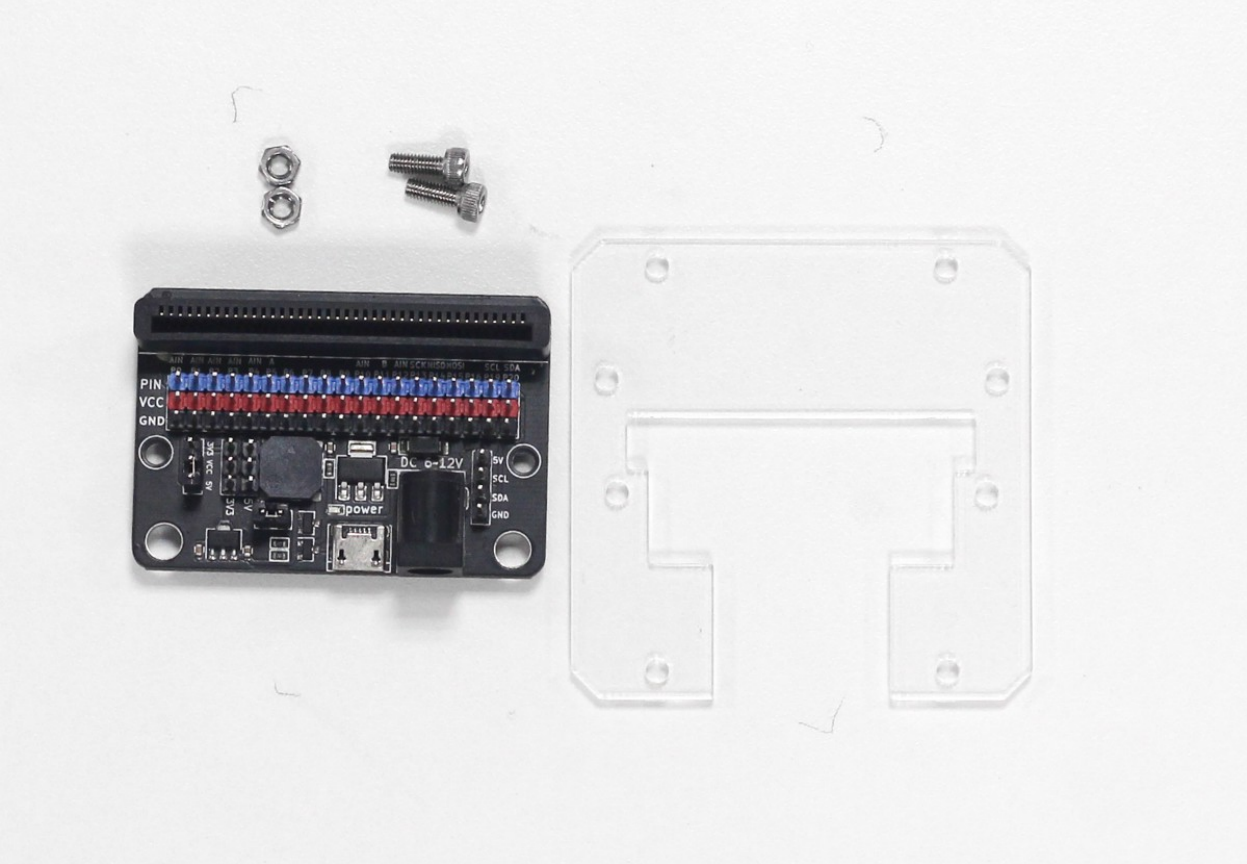
7. 将亚克力支架固定在铜柱顶部，调节金属转轴使视觉传感器可以转到不同角度。

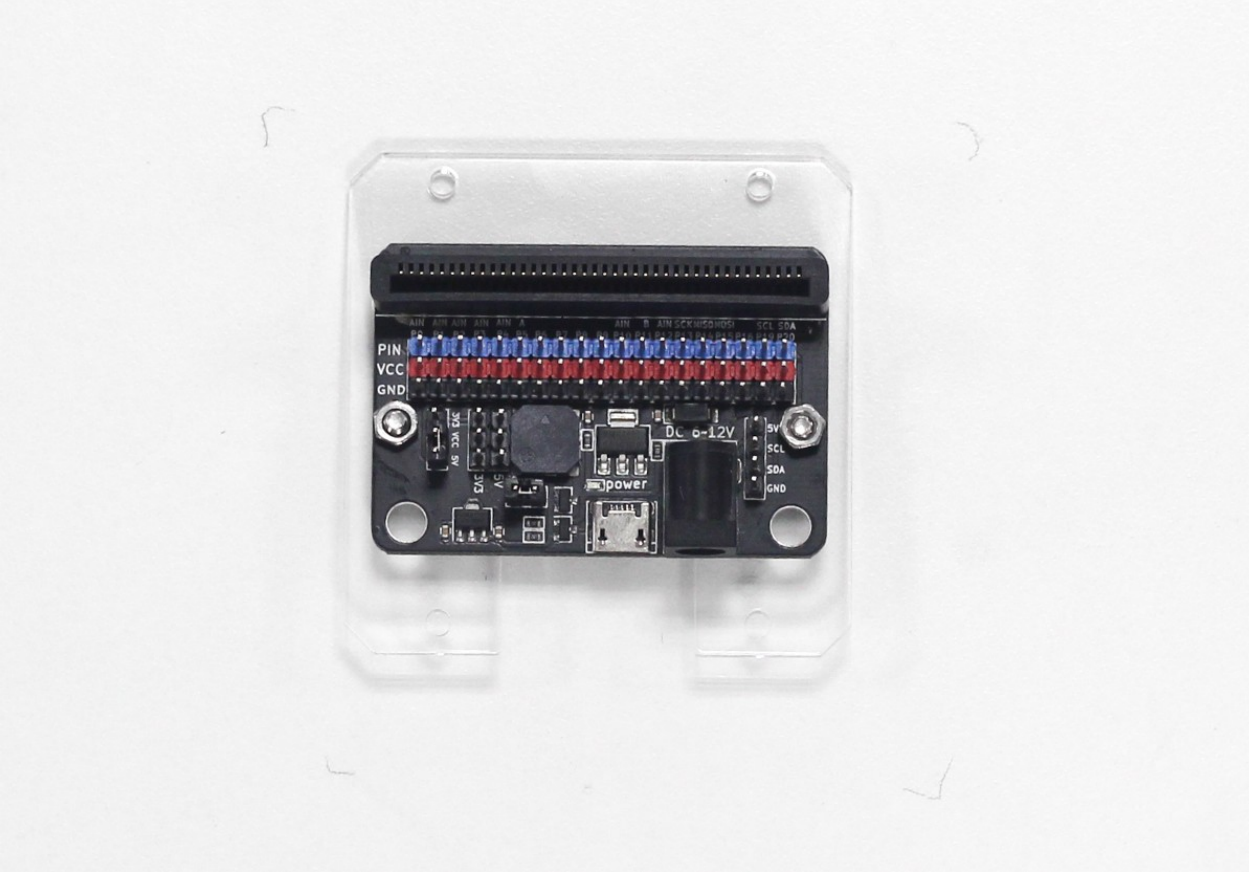


18.2 交通装置拼装

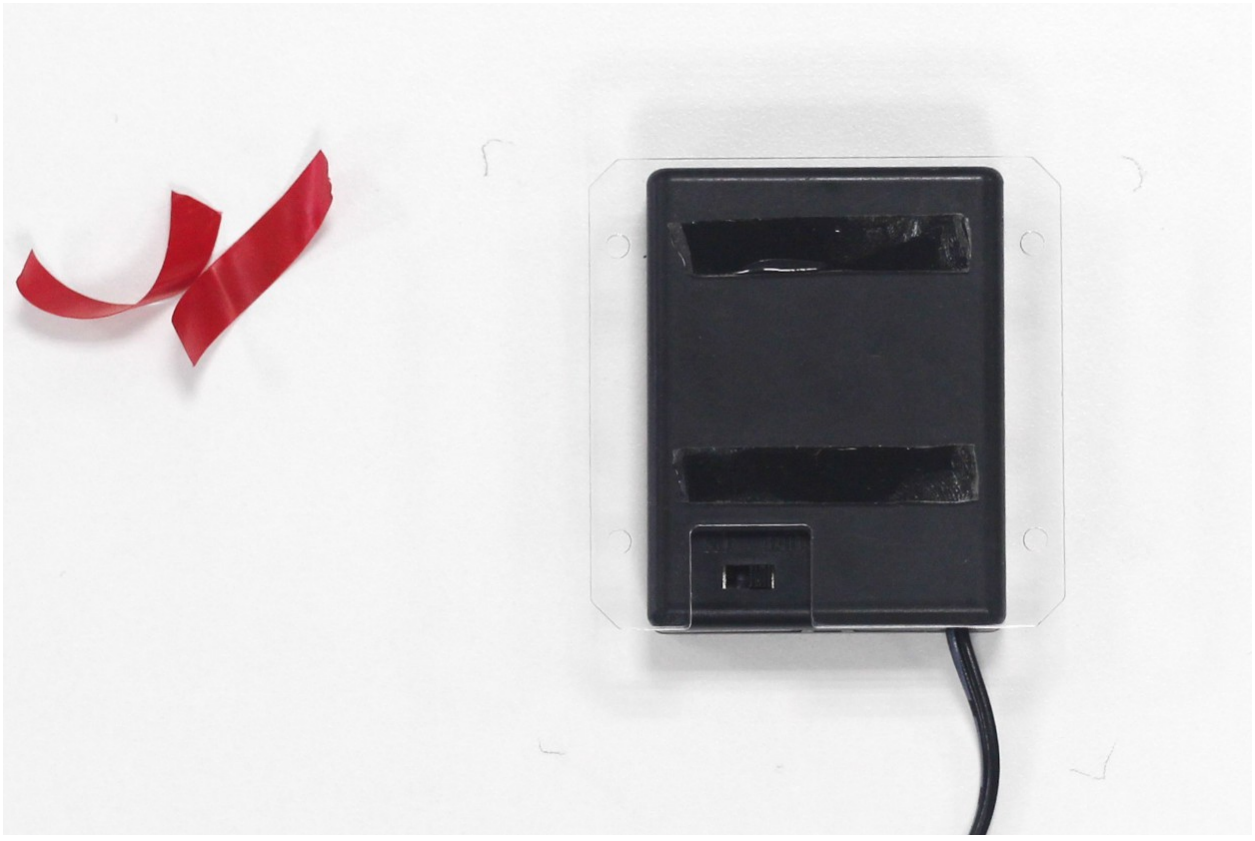
18.2.1 交通控制器

1. 准备 micro:bit 扩展板和亚克力顶板，用螺丝螺母连接。注意螺母在上。

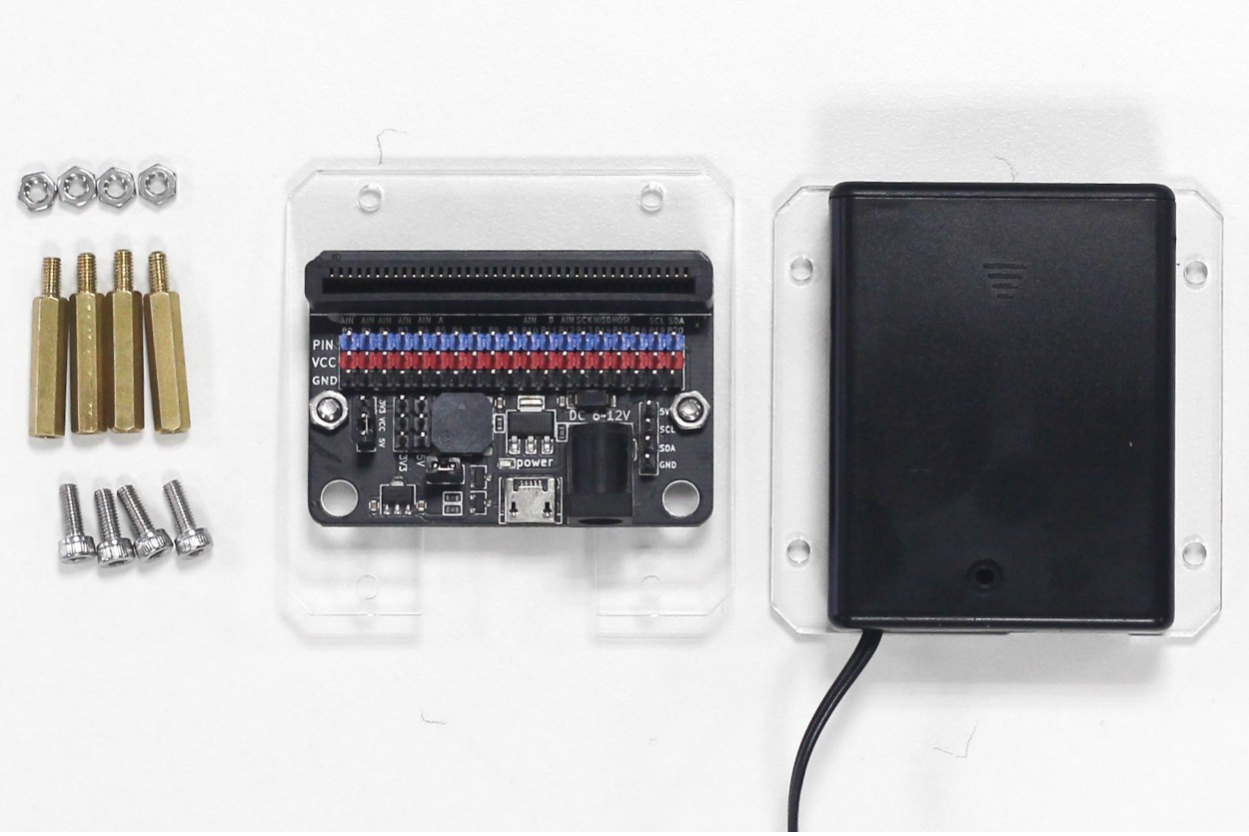

PIN
VCC
GND
DC 6-12V
power
5V
SCL
SDA
GND

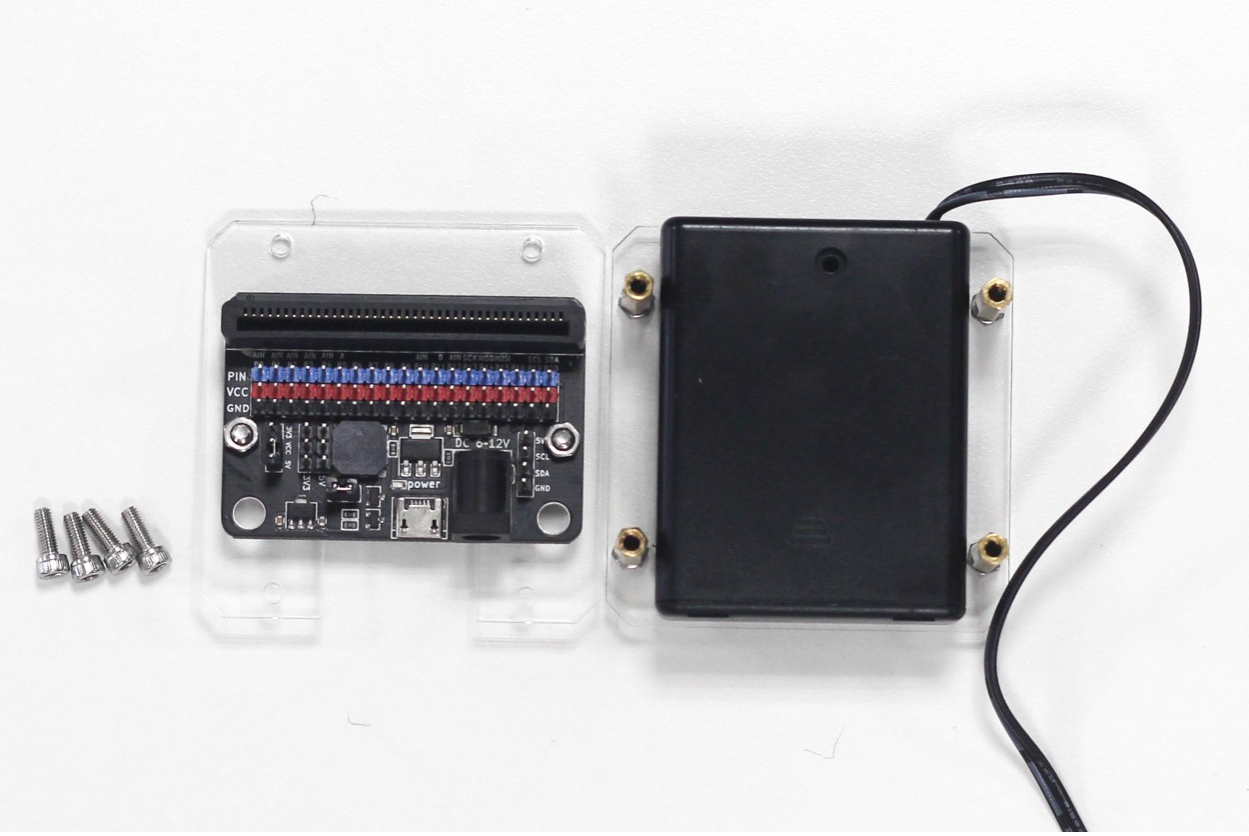


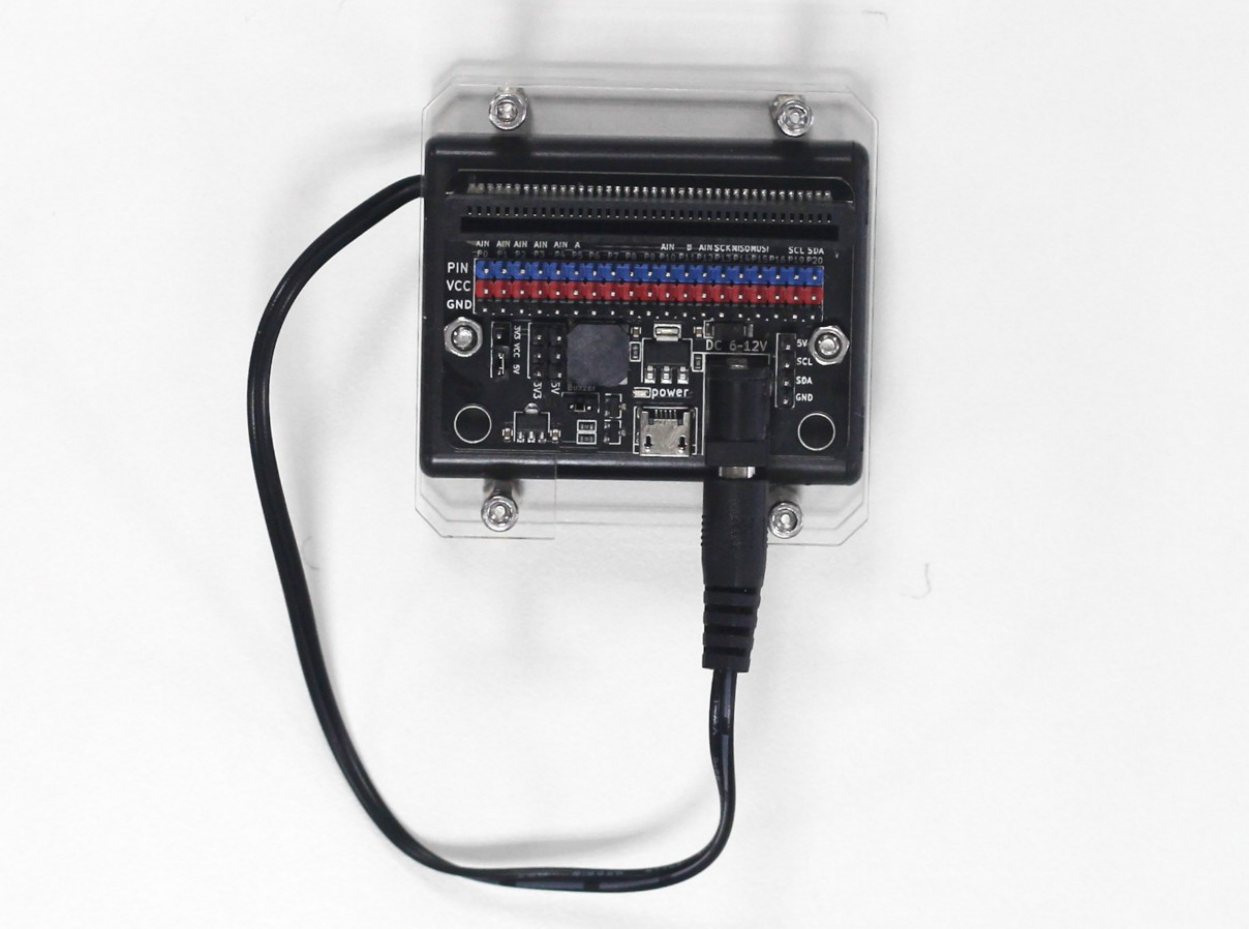
2. 准备 7 号电池盒和亚克力底板，将电池盒双面胶揭开，粘贴到亚克力底板。注意电池盒居中，开关在缺口处。



3. 电池盒装上 4 节 7 号电池，将顶板和底板用铜柱连接，电池盒插口连接到扩展板的接口上。

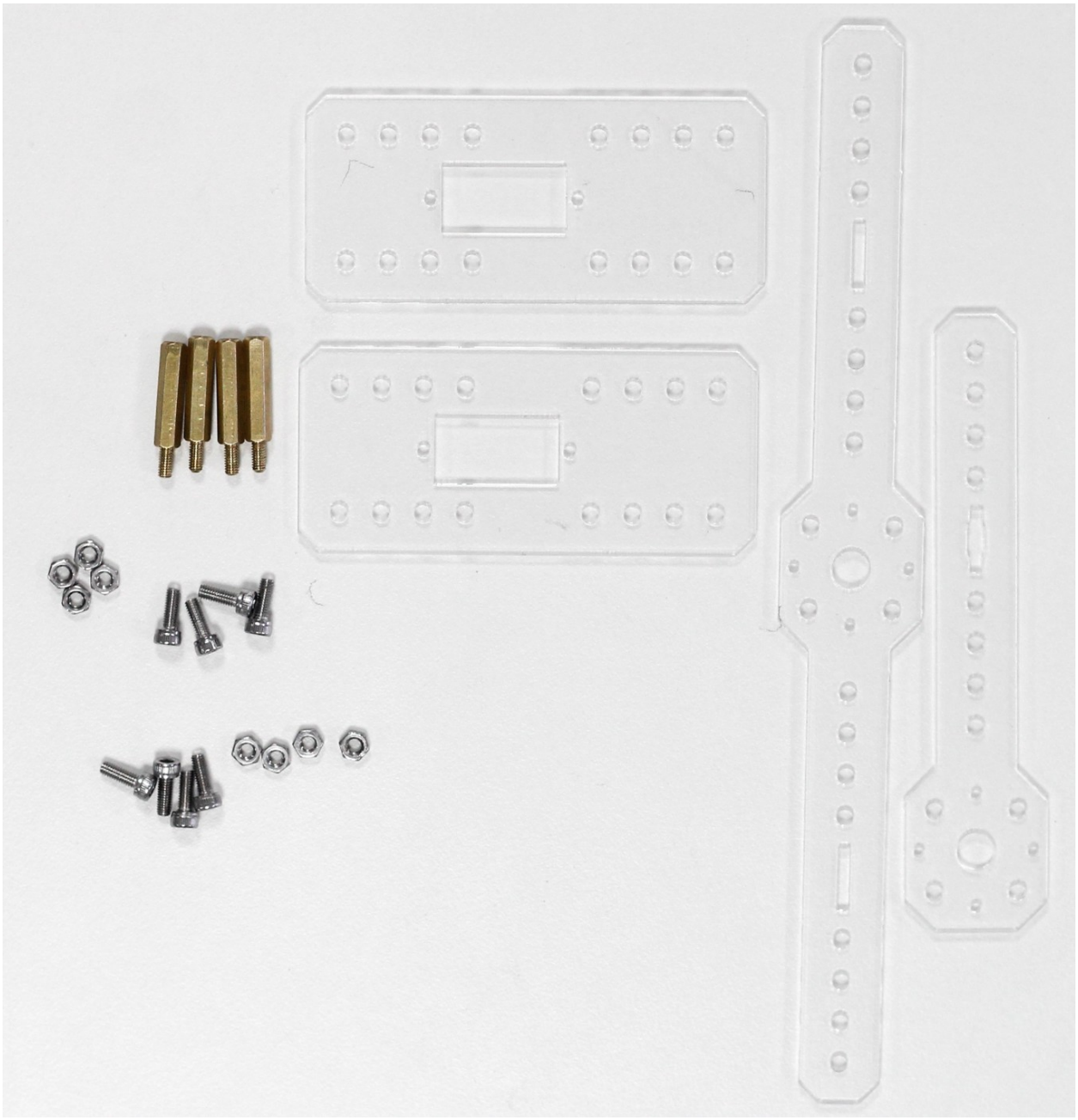


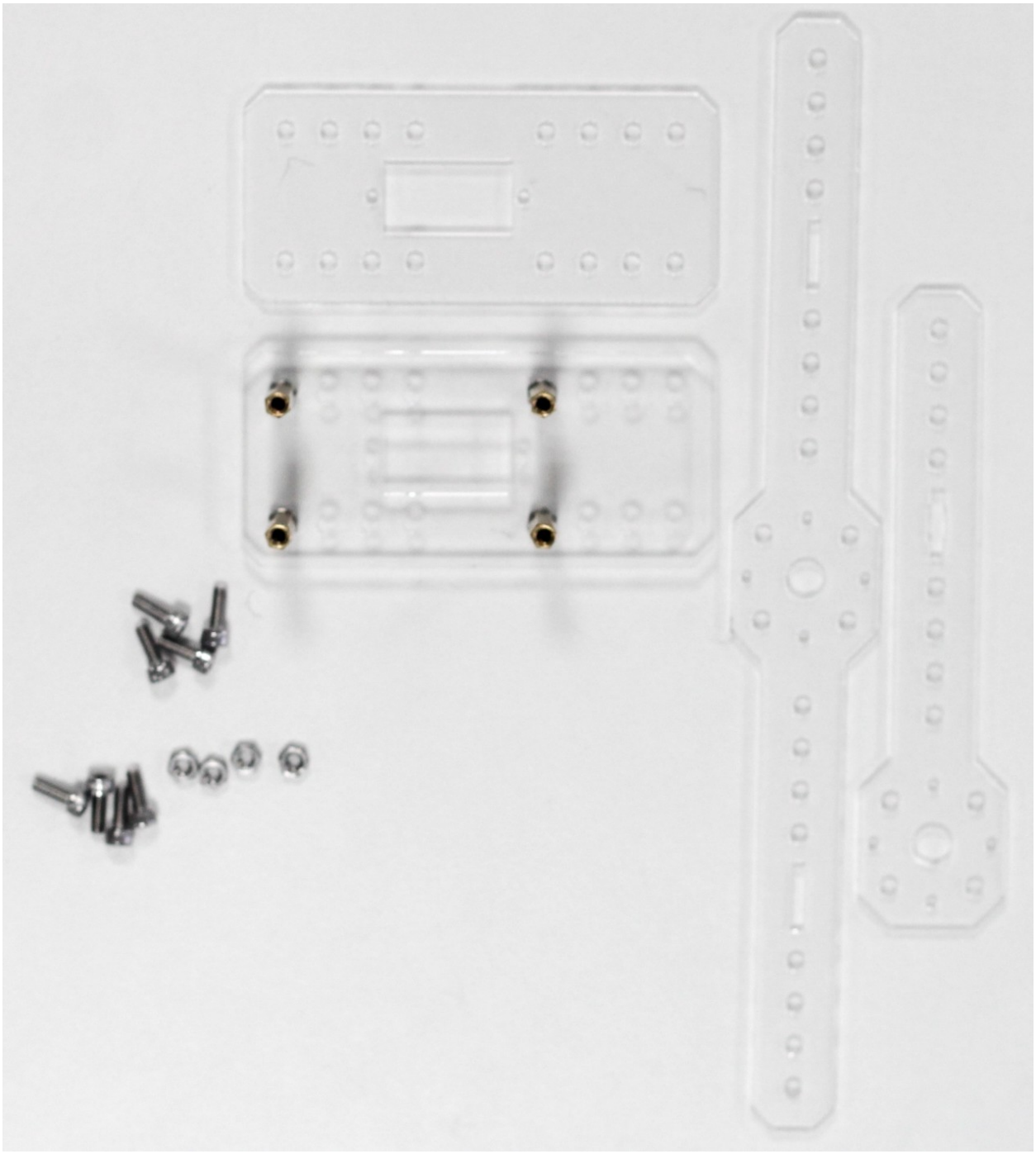

PIN
VCC
GND
DC 6-12V
power
5V
SCL
SDA
GND



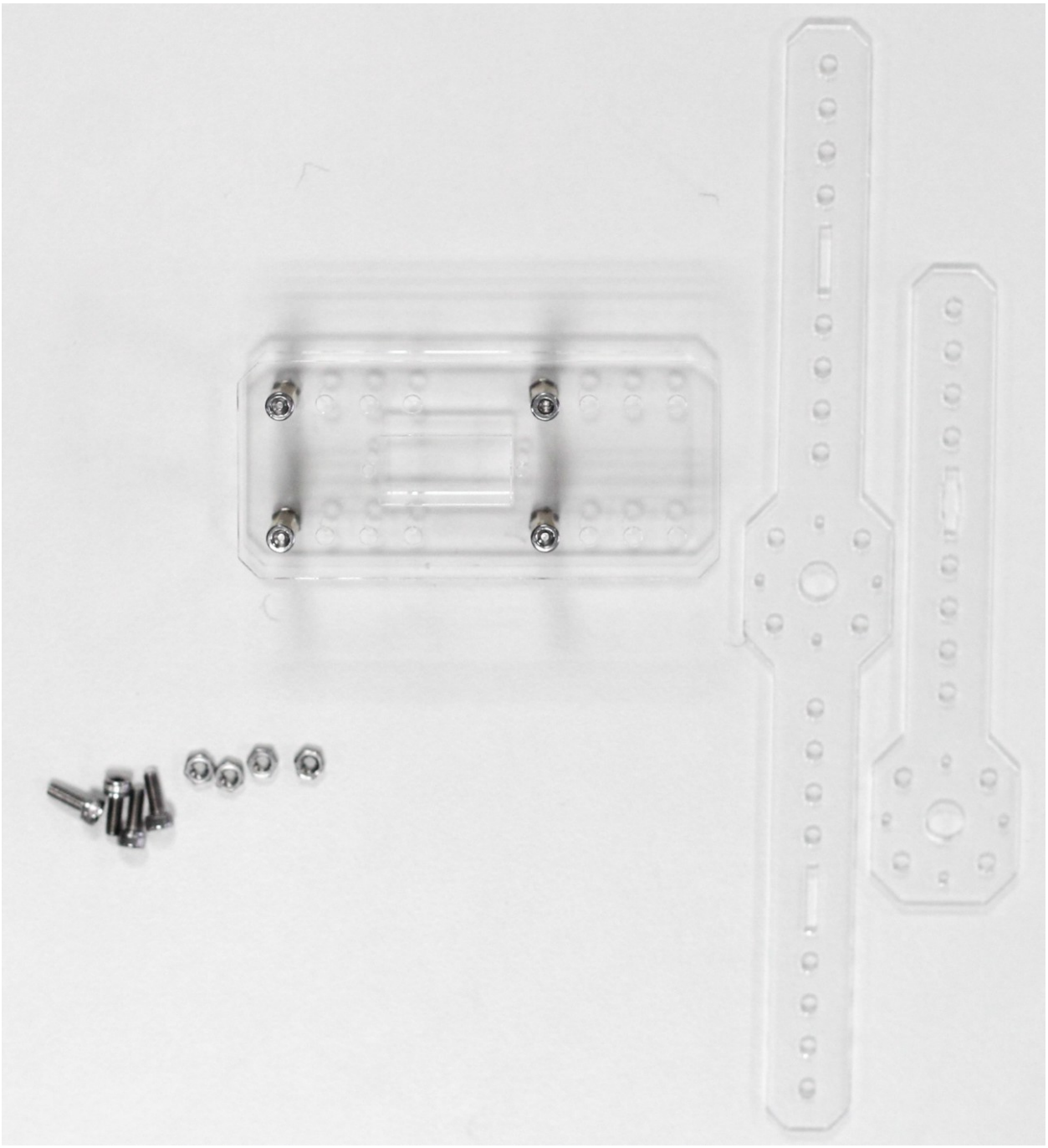
18.2.2 固定指示牌 & 红绿灯

1. 准备亚克力板、铜柱和螺丝螺母，将 4 个铜柱安装到亚克力板。

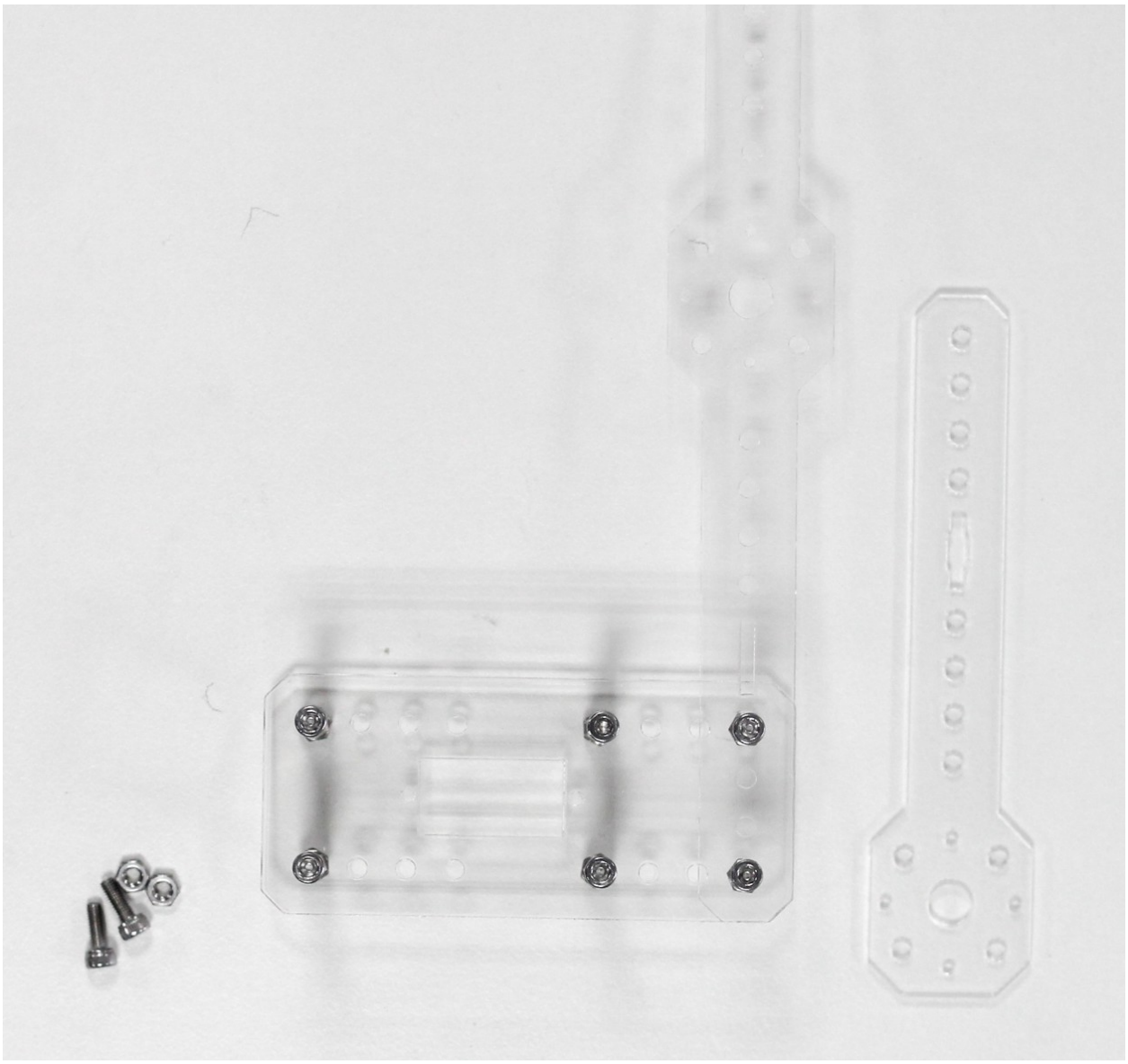




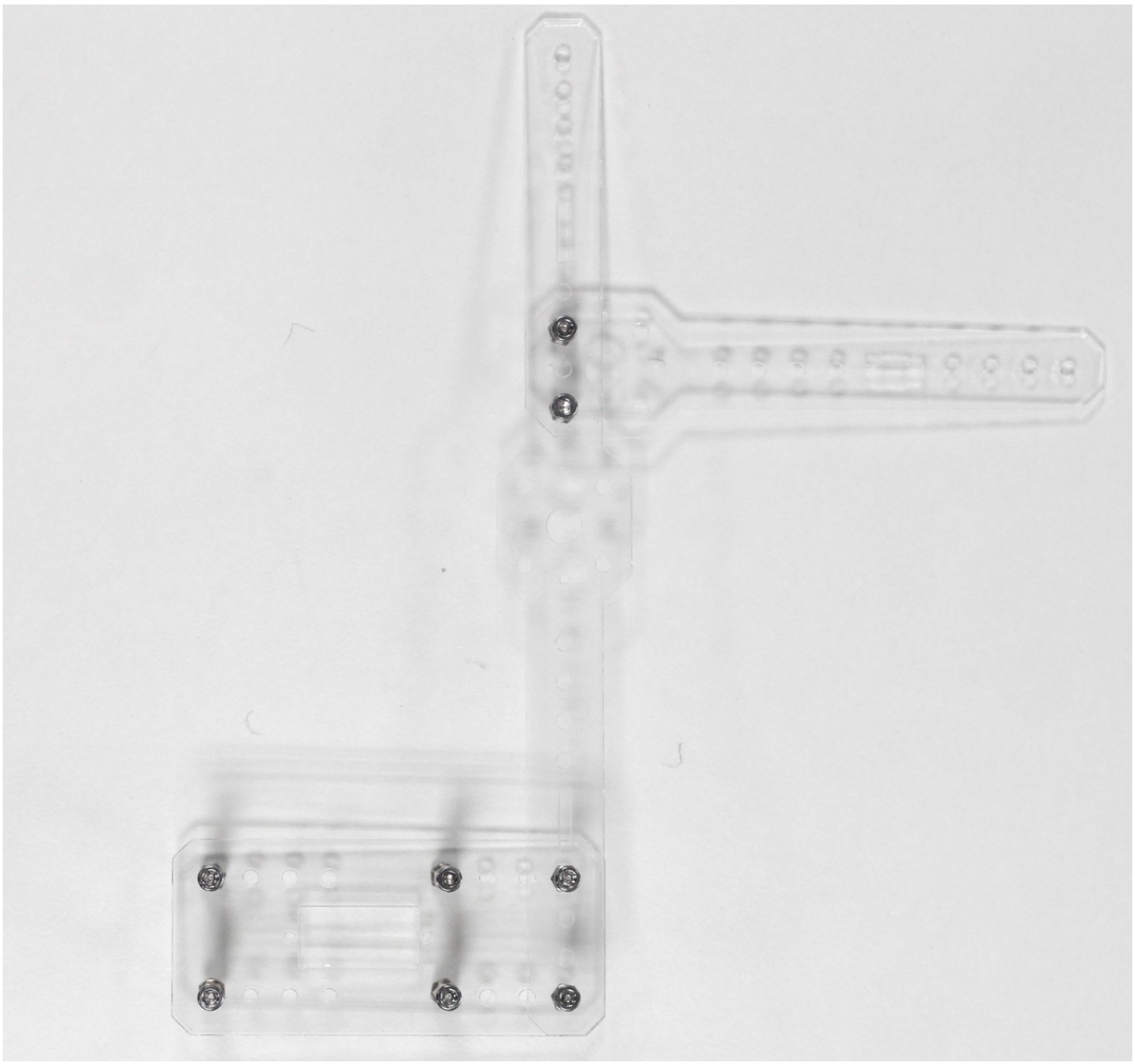
2. 铜柱顶部安装另一块亚克力板。



3. 将亚克力长臂安装到顶部亚克力板的一侧。

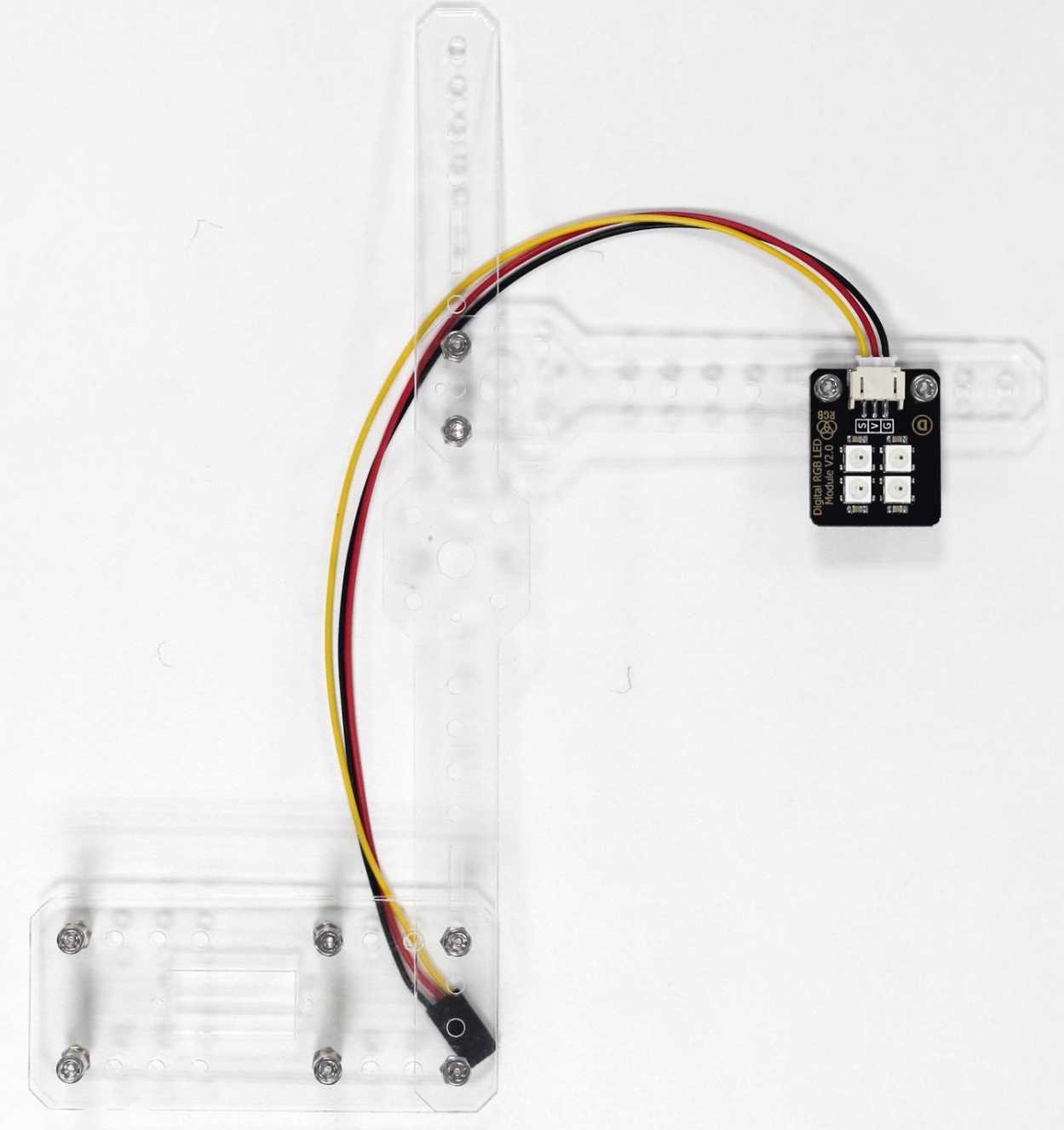


4. 将亚克力短臂安装到长臂上。安装在不同的孔位可调节交通卡片的高度。



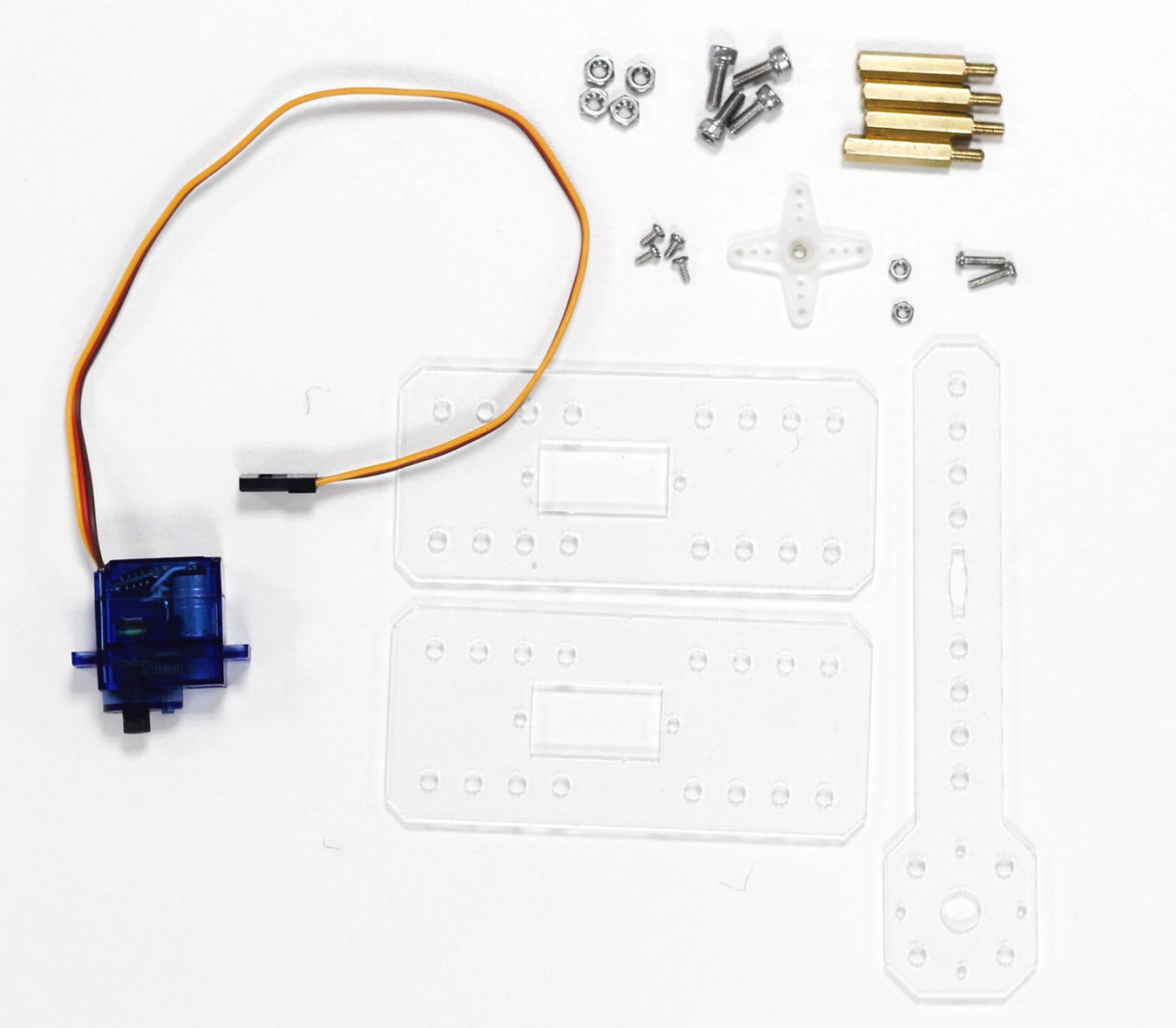
5. 在亚克力短臂上安装卡片或红绿灯

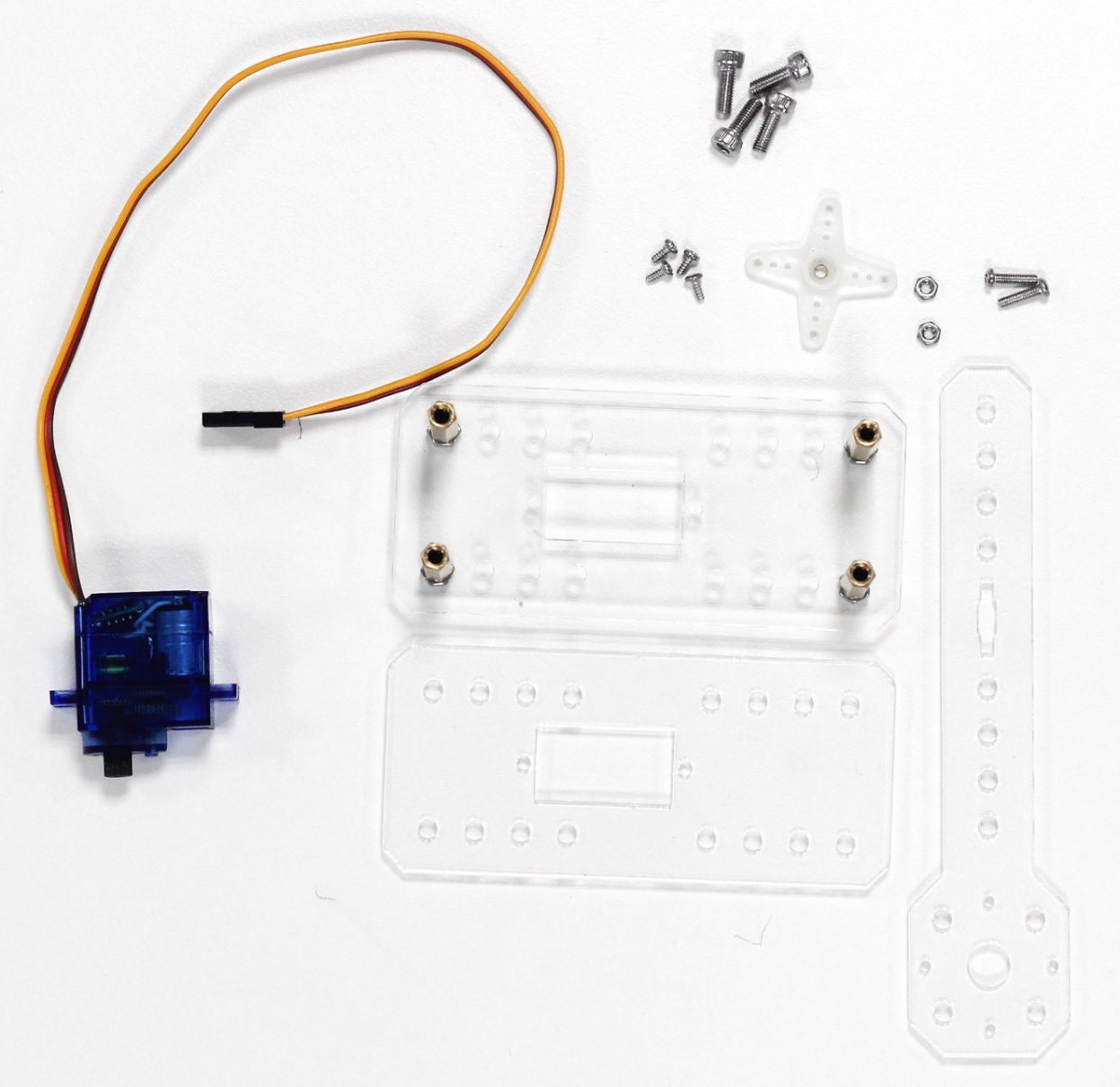

Traffic:Right
Label 3


Digital RGB LED
Module V2.0

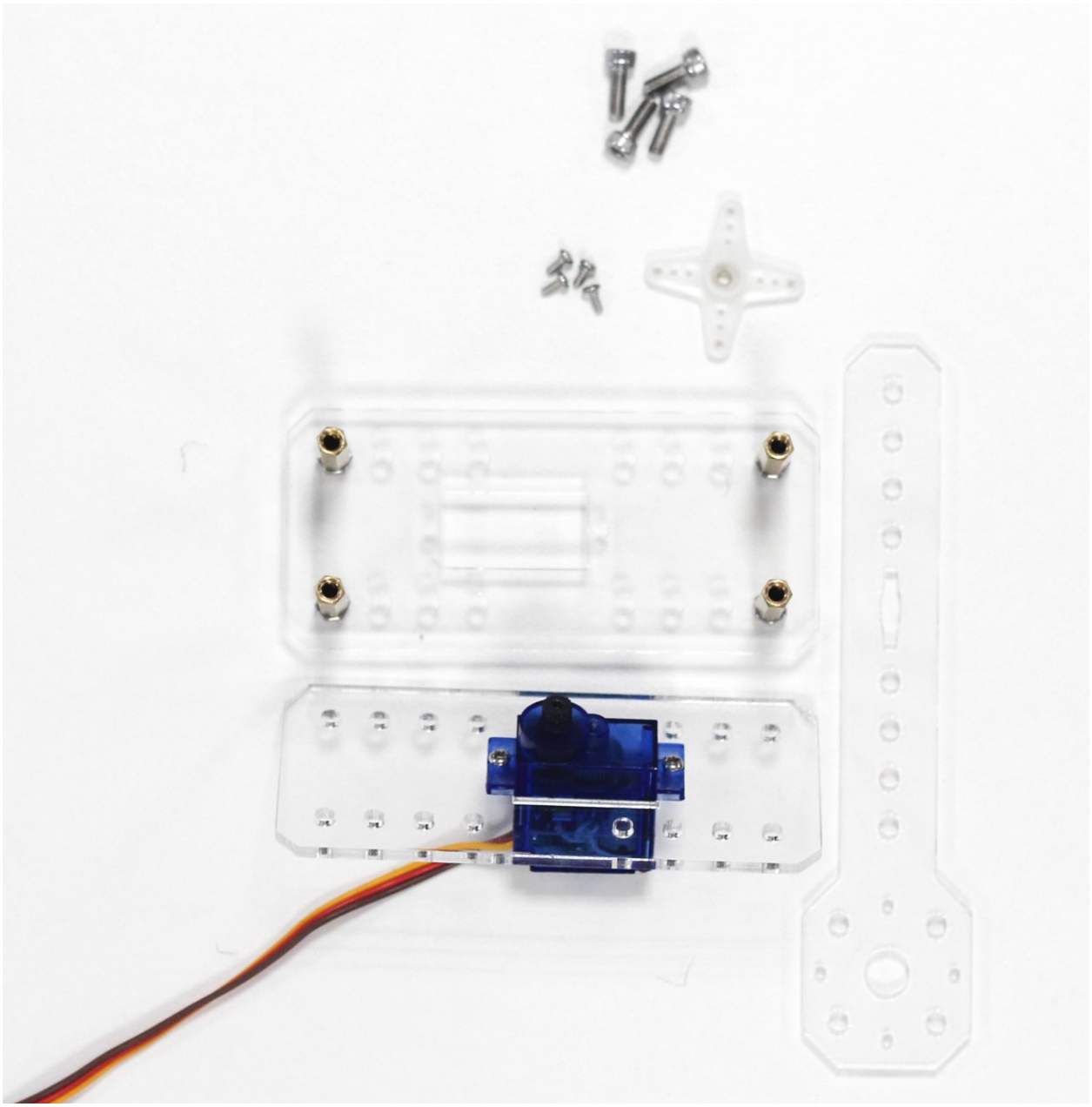
18.2.3 可变指示牌

1. 准备舵机组件，亚克力板，铜柱和螺丝螺母，将四颗铜柱用螺母固定到亚克力底板。

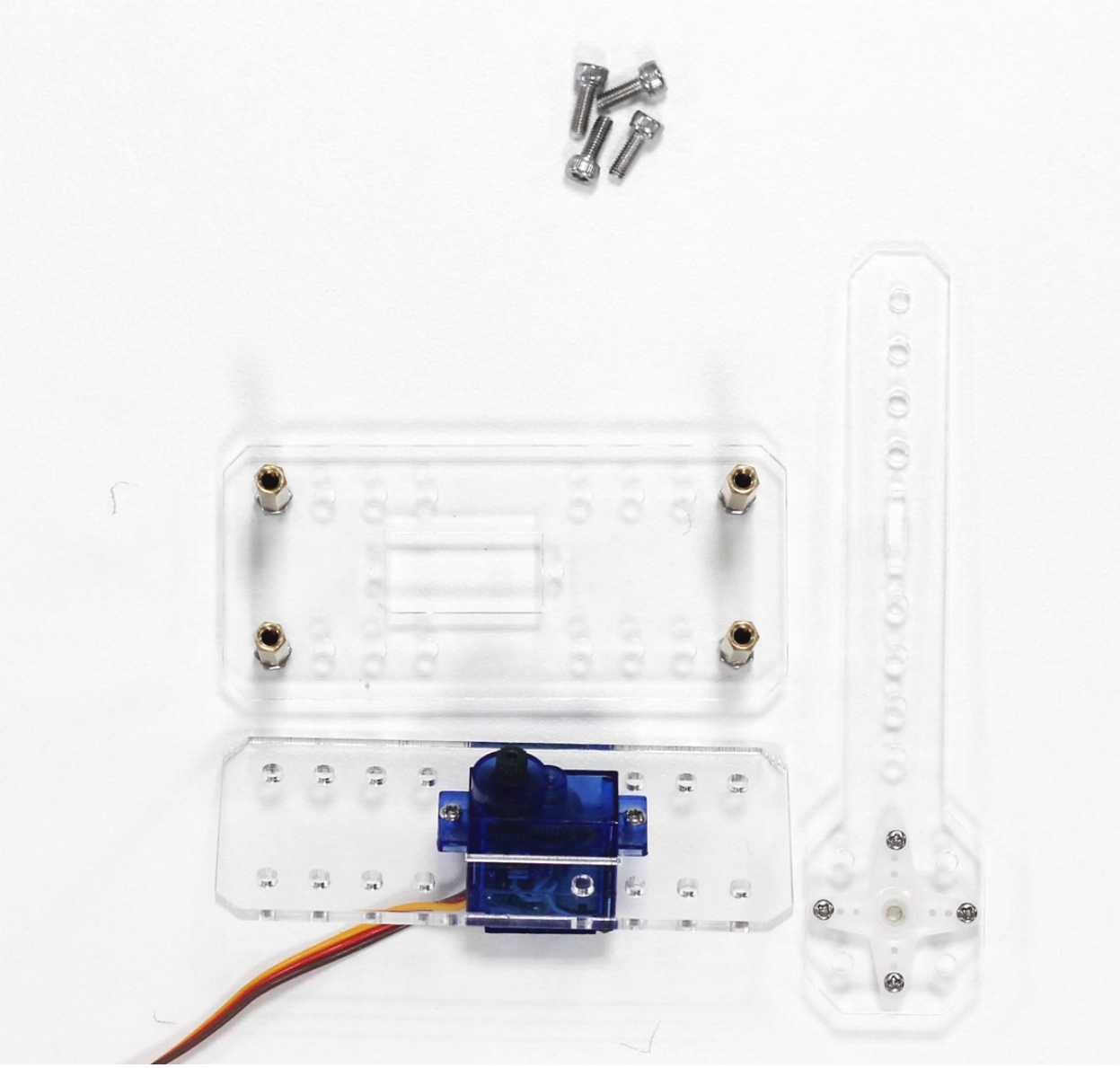




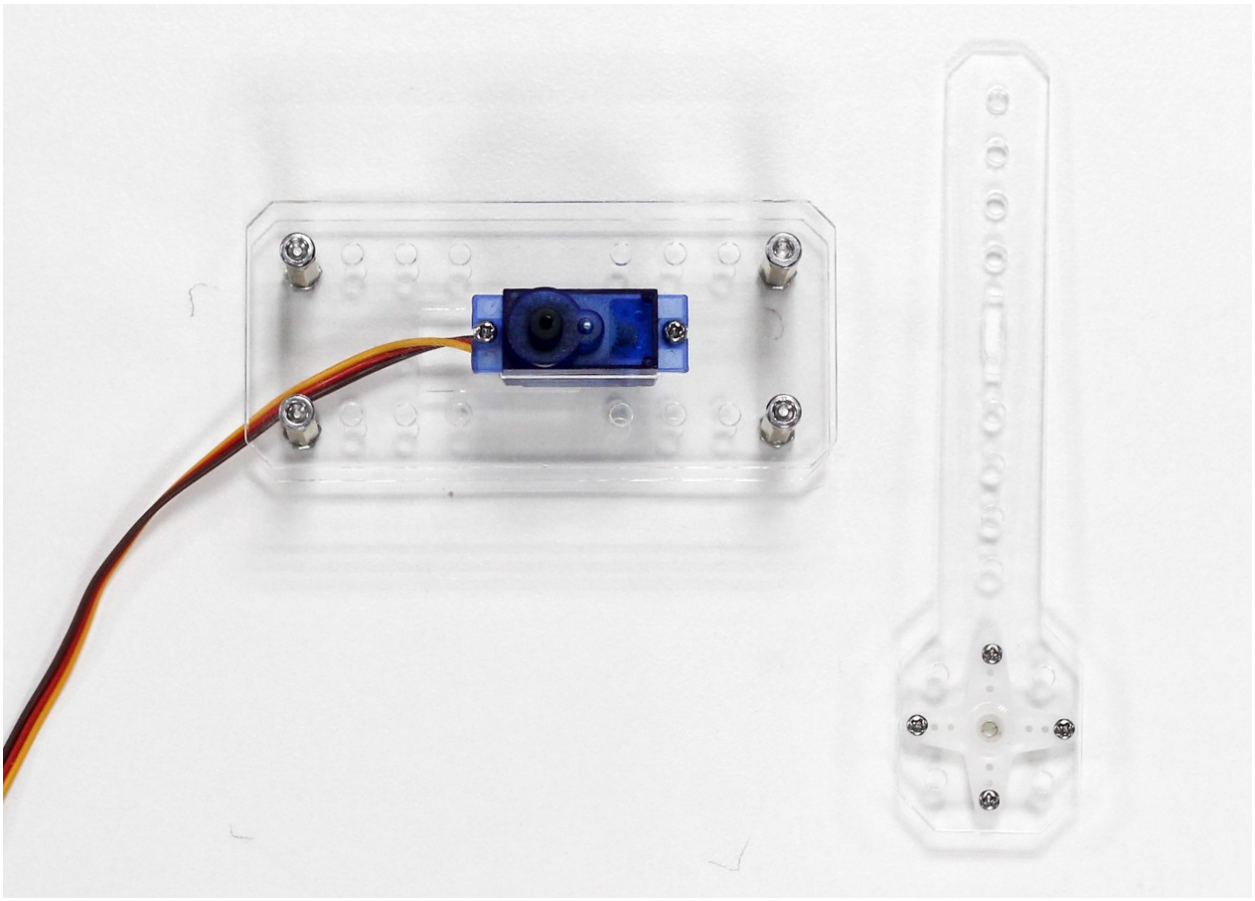
2. 将舵机用 M2 的螺丝和螺母固定到亚克力顶板。注意舵机安装方向，保持舵机轴在正中。



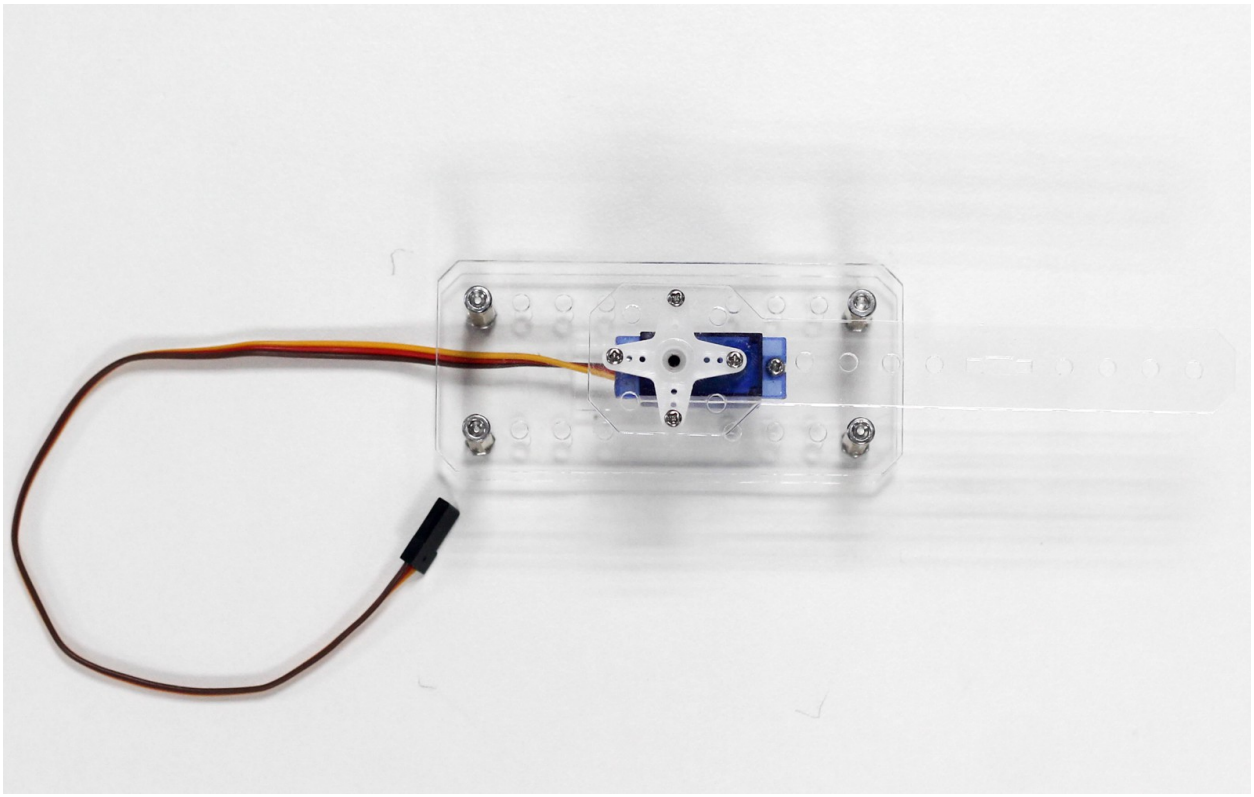
3. 将十字舵机盘用 M1.7 的自攻螺丝固定到亚克力短臂上。



4. 将亚克力顶板与底板铜柱用螺丝连接



5. 将舵机臂安装到舵机轴上，用螺丝固定。后续可在亚克力转臂上安装红绿灯模块或交通卡片。



CHAPTER 19

MU 无人驾驶套件教程

本文介绍 MU 无人驾驶套件在 MakeCode 环境中开发的教程。

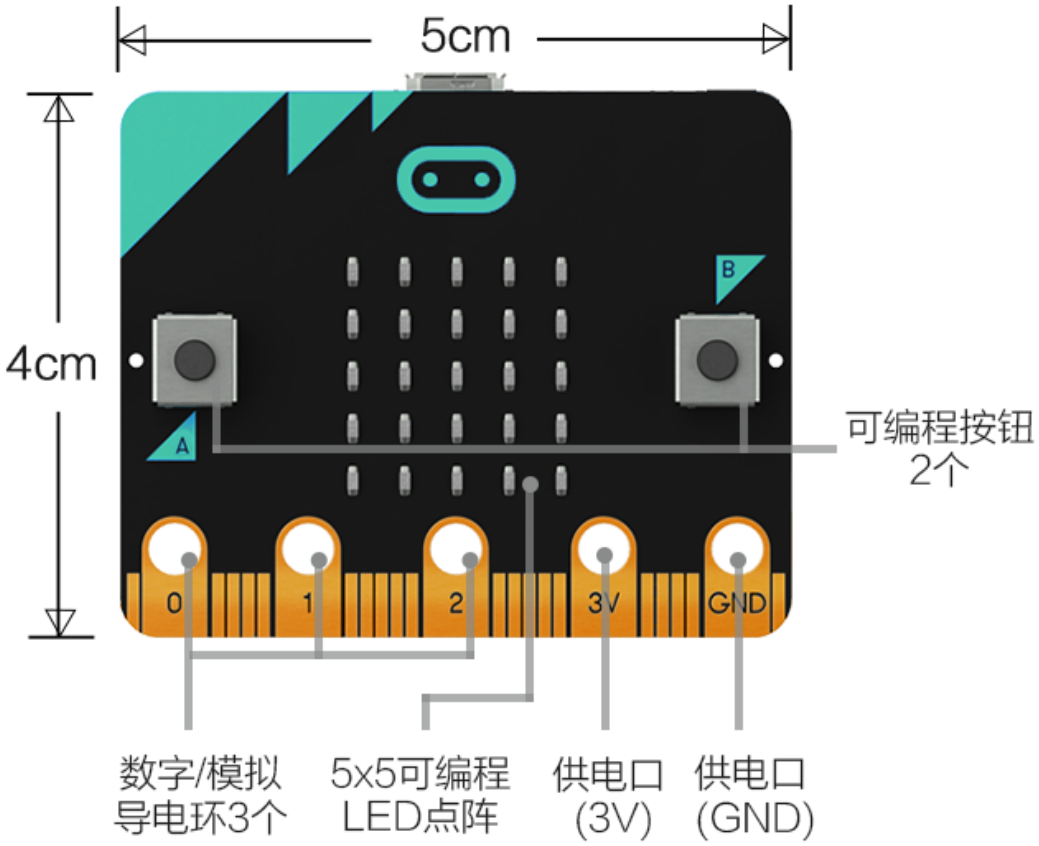
MakeCode 是一款由微软开发的可视化编程软件，适配 Micro:bit、乐高 EV3 等硬件平台。

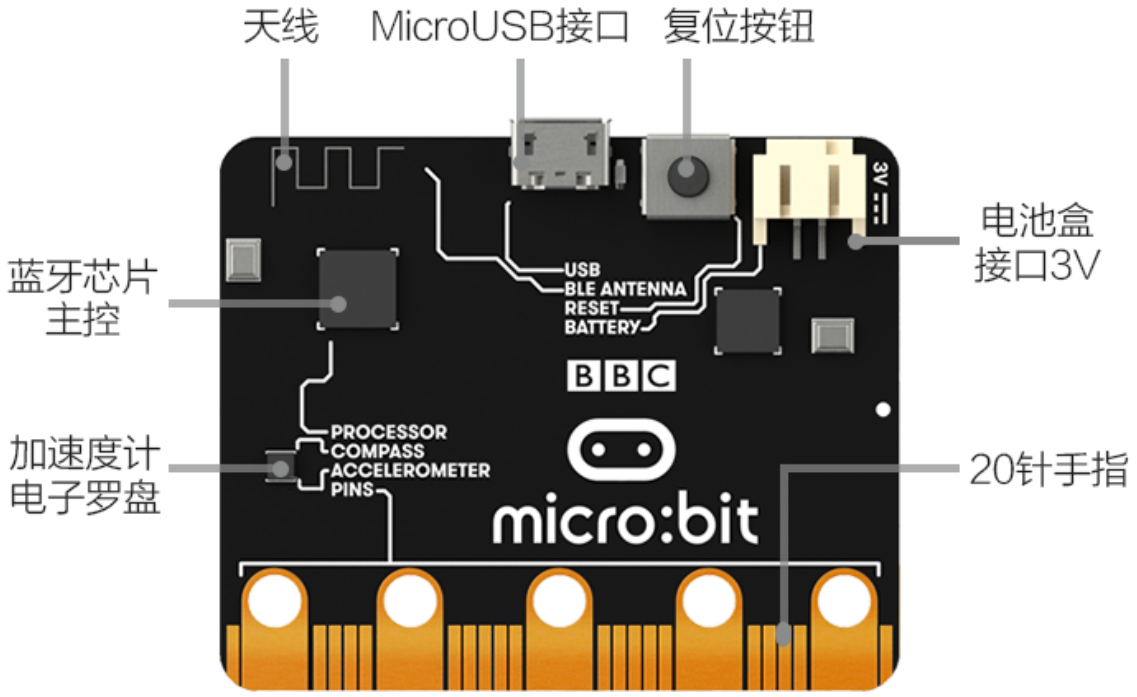
MakeCode 在线编程：MakeCode for micro:bit

19.1 基础教程

19.1.1 简介

micro:bit 是一款全球流行的 STEAM 教育开发板，由英国 BBC 广播公司开发。micro:bit 本身集成了传感器和灯光显示等外部设备，如下图，接入电脑即可编程体验基础功能；当插入扩展板类产品如麦昆小车，可以有更多的控制电机、氛围灯等功能。


5cm
4cm
A
B
0
1
2
3V
GND
可编程按钮
2个
数字/模拟
导电环3个
5x5可编程
LED点阵
供电口
(3V)
供电口
(GND)

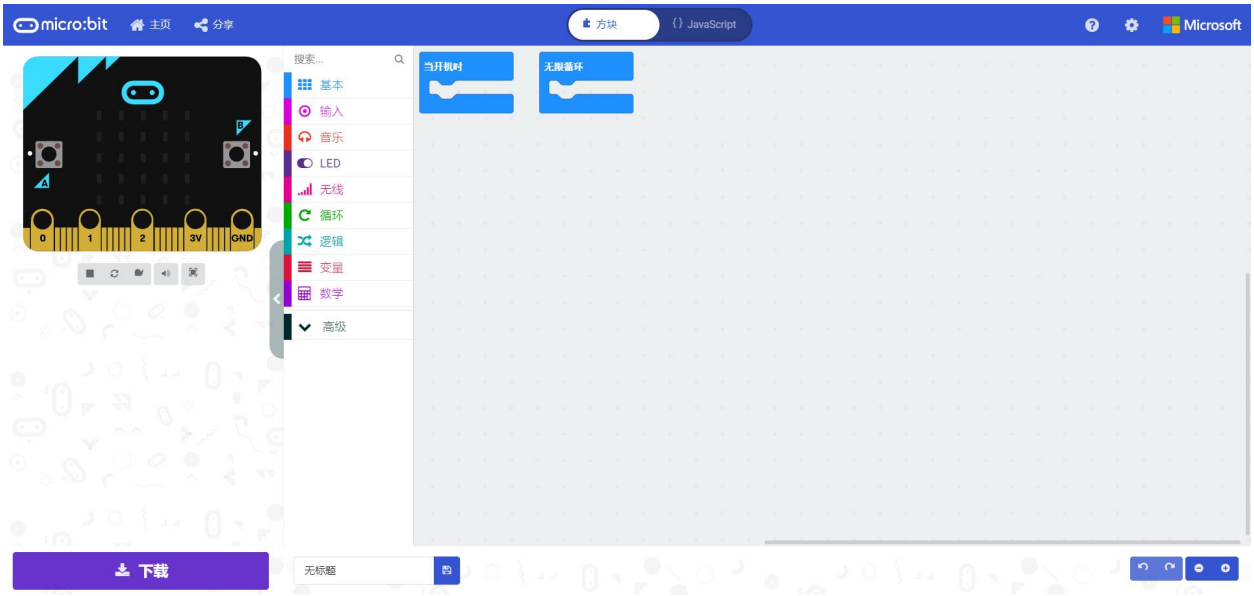

天线
MicroUSB接口
复位按钮
蓝牙芯片
主控
加速度计
电子罗盘
电池盒
接口3V
20针手指
USB
BLE ANTENNA
RESET
BATTERY
BBC
PROCESSOR
COMPASS
ACCELEROMETER
PINS
micro:bit

micro:bit 官方采用 MakeCode 软件进行编程。编程页面：MakeCode 在线编程

本教程围绕 MakeCode 软件进行编程块的教学，逐步加入 AI 内容，帮助用户入门并掌握自动驾驶套件的内容。

19.1.2 编程界面

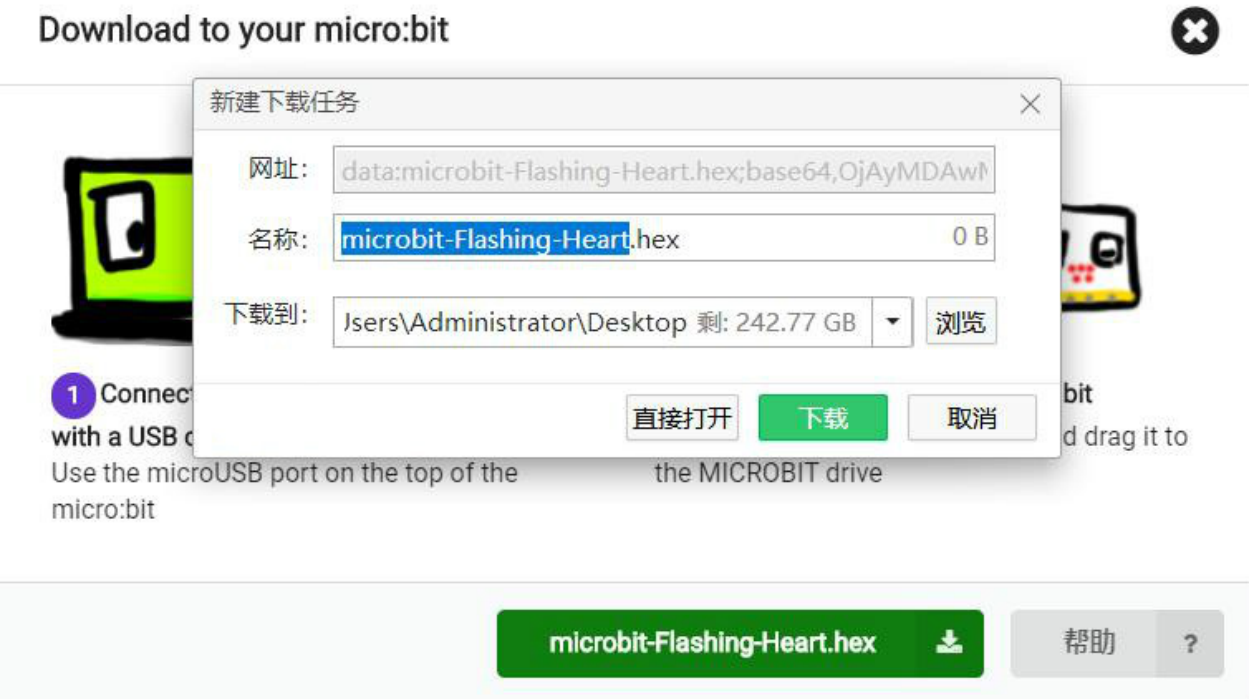
在主页“我的项目”中新建一个项目，进入编程界面。



程序烧录

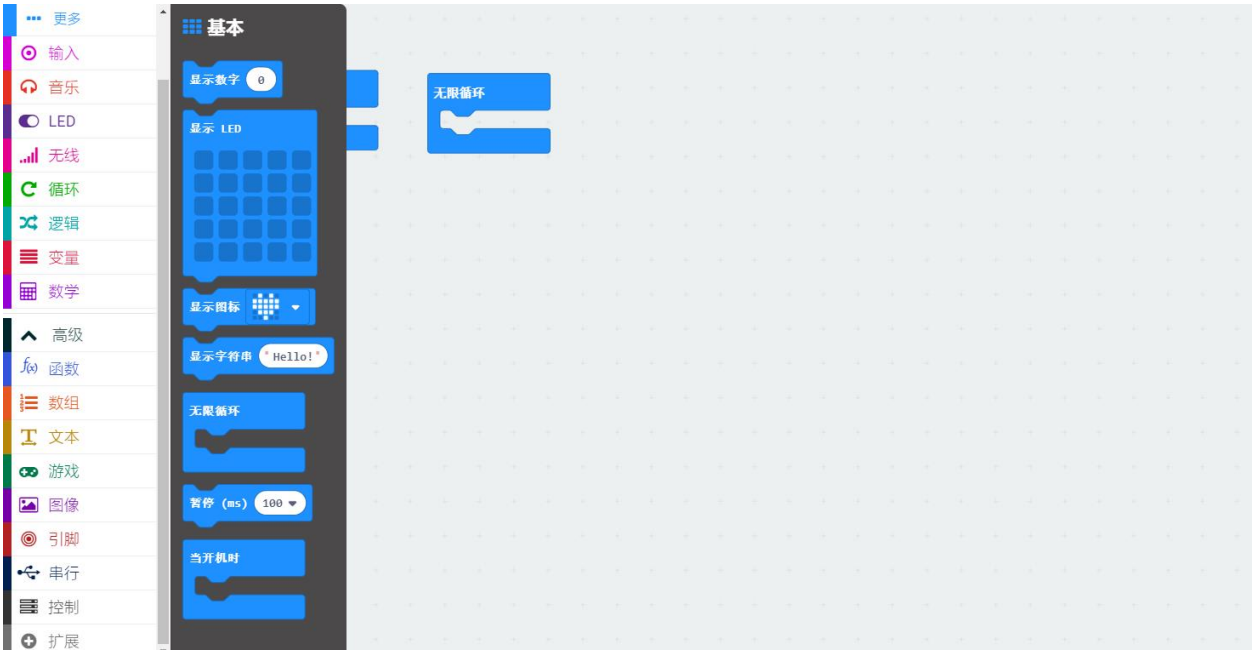
在编程界面中，最左侧是模拟器和下载按钮，部分程序可以直接模拟运行查看效果，所有程序编译完成后都可以下载到 micro:bit 上运行。

micro:bit 连上电脑后显示一个模拟 U 盘，将程序编译后生成的 hex 文件放入 U 盘就会重启 micro:bit 运行该程序。



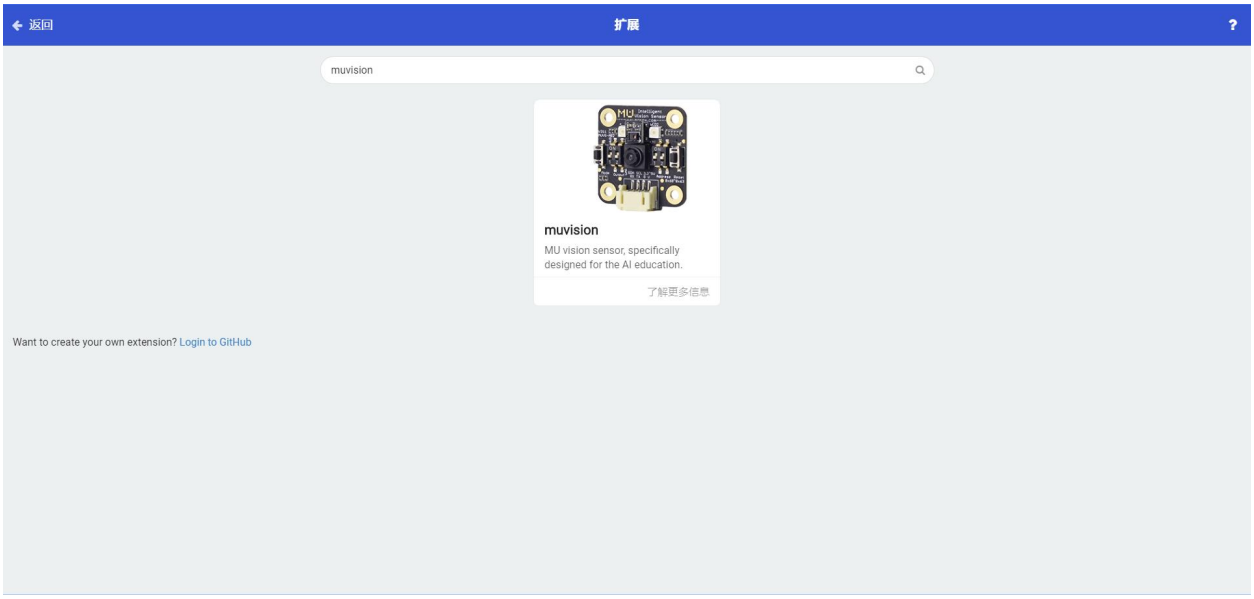
编程块操作

界面中间是编程主界面，拖放可选择的编程块至右边即可编程，鼠标右键点击可重复、删除、显示帮助等操作，鼠标放置在编程块处会显示编程块的简单说明。



添加扩展

除了基础编程块，套件还会用到外部设备的扩展编程块，打开高级-扩展，搜索”muvision”添加 MU 视觉传感器相关编程块，搜索“maqueen”添加麦昆小车相关编程块。其他第三方编程块导入同理。



界面顶部可切换至代码编程，左侧会显示资源管理器，可查看程序相关源文件。本教程以可视化编程为主，代码仅做参考。

19.1.3 完整示例

拖动显示 LED 块到无限循环中，复制一个放到下方，将上方的 LED 灯点亮一个图案比如爱心，就是一个简单的闪动爱心的程序，在左边的模拟器可以看到模拟效果。

点击“下载”，编译生成 hex 文件，将 micro:bit 连接至电脑，将 hex 文件放入 micro:bit 模拟 U 盘。micro:bit 将会自动重启，数秒后看到程序实际运行效果。

Tips：默认块中，“当开机时”中的程序运行一次，然后进入“无限循环”中循环运行。

19.2 执行器类教程

执行器是接受控制信号并对受控对象施加控制运行作用的装置。作为输出类设备，通常用简单的从上至下的顺序控制就可以使执行器动起来。通过相应的编程块来掌握以下执行器的直接控制。

下载本页示例：执行器类程序

19.2.1 电机控制

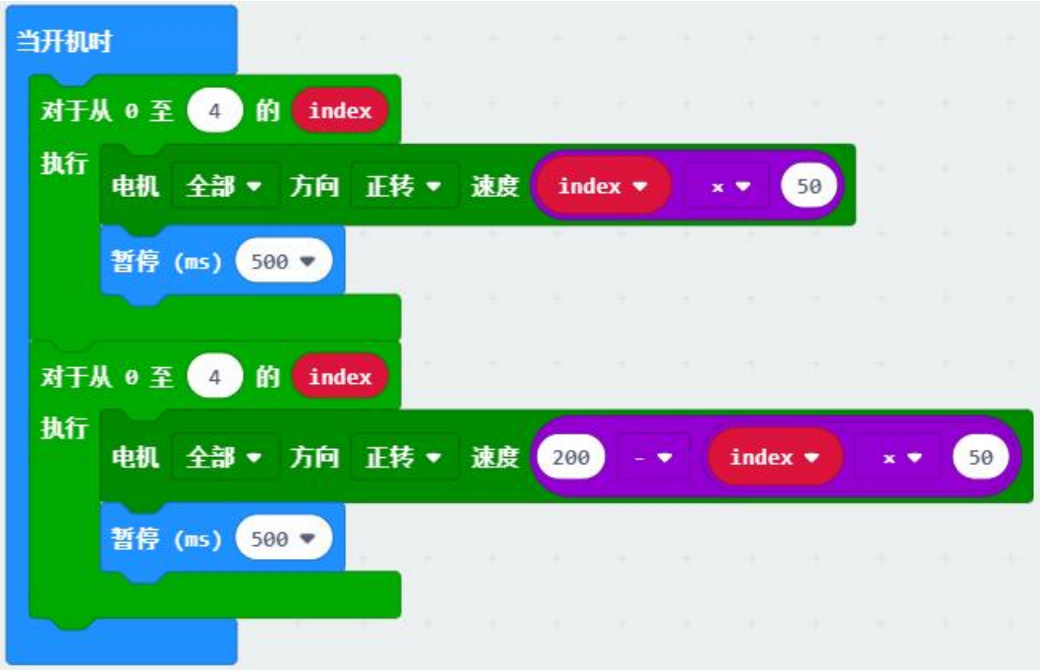
电机 (motor) 俗称马达，是常见的电能转换为机械能的装置。电机可划分为直流、交流，有刷、无刷，永磁、电磁等很多种类。小车常用直流减速电机作为车轮的驱动电机。通过控制左右轮的速度、方向即可实现小车向各个方向的运动。

示例一：控制小车运动方向

程序说明：通过控制两个电机运动方向实现小车方向的变化，让小车分别前进、后退、左转、右转、停止，每个动作持续 2 秒。

示例二：控制小车加减速

程序说明：使用循环控制电机速度依次增加，电机速度从 0 增加到 200，再减到 0，每次变化 50. index 循环中，每执行一次 index 加 1，index 乘以相应的倍数 50 作为电机速度，随循环次数而变化。

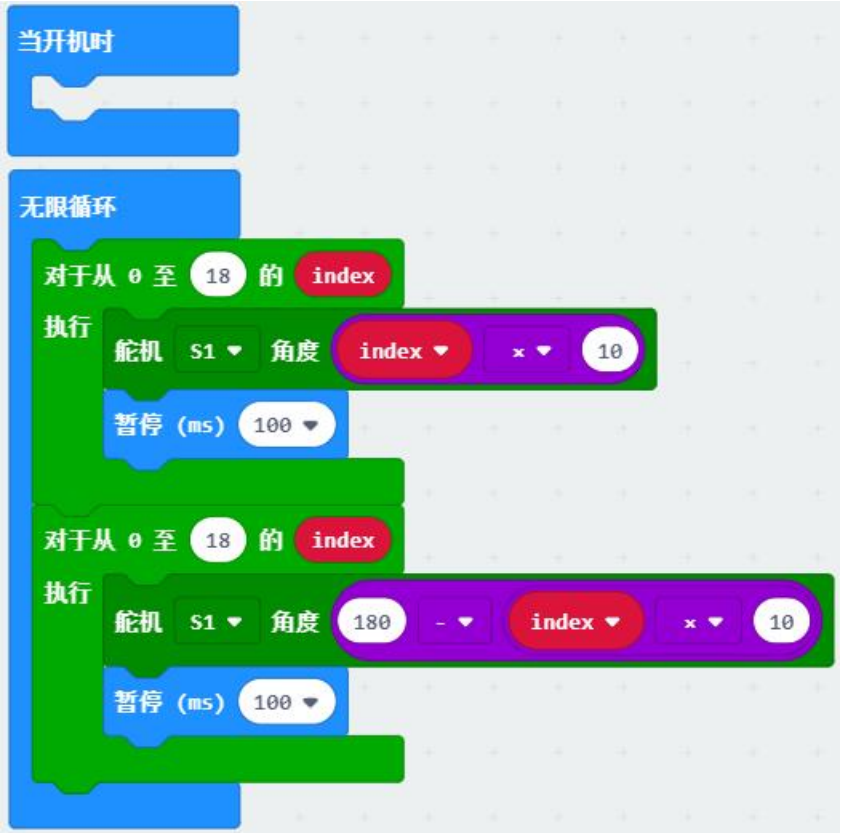


19.2.2 舵机控制

舵机 (servo) 是一种简单的伺服电机，常用在车模、船模、机器人等产品，用于掌舵控制方向。舵机内部采用直流电机驱动，多级齿轮减速器产生大扭矩，电位器确定当前角度。用 PWM 信号控制舵机转到指定角度即可让舵机快速转到某个角度后保持不变。

示例：控制舵机摆动

程序说明：循环写入舵机角度，每 0.1 秒改变一次角度，舵机缓慢在 0-180 度范围摇摆。index 乘以相应的倍数作为舵机角度。



19.2.3 灯光控制

灯光是最基础的输出类设备，可作为装饰、照明、程序调试等功能。这里介绍几种不同的常用灯光和相应的控制。

单 LED 灯

麦昆小车前方有两个 LED 车灯，通过简单的打开和关闭控制两个灯的通断。

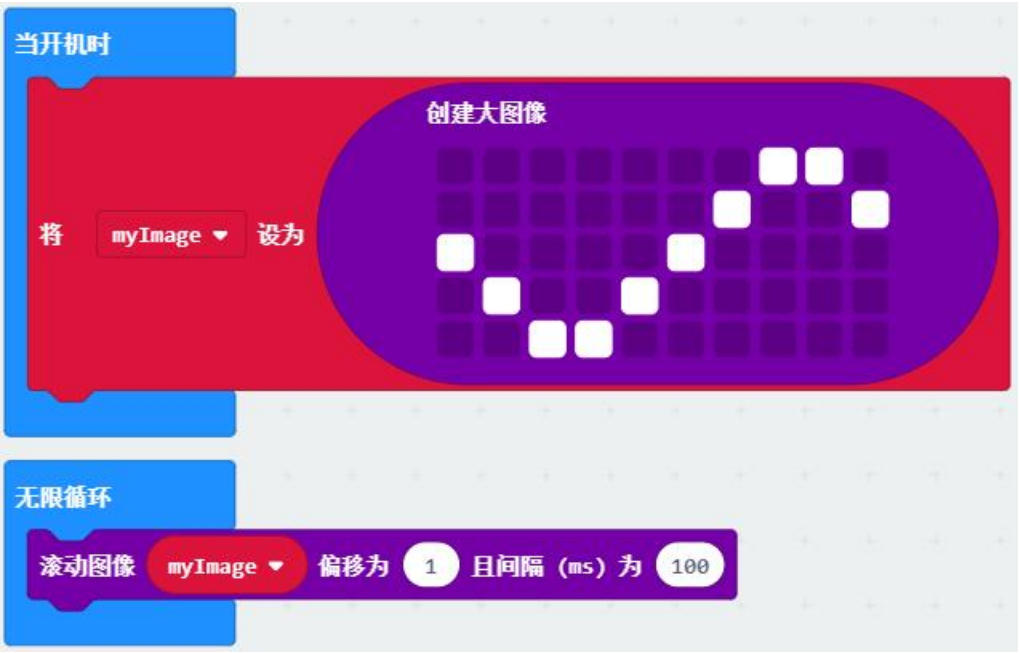
示例：LED 交替闪烁

micro:bit 点阵

micro:bit 自带的点阵由 25 个 LED 灯组成，可以形成丰富的图案和字符。基础示例中已展示绘制爱心，以下程序介绍 LED 滚动显示过程。

示例：显示滚动图像

程序说明：在 LED 高级块中图像，创建一个大图像，在无限循环中调用，每隔 200ms 偏移产生滚动动画。

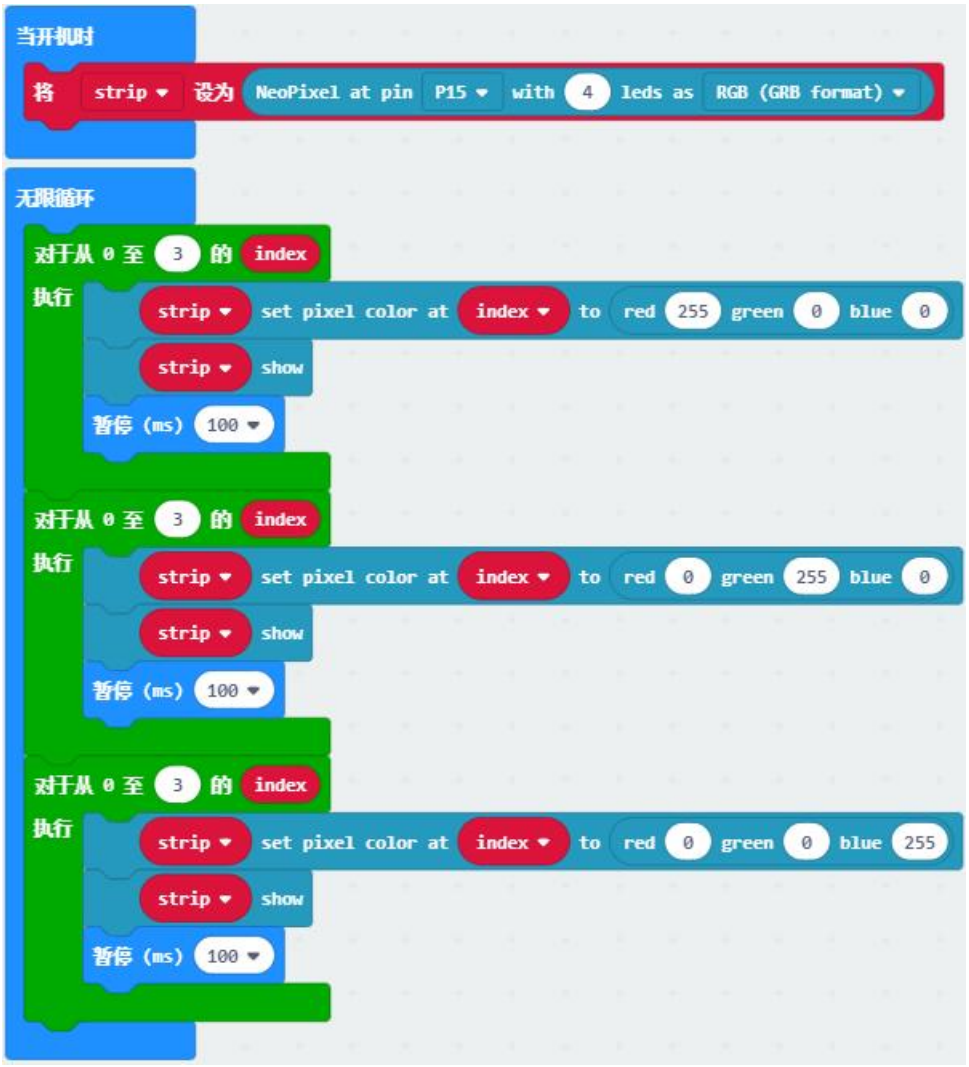


串行灯

麦昆小车车底有 4 个氛围灯，这是一种特殊的带芯片的串行灯，通过发送一串数据可控制整串 LED 的 RGB 值，产生各种各样的颜色。驱动这 4 个灯需要添加”neopixel”扩展。通过车底丝印可以了解这些灯的连接引脚 P15 和串联顺序。

示例：流水灯

程序说明：通过控制每个灯的 RGB 值来点亮所有灯。初始化中配置灯的引脚 P15，数量为 4。LED 的灯号为 0-3，每隔 100ms 依次亮红、绿、蓝灯，形成流水灯。



19.2.4 音乐控制

micro：bit 的 P0 脚可以接蜂鸣器，制作简单的音乐。打开小车上的蜂鸣器可以启用该音乐功能。通过编程块播放内置的音效，或者用音调自制音乐。

示例：小星星

程序说明：通过编程块改变音调和节拍形成顺序的音乐段，以《小星星》的乐谱作示范。

19.3 传感器类教程

传感器（sensor）是一种检测装置，能将测量到的信息，转换为电信号输出。对于主控来说传感器主要是输入设备，在自动控制中起重要作用。传感器的类别非常多，功能和数据也各有不同。本教程通过 micro：bit 相关传感器的使用来表达反馈控制的过程，见微知著。

下载本页示例：传感器类程序

19.3.1 按键

按键是最简单的输入设备，也可以当做是一种传感器。按键主要有通断两种状态，对应程序中的真或假、0 或 1，通过两种状态的检测判断产生反馈，做出相应反应。实际生产生活中还有触摸按键、红外按键、人体红外开关、限位开关等各类按键的变种，控制方式和普通按键也有很多类似之处。

示例一：获得按键状态

程序说明：通过条件逻辑检测按键的状态，如果按键 A 被按下则屏幕显示 A，如果按键 B 被按下则屏幕显示 B，否则不显示，每 0.1 秒检测一次。

示例二：中断事件

程序说明：中断是独立于初始化和无限循环的另一种状态，由某种状态触发，如本例中的当按键被按下。程序在执行完中断程序后会回到原来的位置。本例中主程序为显示爱心，中断程序为 A 或 B 被按下时切换显示 A 或 B，之后回到循环继续显示爱心。

19.3.2 指南针

指南针也称为地磁传感器，能够感应地球磁场并输出当前角度，得到方位信息。

示例：电子指南针

程序说明：通过 micro:bit 板载的指南针获得当前的角度，用屏幕显示东 (E) 西 (W) 南 (S) 北 (N)。将 micro:bit 平放转动即可看到角度。指南针初次使用需校准，翻转 micro:bit 将点阵全部点亮即可。

程序扩展：加入东北、东南、西北、西南，使指南针更丰富。

19.3.3 加速度计

和指南针类似，加速度计也是一种惯性测量单元（IMU），可以测得当前多个方向的加速度值，也可以通过数据的累加、判断大小来获得速度、旋转角度、震动等信息。

示例：计步器

程序说明：类似于运动手环的计步功能，用变量 step 作为步数，屏幕显示当前步数。当震动时中断，step 累加。

程序扩展：如果在野外被五步蛇咬了怎么办？计步器试试计到 4 步就报警。

19.3.4 超声波传感器

超声波传感器是最常见的测距传感器，利用声波的发射和接收时差乘以声速获得距离信息，类似于蝙蝠的测距。超声波传感器测量距离可以达到 0.04-4 米。

示例一：超声波测距

程序说明：用变量 distence 记录超声波测距的值，并通过 micro:bit 的点阵显示该距离。

示例二：避障小车

程序说明：超声波常用于小车避障，是一个典型的反馈控制过程。小车默认以 100 的速度直行，距离前方障碍小于 20cm 时，小车原地调头。

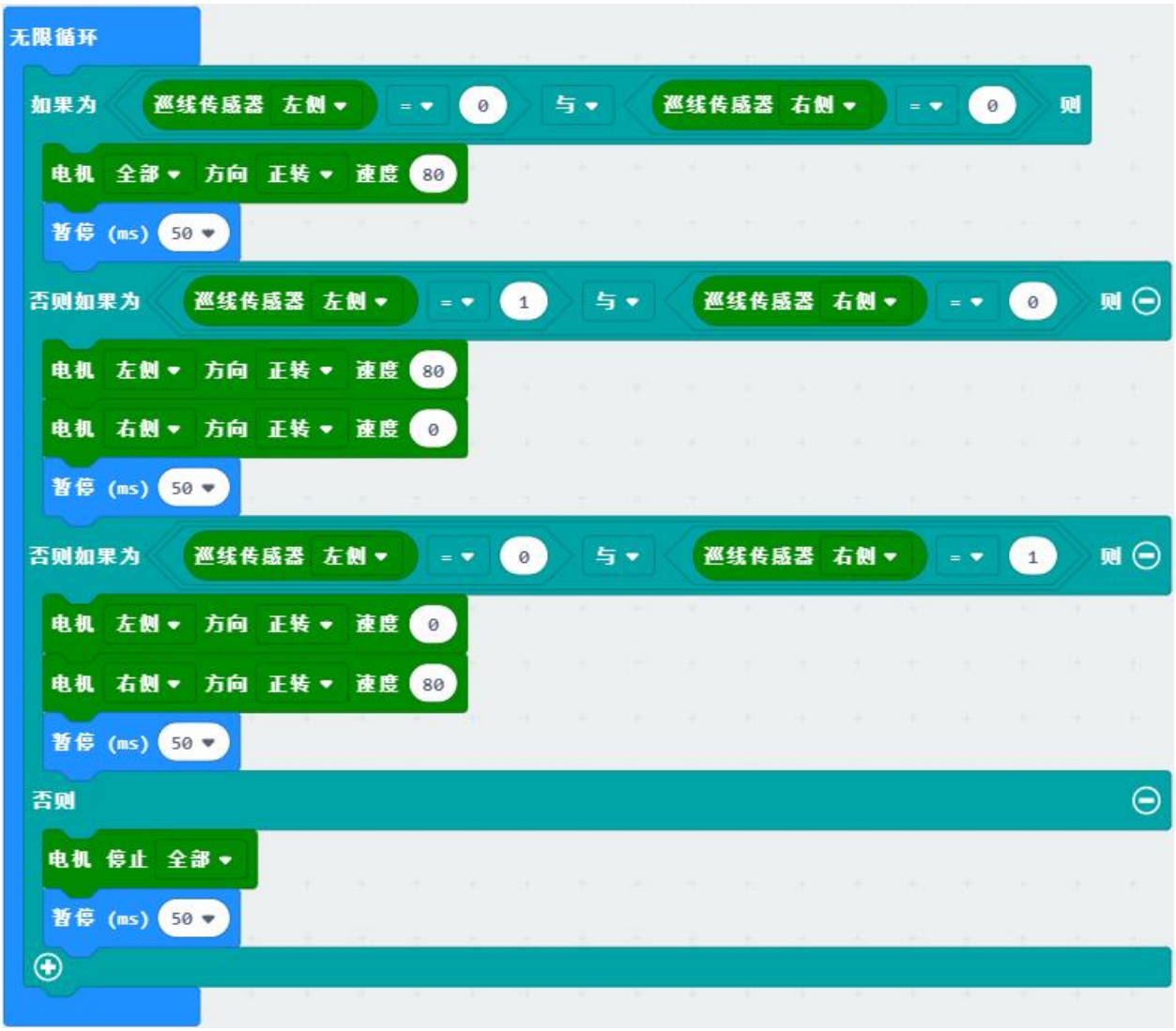
程序扩展：尝试不同避障距离时有不同的避障路线。比如距离小于 10cm 则后退调头，距离 20cm 则右转 90 度。

19.3.5 红外传感器

红外传感器可以检测黑白色，本质是一对红外开关。一个红外灯发射红外光，遇到前方白色时反射，黑色吸收，另一个接收头检测是否接收到，产生 0 和 1 两种状态。使用多组红外开关可以避障或巡线。

示例：红外巡线小车

程序说明：麦昆小车底部有两组红外传感器，可用于红外巡线。红外的 0 和 1 两种状态表示在黑线内还是黑线外，通过两对红外状态的判断让小车直行、左转还是停止，每 50ms 判断一次。



19.4 通讯类教程

在实际的设备与设备间、芯片与芯片间往往有各种通讯协议 (protocol)，用于传输复杂的数据，本质是按照预定的规则收发数据。在可视化编程中部分收发规则已简化，配置了默认的引脚、格式，而内部的数据类型、字符的协议需要用户自己定义。

下载本页示例：通讯类程序

19.4.1 串口

串口通讯（Serial Communication) 是设备间常用的串行通讯方式，比如 micro:bit 与电脑间就可以通过串口收发数据。串口常用于程序的调试环节，可以将程序中的数据配合一些解释通过串口打印在电脑上。串口设备接入电脑后会显示相应的 COM 口，在此电脑右键左侧设备管理器中可以找到。

示例：串口打印数据

程序说明：用串口替代 micro:bit 的点阵来显示超声波测距数据，打印在控制台上。用变量 distence 记录超声波测距的值，串口打印描述性的语句和 distance 的值来显示数据，常用于程序的调试。在编程界面左侧打开“显示控制台设备”即可看到串口返回的信息。

19.4.2 无线

micro:bit 主控芯片自带无线通讯功能，是一种分组广播的通讯形式，当两块 micro:bit 设定在同一分组时即可收发数据。

示例：无线遥控小车

程序说明：此程序需要两块 micro:bit，一块发送，一块接收。

接收端程序：无线接收字符以触发相应的动作。开机时匹配无线组 1，无限循环为空。在接收到无线数据时产生中断，根据无线接收到的数据让小车对应运动。

发射端程序：通过按键 A B 触发无线发送。开机时设置无线组 1，发射功率为 7。循环检测按键状态，当 A、B 或 A+B 被按下时通过无线发出对应字符。

程序扩展：体感遥控小车，将发射端程序改为加速度计触发，倾斜 micro:bit 来控制小车。当手中的 micro:bit 前后左右倾斜时让小车对应运动。

Tips: 这里无线发射和接收就是一个简单的通信协议，用数字 1、2、3、4 表示左转、右转、前进、后退。熟练以后可以设计更多复杂的协议。

19.4.3 蓝牙

19.5 程序高级教程

本文系统性介绍程序的相关概念，较为理论，结合各类实际程序总结归纳。

19.5.1 基本 (basic)

当开机时：初始化程序，开机时最先运行一次。一般将设备的初始化放在此内。

无限循环：主程序，简单程序的不断运行，或者复杂程序中调用函数。

暂停：维持当前状态一段时间，防止程序运行过快跳过某些动作。一般可用几十毫秒至数秒，过小或过大可能出错。

中断：独立于初始化和主程序，满足某种触发条件后执行内部程序，可以是按钮、振动、无线接收等各类外部触发条件，执行完毕后回到被中断的程序处继续运行。

19.5.2 循环 (loop)

repeat 循环：指定循环次数

while 循环：当满足某个条件时循环。“无线循环”本质就是 while true，表示一直循环。

for 循环：使用 index 索引计数来循环，从 0 开始，每次加 1，index 可以被内部调用。

for of 循环：以数组中的元素个数计数循环，用值 (value) 引用数组中的元素

19.5.3 逻辑 (logic)

if 条件：如果，否则如果，否则，否则如果可以有多个，将多种可能性的时间列举在内，常用于传感器的状态判断。

布尔运算：布尔值 (Boolean) 指正确 true(1) 或错误 false(0)。布尔运算指且 (and) 或 (or) 非 (not)，常用在多种条件出现时的判断过程。判断结果返回一个布尔值。

比较运算：数值或字符串间的大小对比。比较结果返回一个布尔值。

19.5.4 变量 (value)

设置变量：新建一个变量，在程序的某些位置调用，变量名要具有可读性。

变量赋值：变量可以赋值为其他数值，或以 n 为幅度累加，x = x + n。

19.5.5 数学 (math)

数学运算：加减乘除、取余、平方根、三角函数、四舍五入等常见数学运算。

随机数：指定某个区间内返回一个随机数，或返回随机布尔值。

19.5.6 函数 (function)

编程中的函数是一个功能性的过程，也称为子程序，通常是进行一段运算或控制一段输入过程。当程序中的过程反复出现时，可以建立一个函数进行调用。

新建函数：建立一个能被调用的函数。可以带有文本、数值或布尔值的参数，在调用时对参数赋值。

19.5.7 数组 (array)

一串数组中每个值都是一个变量，各变量的排序称为索引，从 0 开始。

改变数组中的值：获得目标值的索引，将该位置的值替换，不改变数组长度。

改变数组长度：从首、尾、中间移除或者添加值，重新建立索引，数组的长度改变。

19.5.8 文本 (text)

在打印调试会用到文本来解释数据，而受到一串文本常需要从中解析数据。

文本解析：获取文本的长度，提取字符或字符串，文本间的比较。

文本转换：将文本转换为数字、整数，或将数字转换为文本。

19.6 硬件高级教程

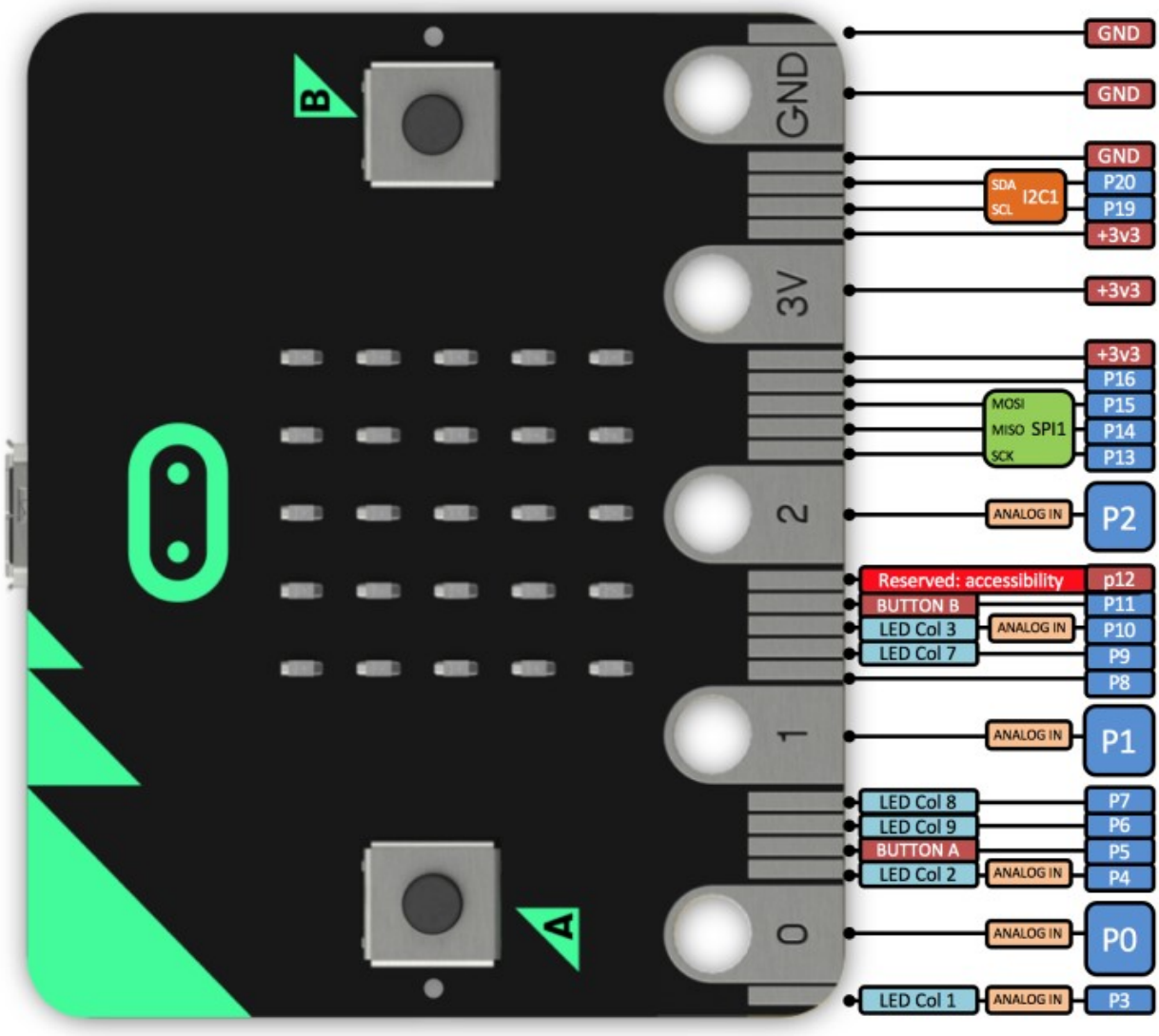
在 Makecode 中很多默认硬件相关的编程已做简化，而在使用外接设备时，比如插入扩展板，就涉及到引脚的相关控制。在使用麦昆小车时就会涉及到引脚的精确使用，如蜂鸣器接在 P0 脚。通过硬件相关的系统学习，在使用 micro:bit 外接传感器或执行器时就能如鱼得水，自由设计。

下载本页示例：硬件高级程序

19.6.1 GPIO

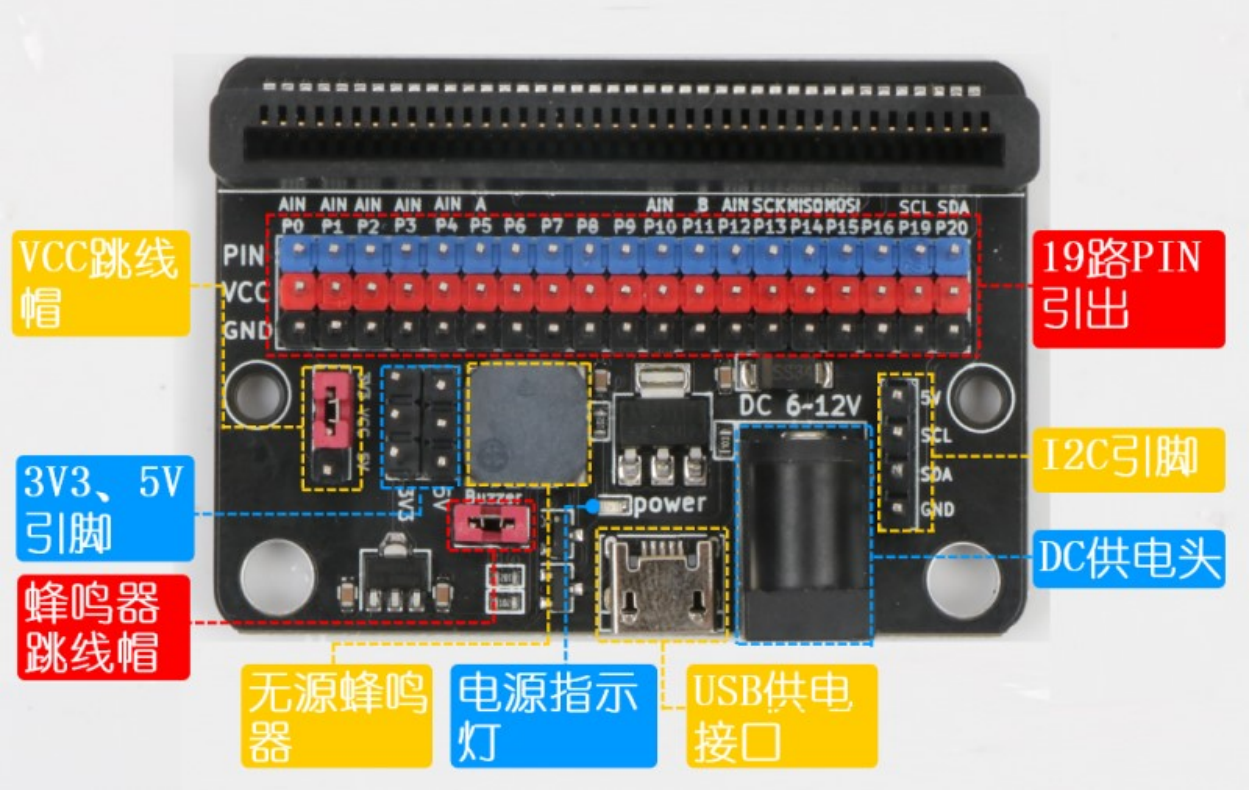
通用输入输出端口（GPIO）是主控芯片提供的可供编程的接口，对应着芯片上的各个引脚。在 micro:bit 中有部分引脚已被系统使用，留出的引脚可供用户编程。在高级-引脚里可以看到引脚 P0-P20 中的部分可以使用。而板载的按键、LED 等外设（外部设备的简称）同样也用到了其中部分引脚。

micro:bit 引脚图：



可以看到 micro:bit 可用的引脚已经通过金色的金手指引出，在套件中配了一块扩展板，则能将金手指转换成排针，方便连接舵机、LED 等外部设备。蓝色的是引脚插口，红色是 VCC，黑色是 GND。扩展板可用圆形 DC 口或 USB 口供电 (扩展板 USB 口只能供电，不能下载)。

扩展板接口图：



示例：点亮红绿灯模块

程序说明：将红绿灯模块连接到扩展板的 P15 口，参考串行灯程序，用扩展板点亮红绿灯模块，程序下载完后用电池盒通过 DC 口供电，脱离 USB 线。

示例：GPIO 输出

程序说明：将 LED 车灯示例用 GPIO 直接控制的方式来实现，在麦昆小车上找到车灯的引脚 P8 和 P12，输出数字量 1 即可点亮，数字 0 关闭。

程序扩展：试试按键示例用数字读取引脚的值来实现。

19.6.2 ADC 与 DAC

模拟量是在一定范围连续变化的变量，比如声音的大小。数字量是时间上是不连续变化（离散）的量，比如开关的状态，只在 0 和 1 之间变化。

模拟量 (Analog) 与数字量 (Digital) 在电路中的转换过程称为 ADC(模拟转数字) 和 DAC(数字转模拟)。micro:bit 中的 ADC 是指测量电压模拟量读取，用 0-1024 的数字返回结果，测得的电压为

Volt = 3.3 * (Value/1024)

可以看到模拟转换为数字时有一定的精度，电压被分为 1024 份，测得的数字量约等于输入的模拟量。

示例：加速度值

程序说明：通过串口查看加速度值。下载程序后打开控制台，左右倾斜 micro:bit，查看绘制的模拟曲线和返回的数字值。

DAC 则相反，指定 0-1024 的数字模拟写入，输出对应的电压值为

Volt = 3.3 * (Value/1024)

示例：呼吸灯

程序说明：麦昆自带的车灯模块并没有调节车灯亮度的功能，但可以通过控制输出引脚的电压改变车灯亮度。用模拟写入 0-1000 的值，车灯就有逐渐变亮的呼吸效果。

19.6.3 PWM

PWM 是指脉冲宽度调制，在指定的周期时间上输出一定比例的高电平脉冲来输出数字信号。最常见的就是舵机的 PWM 信号，脉冲的周期是 20ms，舵机角度 0 度对应高电平 1ms，90 度对应 1.5ms，180 度对应 2ms。

示例: 舵机 PWM 控制

程序说明：通过有引脚的 PWM 输出来控制舵机，向 P1 输出 PWM 信号。

19.6.4 IIC SPI

提供了 IIC、SPI 等接口，在使用相应外设时可以调用接口进行通信。

19.7 MU3 应用教程

19.7.1 MU3 MakeCode 教程

了解 MU3 的基础编程块使用请查看MU3 MakeCode 教程。

19.7.2 MU3 深度开发指南

深入视觉原理和复杂应用请查看MU 视觉传感器 3 深度开发指南。

19.7.3 光线传感器

19.7.4 Wifi 通讯

19.7.5 PID 算法

19.8 综合运用教程

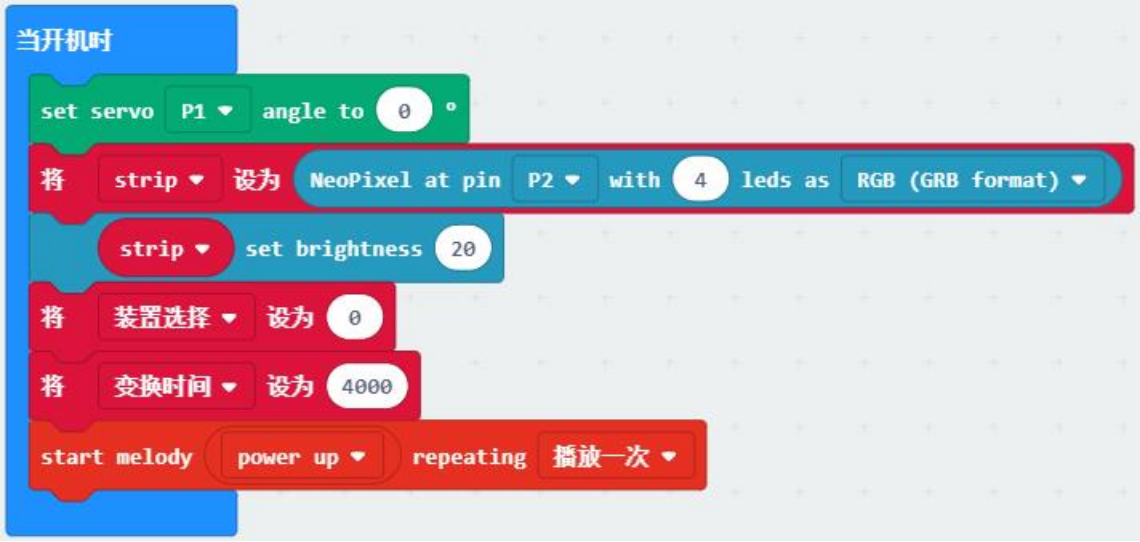
19.8.1 交通控制器

交通控制器中 micro:bit 主控和扩展板作为主控，电池作为供电，舵机和交通灯作为输出，是舵机、交通灯、GPIO 的综运用示例。

下载本页示例：综合运用程序

程序说明：

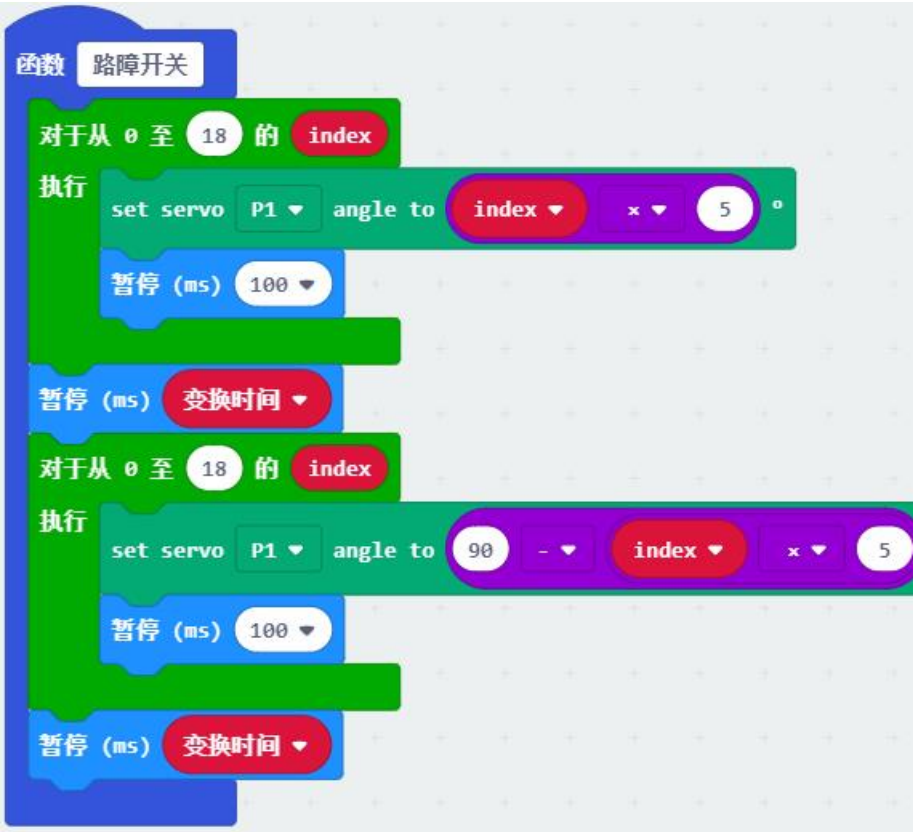
1. 当开机时，设定舵机连接口 P1 和交通灯连接口 P2，初始化用到的变量。

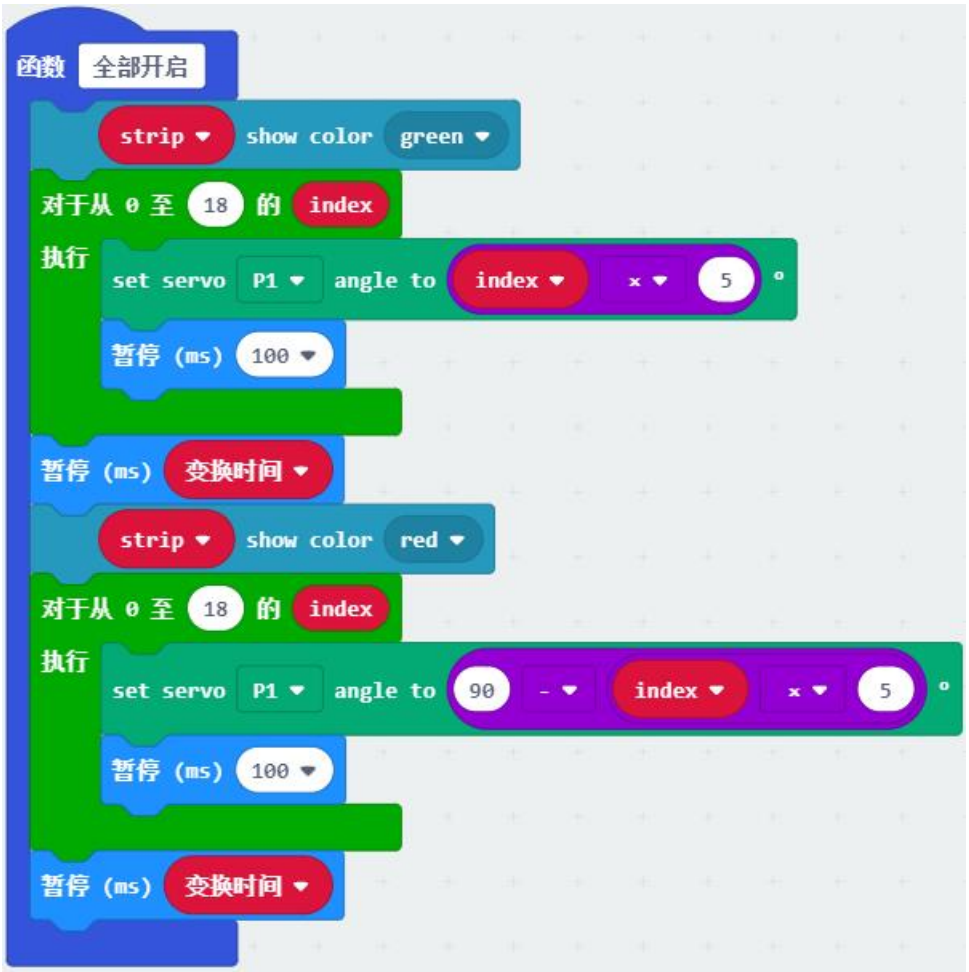


2. 无线循环主程序中，检测变量“装置选择”的数值，启用相应功能。

3. 通过按键中断对“装置选择”进行修改，且约束在 0-3 之间。

4. 各个功能函数，点亮红绿灯，转动舵机，全部开启和全部关闭，可以被主程序调用。





19.8.2 道路内自动驾驶

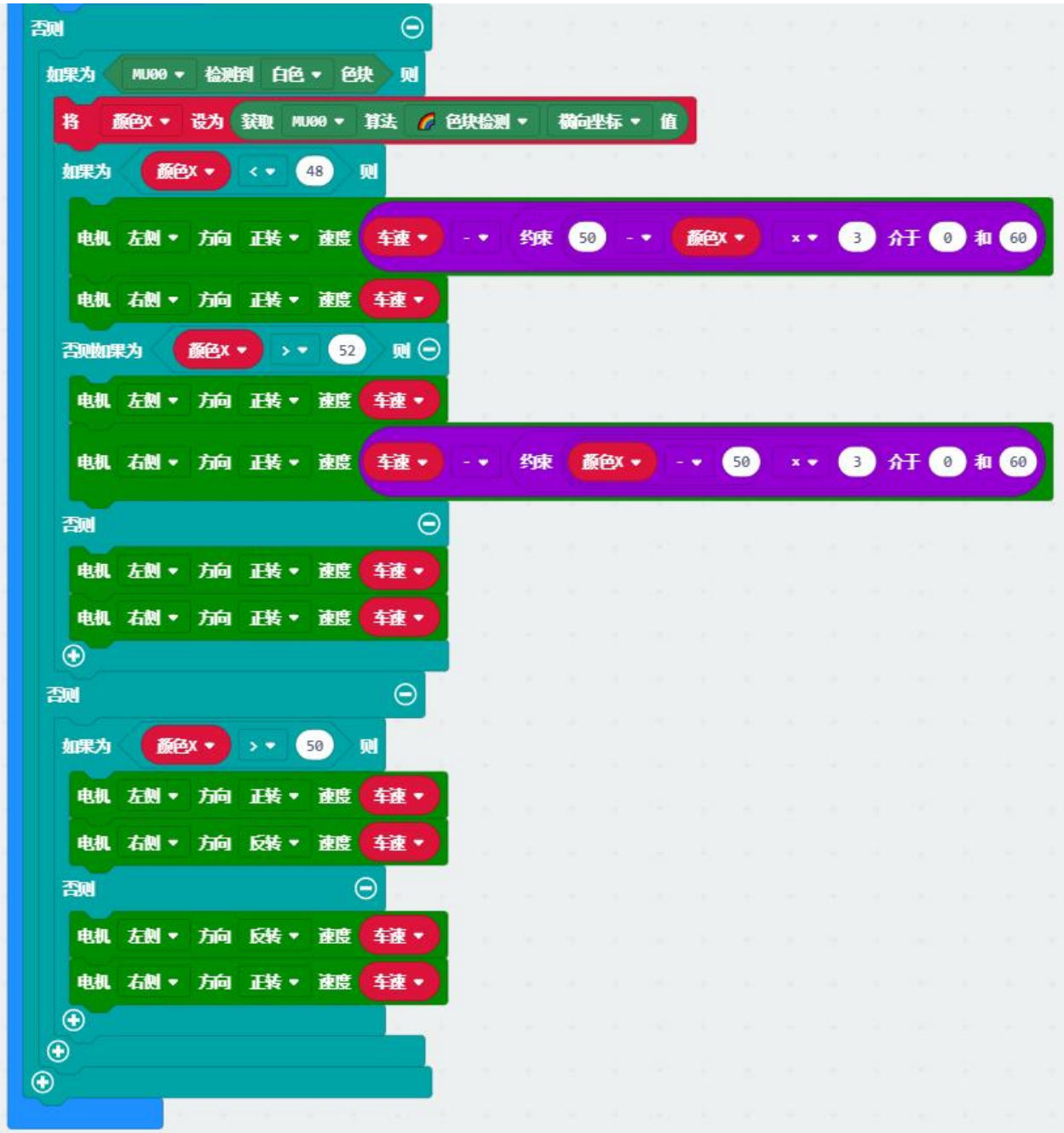
该应用是用 MU 识别白色道路和放置在地上的交通卡片引导小车自动驾驶。调节视觉传感器朝向地面。可用模块化地图块拼成道路地图来形成自动驾驶场景。

程序说明：

1. 当开机时，设定车速和颜色的坐标，初始化视觉传感器，调整摄像头相关的参数。

2. 主程序的第一部分，视觉传感器检测地上是否有交通卡片，并根据卡片类型控制车的动作。

3. 主程序的第二部分，当地上没有交通卡片时，检测白色道路的范围，控制小车前进或转弯，使小车始终在白色道路内。



19.8.3 单线道路自动驾驶

该应用是红外巡线配合用 MU 识别前方的交通卡片和红绿灯来引导小车自动驾驶。调节视觉传感器朝向前方。可用模块化地图块拼成道路地图来形成自动驾驶场景。

程序说明:

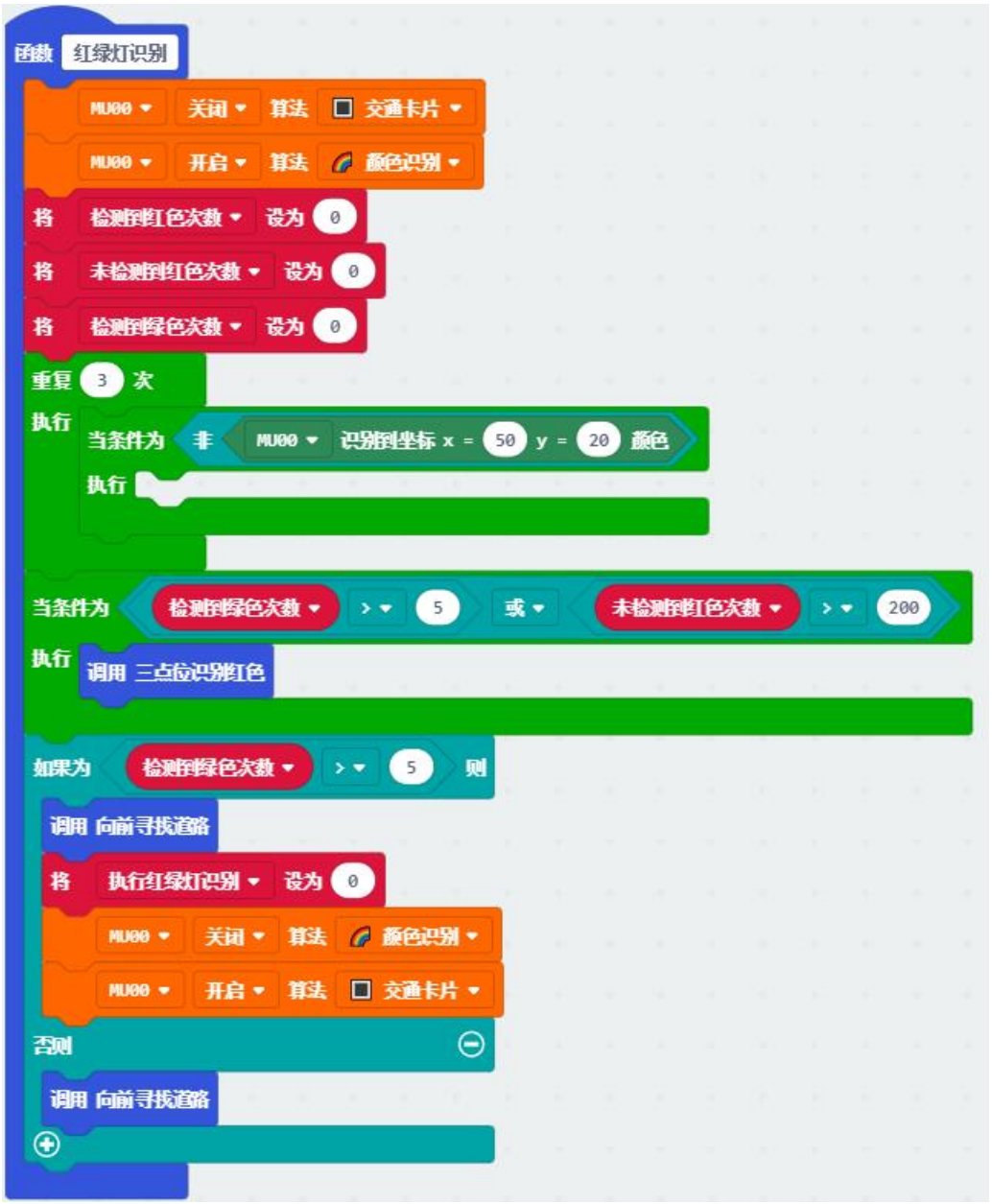
1. 当开机时，设定车速和初始化传感器。主程序就是调用红外巡线。

2. 红外巡线子程序中，当小车在路上时，根据道路判断直行或转弯，可参考红外传感器示例。

当遇到路口时，判断是红绿灯还是交通卡片，调用相关子程序。

3. 在路口首先识别前方交通卡片，根据交通卡片的指示控制车辆转弯或掉头等。

4. 当路口未识别到交通卡片时，继续识别红绿灯，开启颜色识别算法，识别三个坐标位置的颜色，判断是红灯还是绿灯。红灯停，绿灯行。


函数 红绿灯识别
MU00 关闭 算法 交通卡片
MU00 开启 算法 颜色识别
将 检测到红色次数 设为 0
将 未检测到红色次数 设为 0
将 检测到绿色次数 设为 0
重复 3 次
执行
当条件为 非 MU00 识别到坐标 x = 50 y = 20 颜色
执行
当条件为 检测到绿色次数 > 5 或 未检测到红色次数 > 200
执行 调用 三点位识别红色
如果为 检测到绿色次数 > 5 则
调用 向前寻找道路
将 执行红绿灯识别 设为 0
MU00 关闭 算法 颜色识别
MU00 开启 算法 交通卡片
否则
调用 向前寻找道路


函数 三点位识别红色
将 x坐标 设为 48
重复 3 次
执行
当条件为 非 MU00 识别到坐标 x = x坐标 y = 20 颜色
执行
如果为 获取 MU00 颜色识别 颜色 = 红色 则
以 1 为幅度更改 检测到红色次数
否则如果为 获取 MU00 颜色识别 颜色 = 绿色 或 获取 MU00 颜色识别 颜色 = 青色 则
以 1 为幅度更改 检测到绿色次数
以 1 为幅度更改 未检测到红色次数
否则
以 1 为幅度更改 未检测到红色次数
以 2 为幅度更改 x坐标


函数 向前寻找道路
电机 全部 方向 正转 速度 车速 ÷ 2
当条件为 巡线传感器 左侧 = 1 与 巡线传感器 左侧 = 1
执行

CHAPTER 20

MU 无人驾驶套件资源

20.1 技术资料

在以下网址可获取最新的 MU 视觉传感器技术资料：

官网技术支持：http://mai.morpx.com/page.php?a=sensor-support

GitHub：https://github.com/mu-opensource/

20.2 亚克力支架

对于套件中的模块化亚克力板，需要增加数量的用户可自行用激光切割机切割 2mm 的亚克力板。

MU 自动驾驶套件亚克力板

20.3 3D 打印支架

对于购买 MU 裸板的用户，我们提供了 3D 打印的外壳和折叠支架文件，可以在小车等使用场景下固定和调节传感器的角度，有 3D 打印机的创客可自行打印。

MU3 3D 打印支架

20.4 平台链接

MU 视觉传感器可以和各类开源软硬件配合使用，查看各类开源平台以学习其基础知识。

麦昆小车

麦昆小车资料 [wiki 链接]http://wiki.dfrobot.com.cn/index.php?title=(SKU:ROB0148)_micro:Maqueen(V2.0)%E6%9C%BA%E5%99%A8%E4%BA%BA%E5%B0%8F%E8%BD%A6

Micro:bit 教程

Micro:bit 官网 https://microbit.org/zh-CN/

MakeCode 在线编程 https://makecode.microbit.org/#

CHAPTER 21

联系方式

感谢您购买使用本公司的产品，我司会持续更新产品的固件以及配套的资料，您可以从如下网址获取最新的产品信息：

摩图科技官网：http://www.morpx.com/zn.index.html

产品技术支持：http://mai.morpx.com/page.php?a=support

您在使用过程中有遇到任何问题，可以通过以下方式与我司取得联系：

技术支持电话：**0571-81958588**

技术支持邮箱：**support@morpx.com**

- 官方技术支持微信号

- 官方技术支持 QQ 号

- 官方技术交流 QQ 群号

CHAPTER 22

关于本站/About

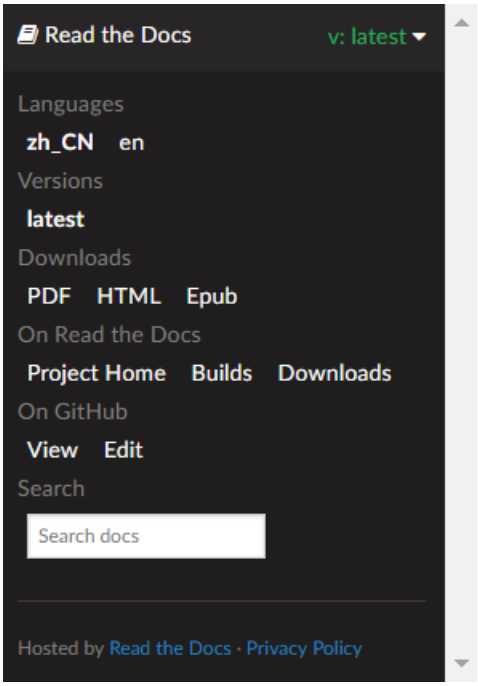
本网站为摩图科技出品的产品相关资料站，在电脑端和移动端均可打开。

This website is document of products produced by Morpx Inc. The website can be opened in computer and mobile devices.

22.1 页面使用说明/Page Usage Notes

点击左下角 **Read the Docs** 按钮打开边栏，可以看到包含的功能。

Click the **Read the Docs** button in the left-bottom corner to open the sidebar pannel. Functions are shown below.



- 切换语言/Change Language

目前大部分文档有中文和英文两种语言的版本，英文版是由中文版翻译而来。若切换语言时显示页面不存在，则为无对应翻译页面。建议点击左上角返回首页后切换语言。如有任何问题，请联系我们。

Most documents have Chinese and English version. English version is the translation of the Chinese. There is no corresponding page if the page does not exist when changing language. Just click the left-top button to return to the homepage and change again. If there is any problem, please let us know.

- 版本控制/Version Control

文档最新版本为 latest，有其他稳定发布版本时会显示。

The newest document is in latest branch.Other stable branch will be shown when published.

- 指南下载/Download the Docs

目前有 pdf、html 和 Epub 文件可供下载，方便离线阅读。

There are pdf, html and Epub files available for downloading.

CHAPTER 23

产品版权

23.1 MU Vision Sensor 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 视觉传感器第 3 代产品（下称产品），本手册所描述内容仅适用于当前固件版本，新版本功能需要更新传感器固件，否则可能导致部分产品功能失效，版本更新不另行通告，请关注摩图科技官网。
- 应仔细阅读和理解本手册中的各项条款，否则可能导致产品无法正常工作，检测效果变差，甚至产品损坏。
- 在未经摩图科技确认及授权的情况下，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。
- 严禁任何组织或个人进行芯片内部代码拷贝、破解等侵权行为，芯片内部所有代码信息均归属于摩图科技所有，对于任何侵权行为，摩图科技将采取法律措施予以维权。
- 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，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措施予以维权。
- 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。
- 文本或图片中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。
- 本手册仅供指定公司客户或个人用户内部资料使用，在产品正式上市或发布之前，使用方应对本产品及手册采取保密措施，在未经摩图科技授权的情况下，不得将产品及本手册传阅至第三方公司或个人。

CHAPTER 24

软件许可

24.1 开源软件

24.1.1 Arduino

- Arduino is an open-source physical computing platform based on a simple I/O board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software on your computer (e.g. Flash, Processing and MaxMSP). The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free at https://www.arduino.cc/en/Main/Software
- Arduino is an open source project, supported by many. The Arduino team is composed of Massimo Banzi, David Cuartielles, Tom Igoe and David A. Mellis. Arduino uses GNU avr-gcc toolchain, GCC ARM Embedded toolchain, avr-libc, avrdude, bossac, openOCD and code from Processing and Wiring. Icon and about image designed by ToDo.

24.1.2 MicroPython

- MicroPython is written in C99 and the entire MicroPython core is available for general use under the very liberal MIT license. Most libraries and extension modules (some of which are from a third party) are also available under MIT or similar licenses.
- You can freely use and adapt MicroPython for personal use, in education, and in commercial products.

- MicroPython is developed in the open on GitHub and the source code is available at the GitHub page, and on the download page. Everyone is welcome to contribute to the project.

索引