大疆教育

ROBOMASTER EP
编程模块手册

目录

（点击章节标题，可跳转到相应位置）

总览 2
系统 3
灯效 14
底盘 20
云台 45
发射器 57
拓展机构 63
智能 75
装甲板 104
传感器 114
转接模块 119
移动设备 125
多媒体 127
控制语句 133
运算符 139
数据对象 158
函数体 184

总览

本文档为 RoboMaster EP 机器人编程指南，旨在帮助玩家快速入门。
RoboMaster 机器人实验室中的图形化编程模块多达百余个，用好它们可以实现 EP 的 PID 控制、机器视觉等特有功能，但这对于没有机器人及编程基础的新手有一定难度，故在此提供各模块的使用说明、简单例程等帮助玩家从 0 到 1。
建议玩家先总览全文，对机器人编程作基本了解，在创作过程中遇到困难时再有针对性地查阅，掌握各类模块的使用方法和编程技巧，玩出名堂。

1、模块类型

RoboMaster EP 实验室图形化编程模块主要分为以下五类：

模块类型	类型说明	模块示例
设置类	设置参数，如速率、频率、数量等	
执行类	控制 EP 执行相应指令	
事件类	事件触发模块，当满足触发条件时，会立刻跳出主函数，开始运行事件类模块内的程序	
信息类	信息获取模块，返回获取到的变量、列表等不同类型的数据	
条件类	条件判断模块，根据是否满足条件执行相应的指令	

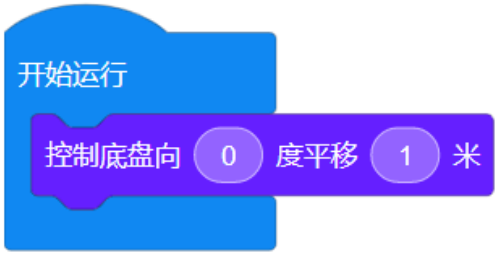
2、阻塞与非阻塞

	类型说明	模块示例
阻塞型模块	在阻塞型运行完之前，后续指令不得运行。“等待 xx”模块是典型的阻塞型	 
非阻塞型模块	非阻塞型模块运行时，不阻碍后续指令的运行	

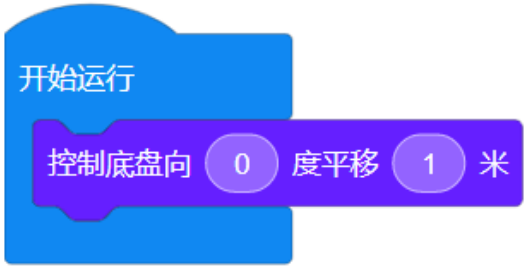
系统

1、开始运行

（1）含义：当机器人启动时首先执行的程序
（2）类型：内置模块
（3）范例：前进 1 米
让 RoboMaster EP 向前平移 1 米。



注意：
如果模块没有放置在“开始运行”内部（事件触发、函数模块除外），将不会执行。例如在如下例程中，RoboMaster EP 不会拍照。

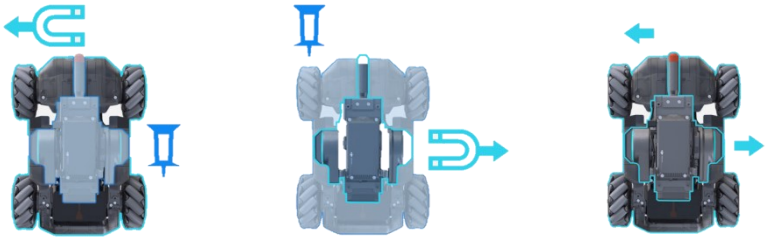


Python API：
Function: start()
Type: Main function

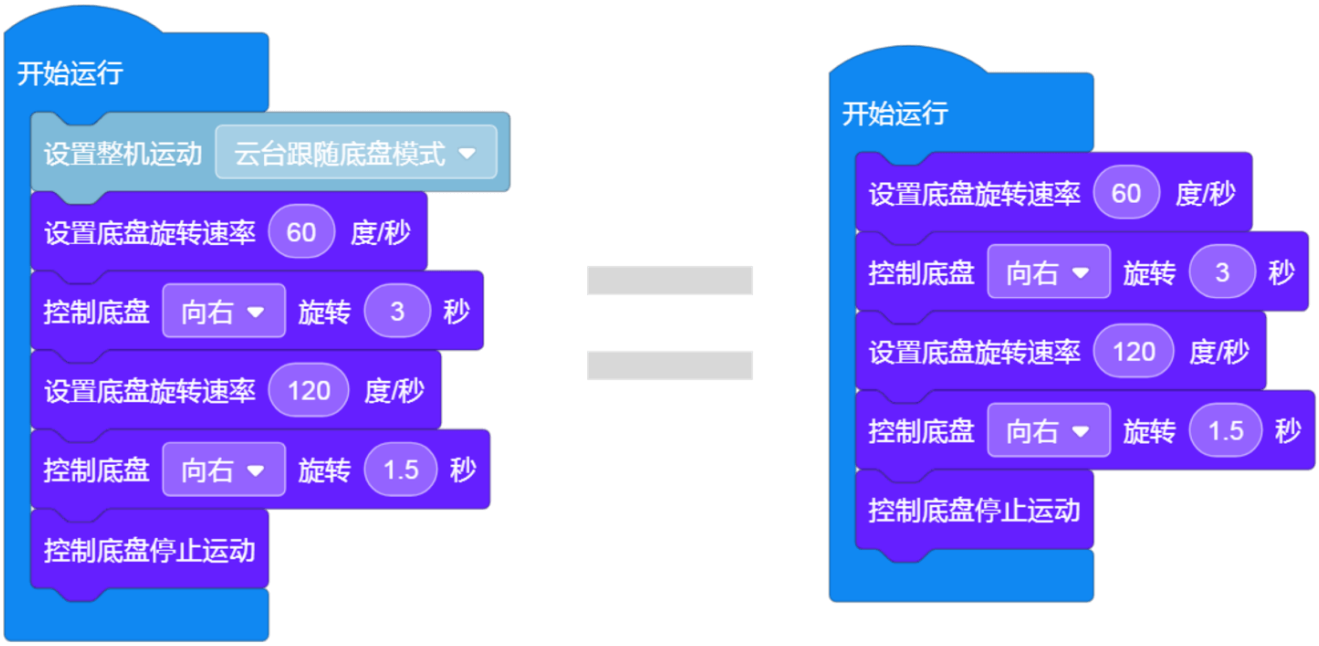
2、设置整机运动（云台跟随底盘模式）

（1）含义：设置整机运动的 3 种模式：

●云台跟随底盘模式：云台始终跟随底盘绕航向轴旋转
●底盘跟随云台模式：底盘始终跟随云台绕航向轴旋转
●自由模式：云台与底盘运动分离，互不影响



（2）类型：设置类
（3）范例：变速旋转、一起摇摆、反向旋转
①变速旋转
在“云台跟随底盘模式”下，只对底盘进行设置：
底盘以不同的旋转速率左右旋转，观察云台是否始终跟随，在底盘停止运动后仍与底盘前侧保持零夹角。



注意：
1）机器人的默认运动模式为“云台跟随底盘模式”，若在程序中未另行设置，则默认选择的是“云台跟随底盘模式”。
2）在“云台跟随底盘模式”下，如果底盘不旋转，云台无法单独左右旋转。
②一起摇摆
在“底盘跟随云台模式”下，只对云台进行设置：
云台绕航向轴来回旋转，观察底盘是否同样左右摇摆，并最终跟随云台回到初始位置。

注意：

在“底盘跟随云台模式”下，如果云台不旋转，底盘无法单独左右旋转。

③反向旋转

在“自由模式”下，控制云台和底盘各自向相反方向旋转。观察它们是否会相互干扰。

注意：

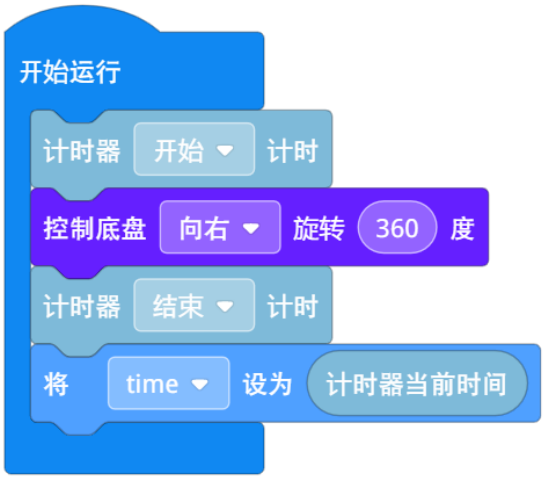
在“自由模式”下，“设置底盘以（x）度跟随云台”和“设置云台以（x）度跟随底盘”这两个模块不生效。

Python API:
Function: robot.set_mode(mode_enum)
Parameters:

- mode_enum(enum)
  - rm_define.robot_mode_gimbal_follow
  - rm_define.robot_mode_chassis_follow
  - rm_define.robot_mode_free

3、计时器（开始）计时

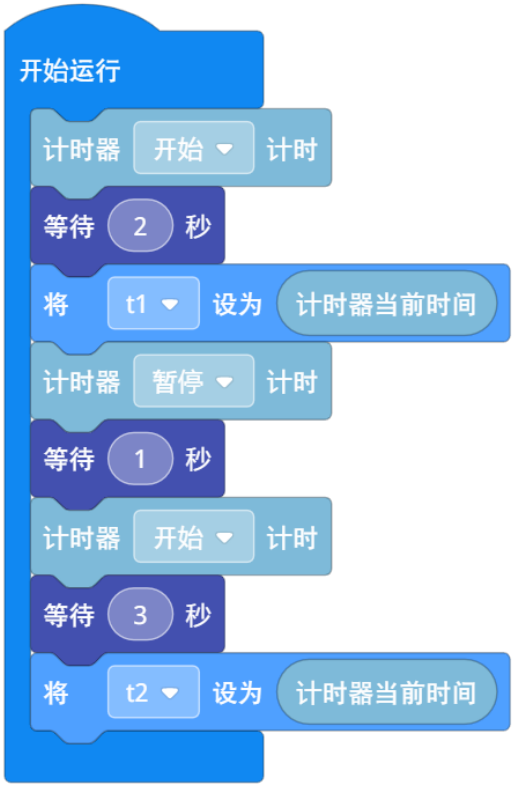
（1）含义：控制计时器开始、暂停或结束计时
（2）类型：执行类
（3）范例：旋转计时
获知底盘旋转 1 圈所需用时。



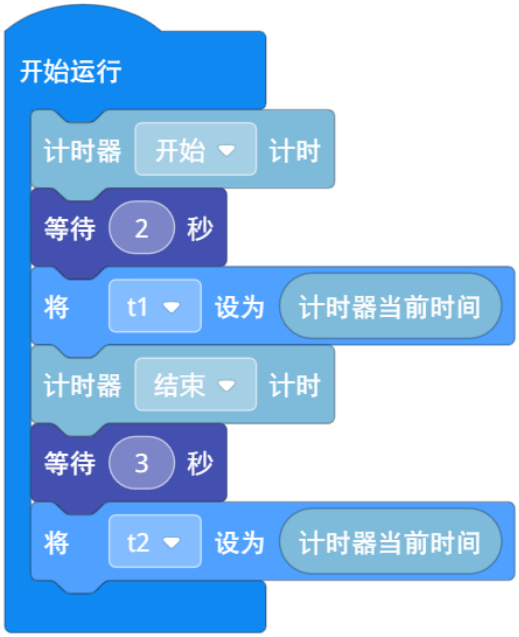
可以在 FPV 窗口观察：

注意：

1）“暂停”会保留计时器当前时间，再开始的话是继续计时。如以下范例中 t1=2，t2=5。



2）“结束"会停止计时，再开始的话会自动清零并重新开始计时。如以下范例中 t1=2，t2=0。



Python API：

Function: tools.timer_ctrl(behavior_enum)

Parameters:

- behavior_enum(enum):
  - rm_define.timer_start
  - rm_define.timer_stop
  - rm_define.timer_reset

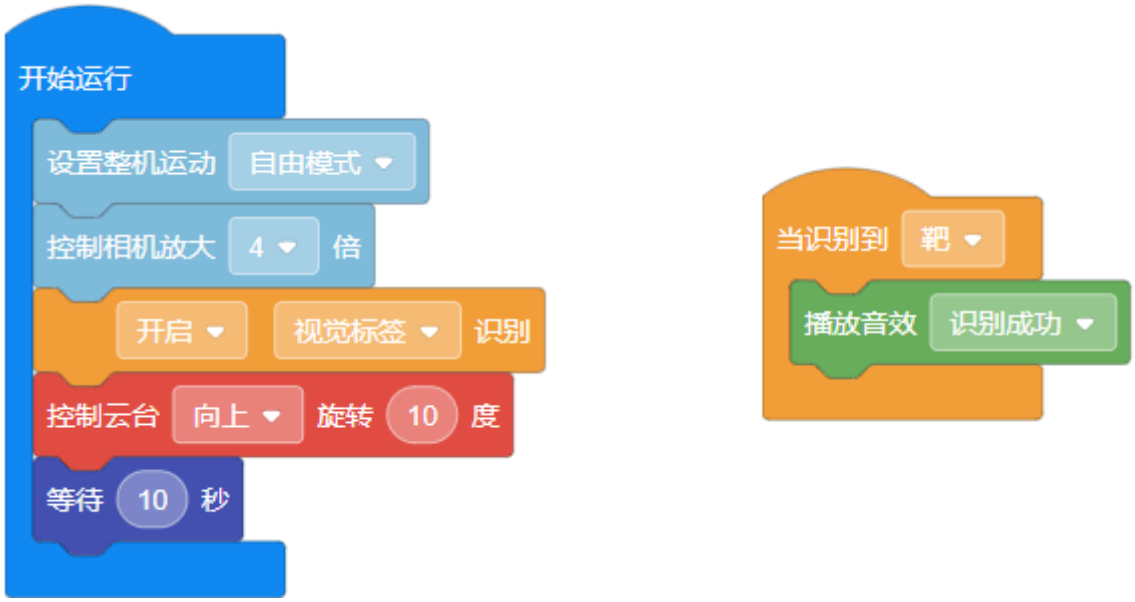
4、控制相机放大（1）倍

（1）含义：放大相机倍镜，让机器人的视觉识别距离更远，局部图像更清晰

（2）类型：执行类

（3）范例：相机画面放大

将视觉标签靶放置在距离机器人云台前 10 米远处，控制相机将画面放大 4 倍，依然可以准确识别。



点击运行前：

相机放大后：

Python API：

Function: media_ctrl.zoom_value_update(value)

Parameters:

- value (int): [1, 4]

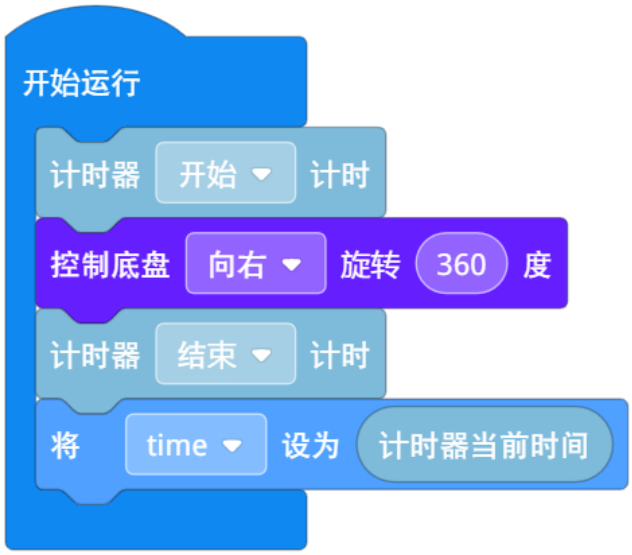
5、计时器当前时间

（1）含义：获取计时器从开始到当前时刻的用时，返回秒数

（2）类型：信息类（变量型数据）

（3）范例：旋转计时

底盘旋转 1 圈计时，通过变量获取返回的时间值。



可在 FPV 窗口观察：

Python API：

Function: tools.timer_current()

Return value:

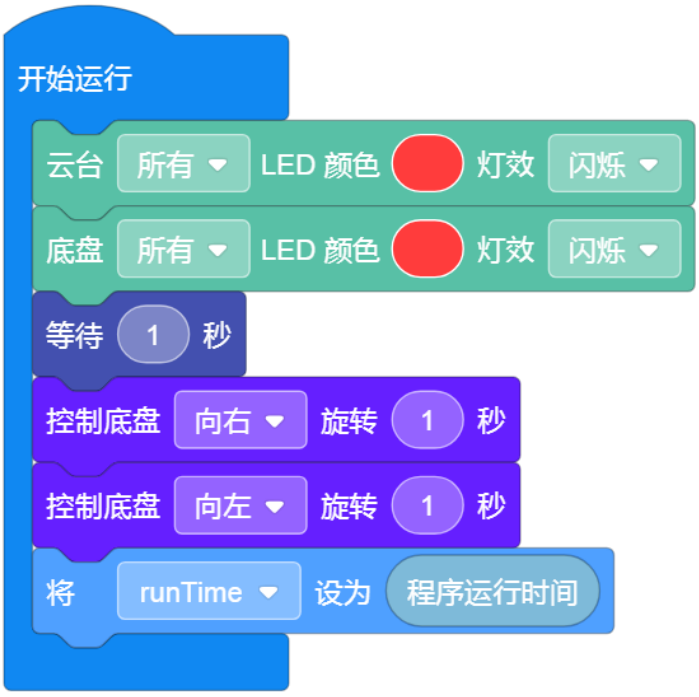
- time_stamp(float)

6、程序运行时间

（1）含义：获取程序运行用时，返回秒数

（2）类型：信息类（变量型数据）

（3）范例：程序运行时间



通过变量获取程序运行时间值，在 FPV 窗口观察。

Python API:
Function: tools.run_time_of_program()
Return value:

- time (float)

7、当前的（年）

（1）含义：获取当前的时间信息，如年/月/日/时/分/秒等。
（2）类型：信息类（变量型数据）
（3）范例：比较大小
如果当前月份数字大于日期，RoboMaster EP 会“点头”；如果当前月份数字小于或等于日期，EP 会“摇头”。

Python API：

Function: tools.get_localtime(time_enum)

Parameters:

- time_enum (enum):
  - ■rm_define.localtime_year
  - ■rm_define.localtime_month
  - ■rm_define.localtime_day
  - ■rm_define.localtime_hour
  - ■rm_define.localtime_minute
  - ■rm_define.localtime_second

Return value

- time (int)

8、整机运行时间

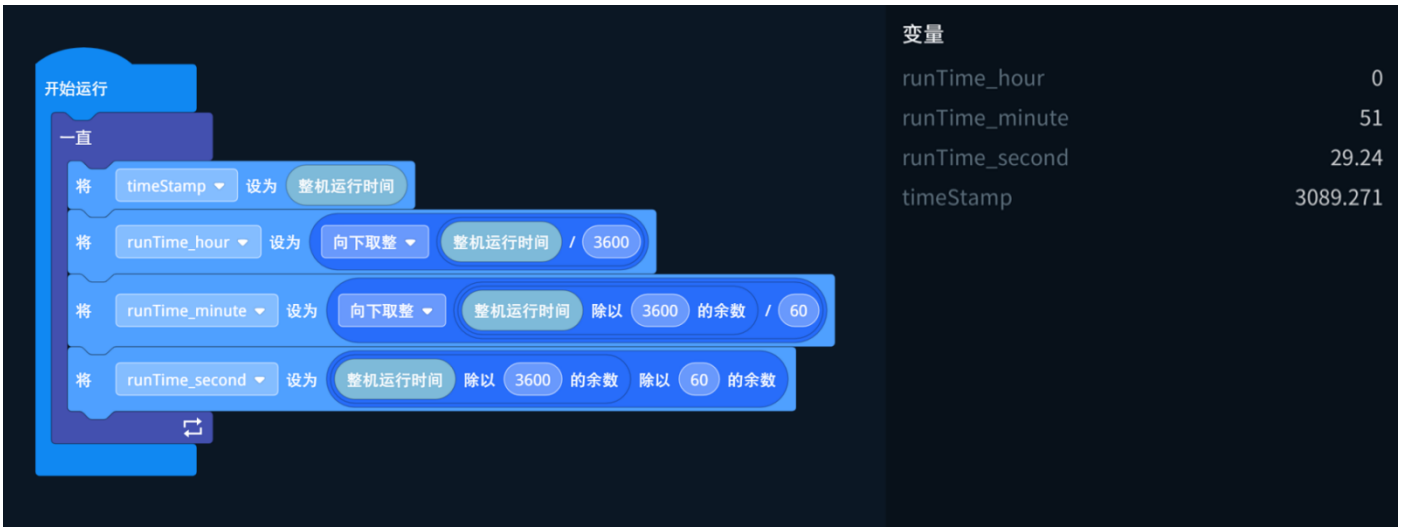
（1）含义：机器人启动时刻至今的时间间隔，返回累计的秒数

（2）类型：信息类（变量型数据）

（3）范例：整机运行时间计算

计算机器人自启动至今的运行时长（时、分、秒）。

可在 FPV 窗口观察数据变化。
当运行用时超过 1 小时（runTime_hour＞1），要注意休息。



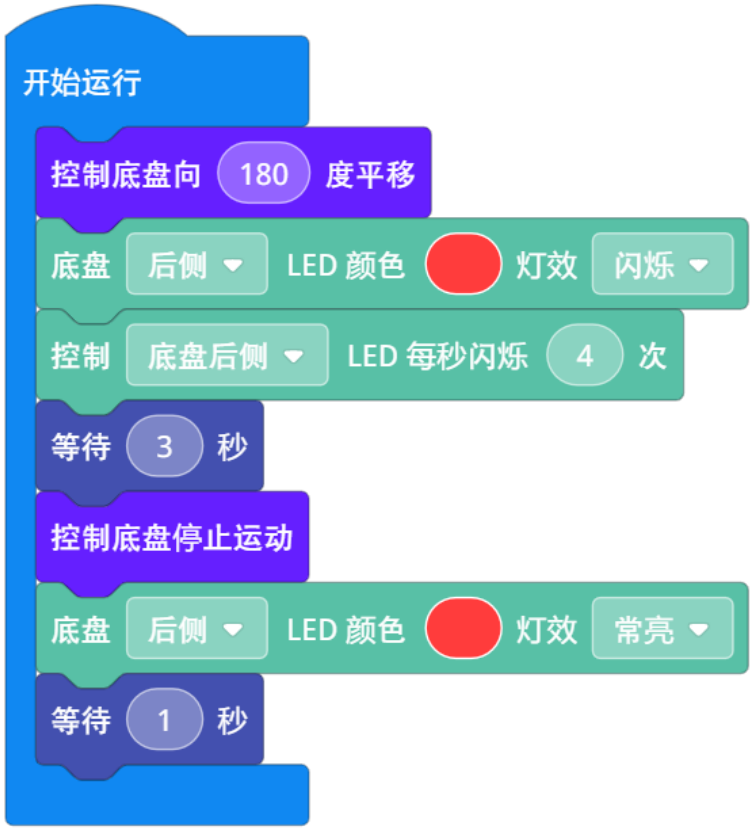
注意：
1）机器人的启动时刻是指上电时刻。
2）如果机器人在断电后重启，会重新累计运行时间。

Python API：
Function: tools.get_unixtime()
Return value:
- time (float)

灯效

1、控制（所有）LED 每秒闪烁（2）次

（1）含义：控制指定位置 LED 灯每秒闪烁次数
（2）类型：设置类
（3）范例：倒车灯
在 RoboMaster EP 倒车时，底盘后侧的 LED 灯以每秒 4 次的频率闪烁红灯示意。



Python API：
Function: led_ctrl.set_flash(armor_enum, frequency)
Parameters:

- armor_enum(enum):
  - rm_define.armor_all
  - rm_define.armor_bottom_front
  - rm_define.armor_bottom_back
  - rm_define.armor_bottom_left
  - rm_define.armor_bottom_right
  - rm_define.armor_top_left
  - rm_define.armor_top_right

- frequency(int): [1, 10]

2、底盘（所有）LED 颜色（青色）灯效（常亮）

（1）含义：控制底盘指定位置 LED 灯的颜色和灯效：

- 常亮，LED 灯保持点亮状态
- 熄灭，LED 灯关闭
- 呼吸，LED 灯明暗变化（由暗变亮再变暗）
- 闪烁，LED 灯以一定频率闪烁

（2）类型：执行类

（3）范例：流光溢彩

底盘所有 LED 熄灭 1 秒后，分别闪烁各色 LED 灯。

Python API：

Function: led_ctrl.set_bottom_led(armor_enum, r, g, b, led_effect_enum)

Parameters:

- armor_enum(enum):
  - rm_define.armor_bottom_all
  - rm_define.armor_bottom_front
  - rm_define.armor_bottom_back
  - rm_define.armor_bottom_left
  - rm_define.armor_bottom_right
- r(int): [0, 255]
- g(int): [0, 255]

- b(int): [0, 255]
- led_effect_enum(enum):
  - rm_define.effect_always_on
  - rm_define.effect_always_off
  - rm_define.effect_breath
  - rm_define.effect_flash

3、云台（所有）LED 颜色（青色）灯效（常亮）

（1）含义：设置云台指定位置 LED 灯的颜色和灯效：

- 常亮，LED 灯保持点亮状态
- 熄灭，LED 灯关闭
- 呼吸，LED 灯明暗变化（由暗变亮再变暗）
- 闪烁，LED 灯以一定频率闪烁
- 跑马灯，呈圆形排布的 8 颗 LED 灯顺时针滚动点亮

（2）类型：执行类

（3）范例：云台灯光秀

云台所有 LED 依次展示五类灯效。

Python API：
Function: led_ctrl.set_top_led(armor_enum, r, g, b, led_effect_enum)
Parameters:

- armor_enum(enum):
  - rm_define.armor_top_all
  - rm_define.armor_top_left
  - rm_define.armor_top_right
- r(int): [0, 255]
- g(int): [0, 255]
- b(int): [0, 255]
- led_effect_enum(enum):
  - rm_define.effect_always_on
  - rm_define.effect_always_off
  - rm_define.effect_breath
  - rm_define.effect_flash
  - rm_define.effect_marquee

4、云台（所有）LED 序号（1）灯效（常亮）

（1）含义：设置云台指定序号 LED 灯的亮灭，序号 1~8 分别对应云台两侧可独立控制的 8 颗 LED 灯
（2）类型：执行类
（3）范例：点亮单号灯
在云台所有 LED 灯熄灭后，依次点亮云台两侧奇数位的 LED 灯，再统一熄灭。

灯光序号呈逆时针排布：

注意：LED 灯支持多选。

Python API：

Function: led_ctrl.set_signle_led(armor_enum, led_index, led_effect_enum)

Parameters:

- armor_enum(enum):
  - rm_define.armor_top_all
  - rm_define.armor_top_left
  - rm_define.armor_top_right
- index(int/list): [1, 8]
- led_effect_enum(enum):
  - rm_define.effect_always_on
  - rm_define.effect_always_off

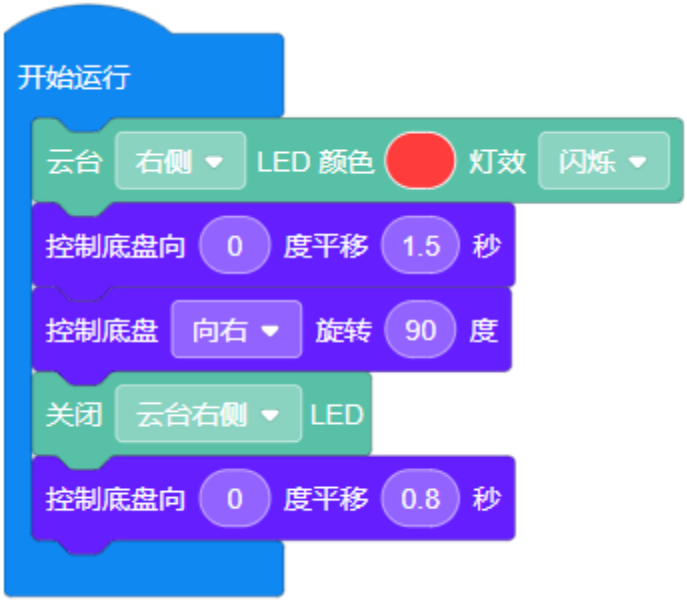
5、关闭（所有）LED

（1）含义：关闭指定位置的 LED 灯

（2）类型：执行类

（3）范例：转向灯

在 RoboMaster EP 右转之前，提前闪烁云台右侧 LED 灯示意；转向完成后关闭右转向灯。



Python API：
Function: led_ctrl.turn_off(armor_enum)
Parameters:

- armor_enum(enum)
  - rm_define.armor_all
  - rm_define.armor_bottom_front
  - rm_define.armor_bottom_back
  - rm_define.armor_bottom_left
  - rm_define.armor_bottom_right
  - rm_define.armor_top_left
  - rm_define.armor_top_right

6、（开启）弹道灯

（1）含义：控制发射器弹道灯的亮灭
（2）类型：执行类
（3）范例：弹道灯
在发射水弹时亮起弹道灯。

Python API：
Function: led_ctrl.gun_led_on()
led_ctrl.gun_led_off()

底盘

1、 设置（PWM_all）输出百分比为（7.5）

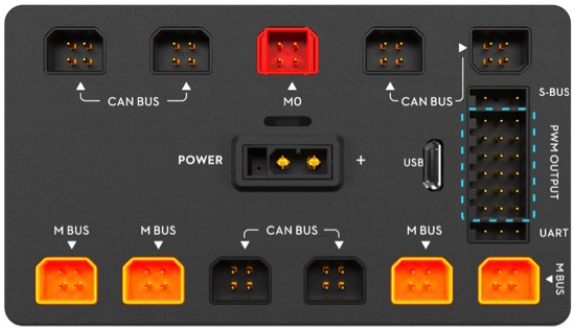
（1）含义：设置 PWM 输出百分比，数值越大，在某一周期内高电平的持续时间越长。该 PWM 基础频率为 50Hz。
（2）类型：设置类
（3）范例：灯的亮灭、舵机转动
①灯的亮灭
在任一 PWM 口外接 LED 灯条，控制其亮灭切换。

②舵机转动
在任一 PWM 口上外接舵机，控制其转动。

注意：

1）PWM 口位于底盘控制模块上，拿开底盘后侧的透明盖板即可看到。

从上至下共 6 个 PWM 口。



2）PWM 又称脉冲宽度调制，控制的是某一周期内高电平的持续时间，现广泛应用于 LED 灯、舵机等的控制上。

3）上电后，PWM 接口默认输出 7.5%占空比的信号，每次程序运行结束后，也会恢复默认的输出信号。

4）对灯条来说，PWM 输出百分比范围为 0%~100%，0 意味着灯最暗，100 意味着灯最亮。

5）对舵机来说，PWM 输出百分比范围为 2.5%～12.5%。因为大部分舵机的控制脉冲频率为 50 Hz，控制周期为 20 ms，可调节角度-90 °～ 90° 对应的高电平脉宽为 0.5 ms ～ 2.5 ms， 因此舵机占空比的控制范围便是 0.5/20~2.5/20，即 2.5%～12.5%。

玩家们可以根据自己想要控制的旋转角度设置舵机 PWM 的输出百分比。

脉冲宽度	舵机角度
	
	
	
	
	

Python API：
Function: chassis_ctrl.set_pwm_value(pwm_port_enum, output_percent)
Parameters:

- pwm_port_enum(enum)
  - rm_define.pwm_all
  - rm_define.pwm1
  - rm_define.pwm2
  - rm_define.pwm3
  - rm_define.pwm4
  - rm_define.pwm5
  - rm_define.pwm6
- output_percent(int): [0, 100]

2、（开启）底盘速度杆量叠加

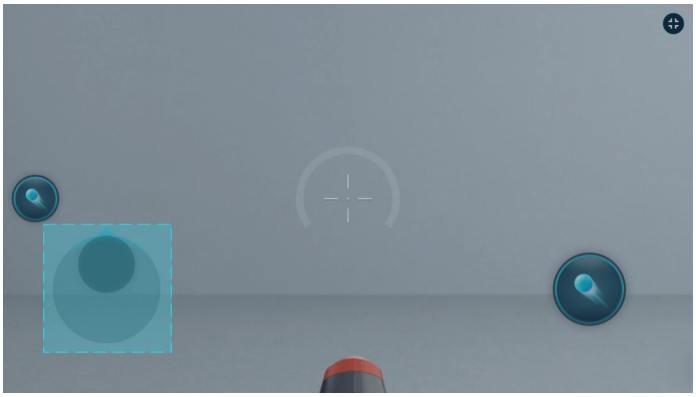
（1）含义：开启或关闭底盘速度杆量叠加
（2）类型：设置类
（3）范例：速度杆量叠加
在底盘自动运动的过程中，同时使用摇杆手控底盘旋转或平移，并且叠加平移速率。

注意：

1）如果不添加“开启底盘速度杆量叠加”模块，在运行程序时我们无法手控底盘；而添加此模块后，我们就可以对运行中的机器人进行移动控制，并且控制量会叠加。

2）杆量是指摇杆幅值的推动大小。杆量范围为 -1 ~ 1。

如图，在 FPV 界面中将虚拟摇杆推满，杆量便为 1。



3）速度杆量叠加是将底盘在程序中的速度与摇杆速度相加。

如图，在程序中设置底盘以 0.5m/s 向前平移，同时我们将杆量推满， 开启底盘速度杆量叠加后，机器人就会将两种控制数据“叠加”， 最终机器人将以（0.5+1*当前最大向前速度）m/s 的速率向前平移。

Python API：

Function: chassis_ctrl.enable_stick_overlay()

chassis_ctrl.disable_stick_overlay()

3、设置底盘以（0）度跟随云台

（1）含义：在“底盘跟随云台模式”下，当云台左右旋转时，底盘始终与云台保持指定夹角

（2）类型：设置类

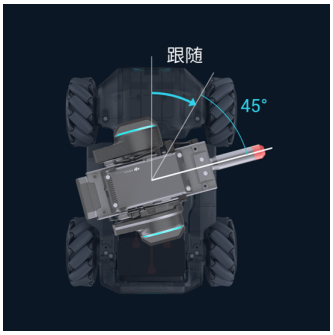
（3）范例：底盘跟随云台

底盘和云台间夹角不断增大，而后方向又保持统一。

注意：

1）在“云台跟随底盘模式”或“自由模式”下，此模块不生效。

2）0 度意味着没有夹角，底盘和云台绕航向轴的运动方向始终保持一致。



Python API：

Function: chassis_ctrl. set_follow_gimbal_offset(degree)

Parameters:

- degree(int): [-180, 180]°

4、设置底盘平移速率（0.5）米/秒

（1）含义：设置底盘平移速率，默认平移速率是 0.5 米/秒。数值越大，移动越快。
（2）类型：设置类
（3）范例：倒车减速
底盘以 1 米/秒向前平移 1.5 秒，再以 0.5 米/秒的速率向后平移 3 秒回到原地。

注意：
在为底盘设定较大的平移速率前，请确保底盘运动方向上没有障碍物。

Python API：
Function: chassis_ctrl.set_trans_speed(speed)
Parameters:

- speed(float): [0, 3.5] m/s

5、设置底盘旋转速率（30）度/秒

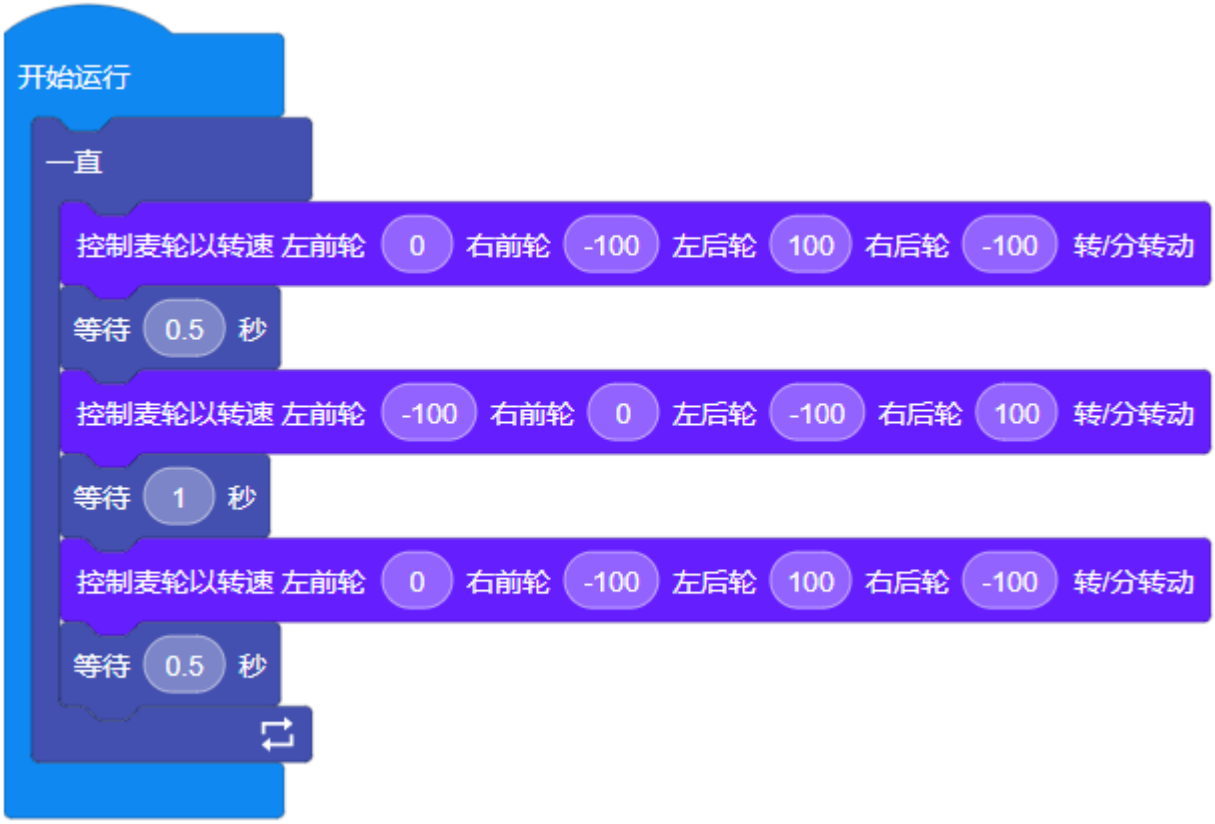
（1）含义：设置底盘平移速率，默认平移速率是 0.5 米/秒。数值越大，移动越快。
（2）类型：设置类
（3）范例：倒车减速
底盘以 1 米/秒向前平移 1.5 秒，再以 0.5 米/秒的速率向后平移 3 秒回到原地。

Python API：
Function: chassis_ctrl.set_rotate_speed(speed)
Parameters:

- speed(int): [0, 600] °/s

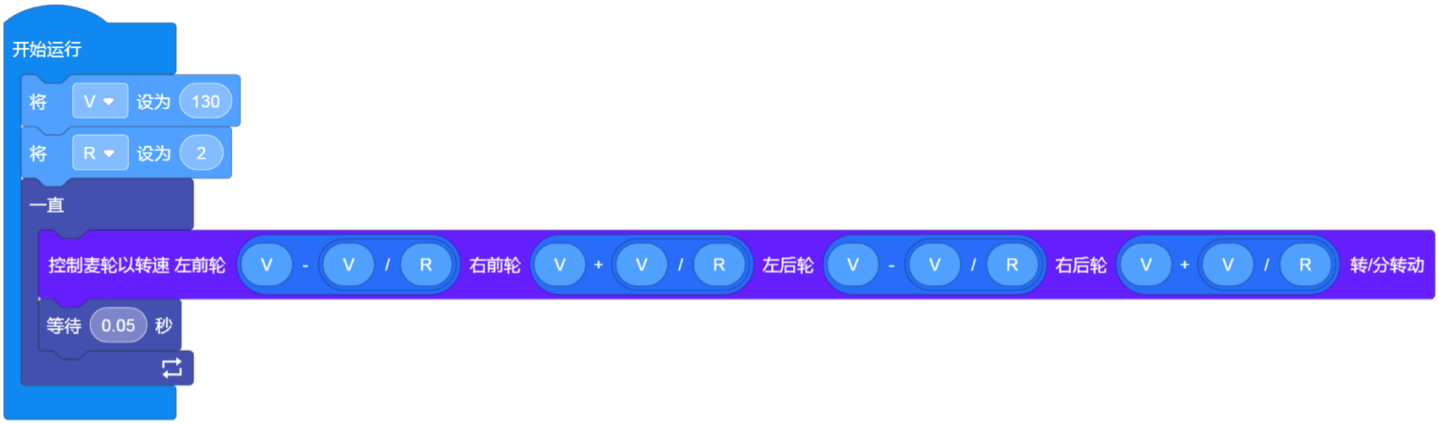
6、控制麦轮以转速 左前轮（100）右前轮（100）左后轮（100）右后轮（100）转/分转动

（1）含义：独立控制四个麦轮的转速，符合麦轮转动方向和速度的有效组合才会生效。
（2）类型：执行类
（3）范例：S 形倒退、跑圈
①S 形倒退
控制底盘以“S 形”轨迹倒退。



②跑圈

控制机器人跑圈。



注意：

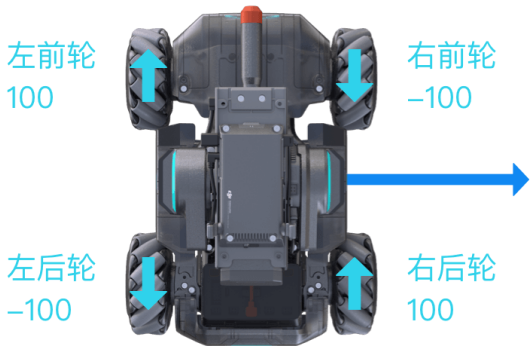
1）默认的麦轮组合“左前轮（100）右前轮（100）左后轮（100）右后轮 （100）转/分”是控制机器人向前平移。

2）如何判定组合是否有效?

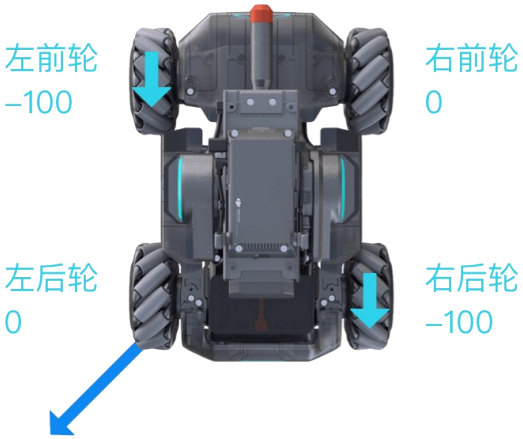
请用手推着机器人作期望动作，俯视观察每个麦轮的转动方向——向前转动为正值，向后转动为负值。

例：

当机器人向右平移时，左前轮和右后轮向前转动，所以转速值为正值；右前轮和左后轮向后转动，所以转速值为负值。



当机器人向左后平移时，左前轮和右后轮向后转动，所以转速值为负值；右前轮和左后轮不动，所以转速值为 0。



当机器人向左旋转时，右前轮和右后轮向前转动，所以转速值为正值；左前轮和左后轮向后转动，所以转速值为负值。

Python API：

Function: chassis_ctrl.set_wheel_speed(lf_speed, rf_speed, lr_speed, rr_speed)

Parameters:

- lf_speed(int): [-1000, 1000] rpm
- rf_speed(int): [-1000, 1000] rpm
- lr_speed(int): [-1000, 1000] rpm

- rr_speed(int): [-1000, 1000] rpm

7、控制底盘向（0）度平移

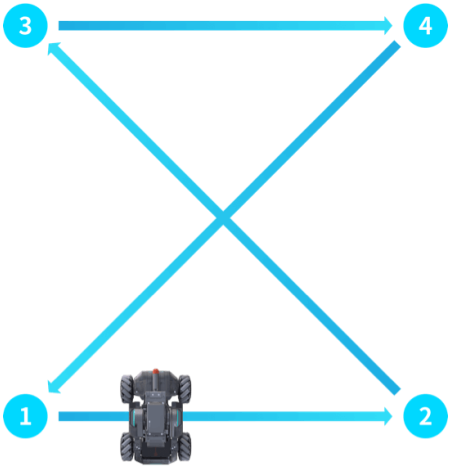
（1）含义：控制底盘向指定方向平移
（2）类型：执行类
（3）范例：往返运动
RoboMaster EP 向前平移一秒，掉头后再驶回出发点。

注意：
此模块会控制底盘向指定方向持续平移，直到收到“控制底盘停止运动”、“等待（x）秒”或其它控制底盘运动的指令。

Python API：
Function: chassis_ctrl.move(degree)
Parameters:
- degree (int): [-180, 180] °

8、控制底盘向（0）度平移（1）秒

（1）含义：控制底盘向指定方向平移指定时长
（2）类型：执行类
（3）范例：交叉平移
控制机器人沿“右-左前-右-左后”的交叉路径平移。



Python API：

Function: chassis_ctrl.move_with_time(degree, time)

Parameters:

- degree(int): [-180, 180] °
- time(float): [0, 20] s

9、控制底盘向（0）度平移（1）米

（1）含义：控制底盘向指定方向平移指定距离

（2）类型：执行类

（3）范例：“十”字形走位

控制机器人沿“前-后-左-右”的“十”字形路径平移。

Python API：

Function: chassis_ctrl.move_with_distance(degree, distance)

Parameters:

- degree(int): [-180, 180] °
- distance(float): [0, 5] m

10、控制底盘以（0.5）米/秒向（0）度平移

（1）含义：控制底盘以指定的平移速率向指定方向平移

（2）类型：执行类

（3）范例：回到原地

底盘以 1 米/秒向右前方平移，再以 0.5 米/秒向左后方平移，回到原地。

Python API：

Function: chassis_ctrl.move_degree_with_speed(speed, degree)

Parameters:

- speed(float): [0, 3.5] m/s
- degree(int): [-180, 180] °

11、控制底盘（向右）旋转

（1）含义：控制底盘向指定方向旋转

（2）类型：执行类

（3）范例：变速旋转

底盘向右旋转得越来越快。

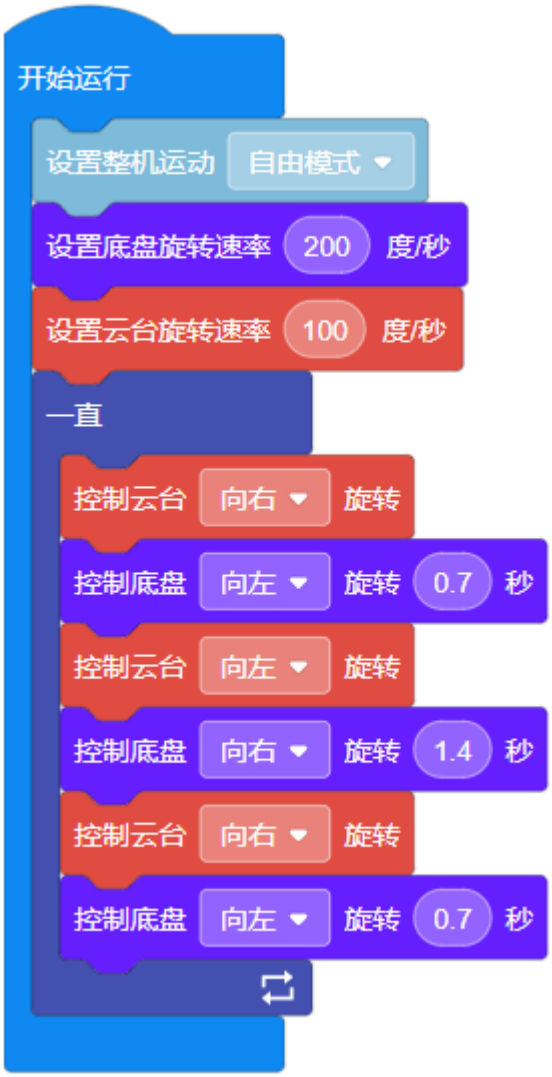
注意：
1）在“底盘跟随云台模式”下，此模块不生效。
2）在为底盘设定较大的旋转速率前，请确保机器人周围没有障碍物。
3）此模块会控制底盘向指定方向持续旋转，直到收到“控制底盘停止运动”、“等待（x）秒”或其它控制底盘运动的指令。

Python API：
Function: chassis_ctrl.rotate(direction_enum)
Parameters:

- direction_enum(enum):
  - rm_define.clockwise
  - rm_define.anticlockwise

12、控制底盘（向右）旋转（1）秒

（1）含义：控制底盘向指定方向旋转指定时长
（2）类型：执行类
（3）范例：云台、底盘交叉旋转



注意：
在“底盘跟随云台模式”下，此模块不生效。

Python API：
Function: chassis_ctrl.rotate_with_time(direction_enum, time)
Parameters:

- direction_enum(enum):
  - rm_define.clockwise
  - rm_define.anticlockwise
- time(float): [0, 20] s

13、控制底盘（向右）旋转（0）度

（1）含义：控制底盘向指定方向旋转指定角度
（2）类型：执行类
（3）范例：持续往返
RoboMaster EP 云台始终朝向外侧，巡回往返。

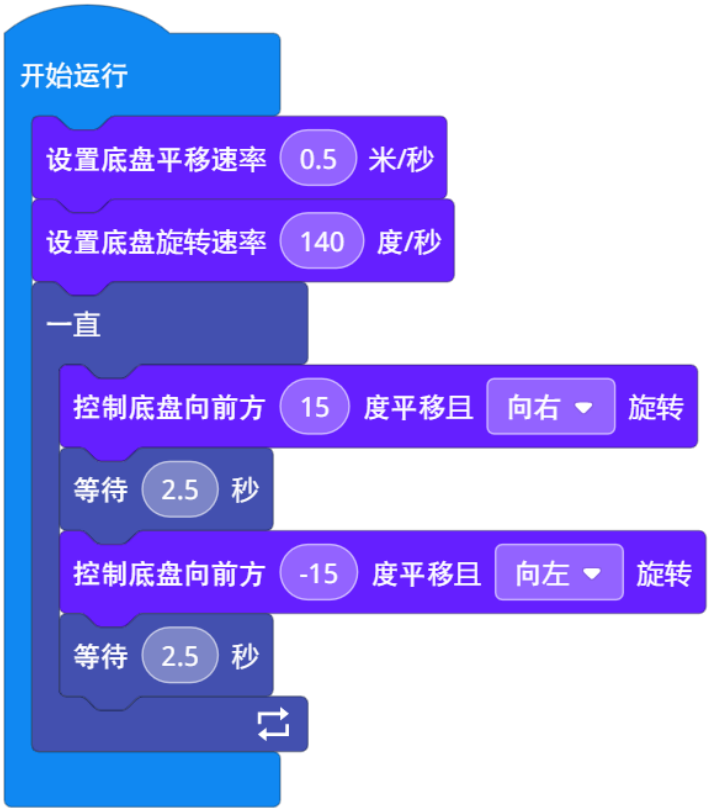
注意：
在“底盘跟随云台模式”下，此模块不生效。

Python API：
Function: chassis_ctrl.rotate_with_degree(direction_enum, degree)
Parameters:

- direction_enum(enum):
  - rm_define.clockwise
  - rm_define.anticlockwise
- degree(int): [0, 1800] °

14、控制底盘向前方（0）度平移且（向右）旋转

（1）含义：控制底盘向指定方向平移的同时做旋转运动
（2）类型：执行类
（3）范例：“8”字形运动
控制机器人绕“8”字形路径运动。

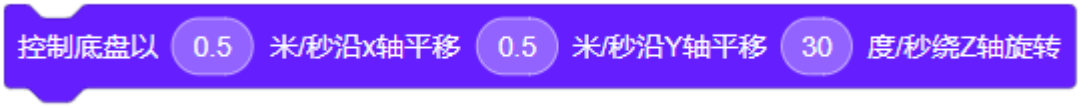


注意：
在“底盘跟随云台模式”下，此模块中“控制底盘旋转”的部分不生效，“控制底盘平移”的部分不受影响。

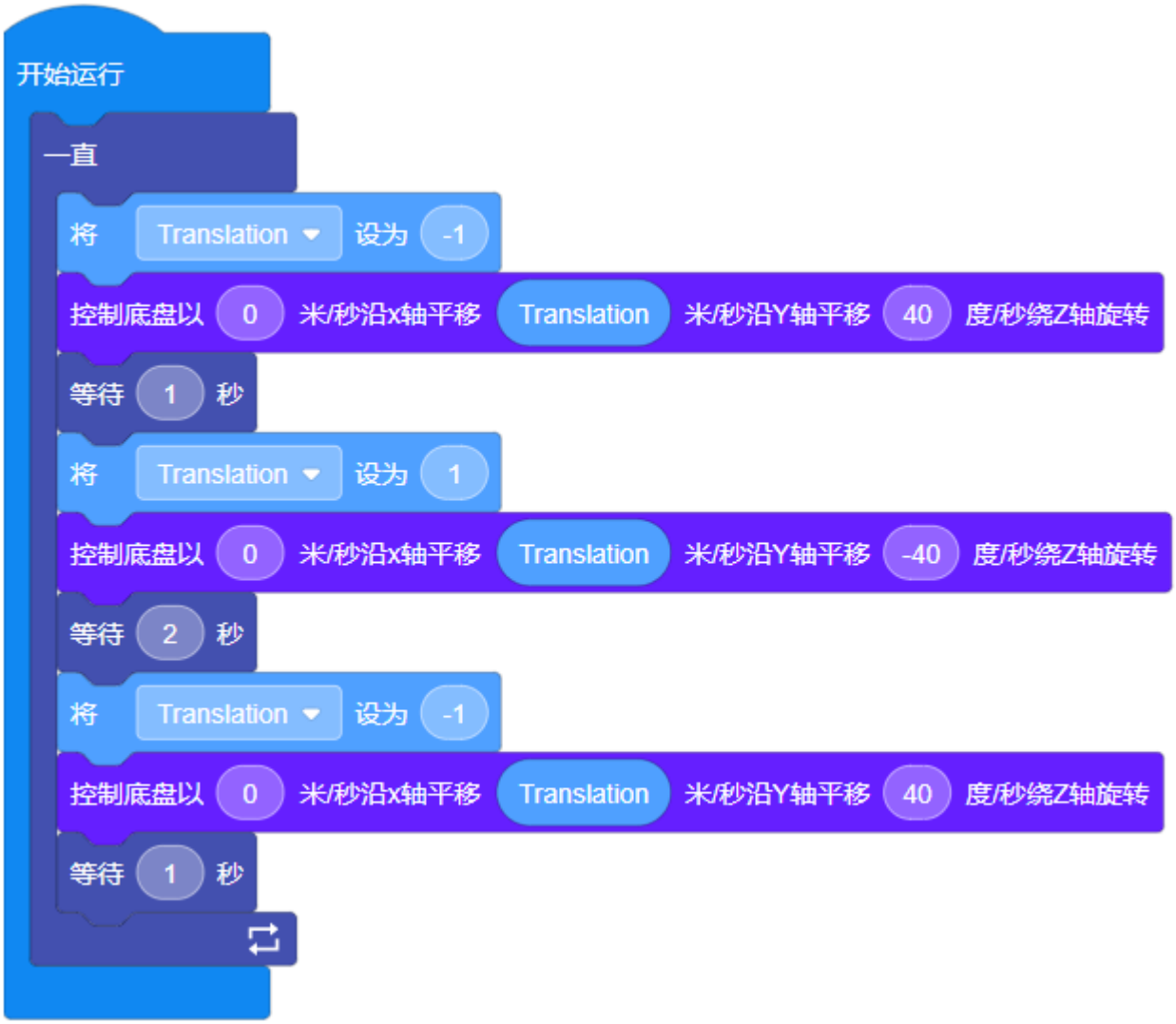
Python API：
Function: chassis_ctrl.move_and_rotate(degree, direction)
Parameters:

- degree(int): [-180, 180] °
- direction_enum(enum):
  - rm_define.clockwise
  - rm_define.anticlockwise

15、控制底盘以（0.5）米/秒沿 X 轴平移 （0.5）米/秒沿 Y 轴平移（30）度/秒绕 Z 轴旋转



（1）含义：控制底盘以指定速度在指定方向运动
（2）类型：执行类
（3）范例：刷锅运动



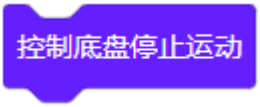
Python API：

Function: chassis_ctrl.move_with_speed(speed_x, speed_y, speed_rotation)

Parameters:

- speed_x(float): [0, 3.5] m/s
- speed_y(float): [0, 3.5] m/s
- speed_rotation(int): [-600, 600] °/s

16、控制底盘停止运动

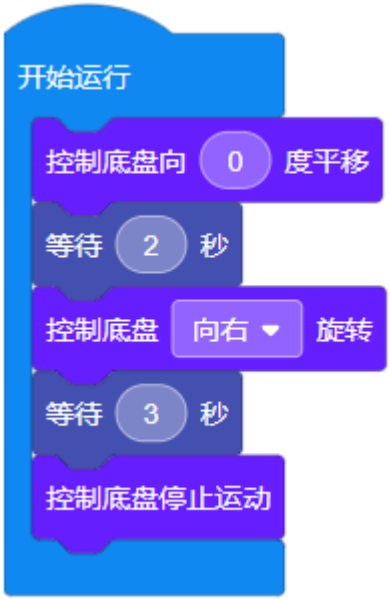


（1）含义：停止底盘的所有运动

（2）类型：执行类

（3）范例：EP 右转

控制底盘以默认速率向前平移 2 秒后右转弯，停止运动。



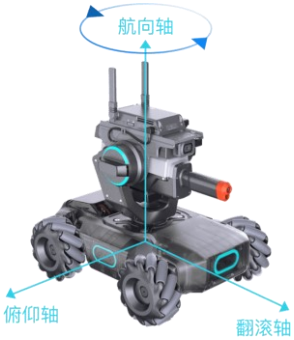
Python API：
Function: chassis_ctrl.stop()

17、底盘（航向轴）姿态角

（1）含义：以上电时刻底盘位置为基准，获取底盘当前在航向轴、俯仰轴或翻滚轴上的姿态角值
（2）类型：信息类（变量型数据）
（3）范例：转向示意
当用手控制机器人底盘向左旋转，云台黄光常亮；当控制机器人底盘向右旋转，云台蓝光常亮。

注意：

1、底盘绕航向轴：左右旋转



2、底盘绕俯仰轴：上下旋转

3、底盘绕翻滚轴：侧翻

Python API:

Function: chassis_ctrl.get_attitude(attitude_enum)

Parameters:

- attitude_enum(enum):
  - rm_define.chassis_yaw
  - rm_define.chassis_pitch
  - rm_define.chassis_roll

Return value:

- degree(float)

18、底盘当前位置（X 坐标）

（1）含义：获取底盘当前位置的坐标和朝向数据

（2）类型：信息类（变量型数据）

（3）范例：当前位置信息

用手推动机器人前进、后退、左右旋转，观察底盘在当前位置数值的变化。

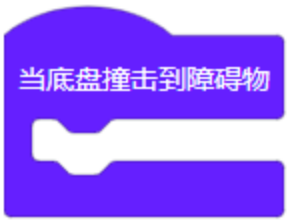
可以在 FPV 窗口观察：

注意：
因为底盘处于闭环控制状态，所以用手推动它时会感觉有阻力，这是正常的。

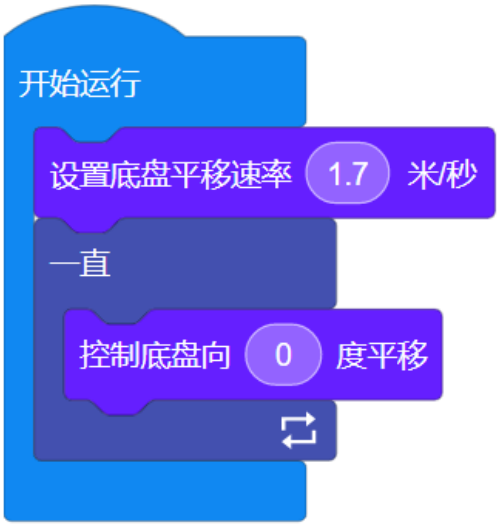
Python API：
Function: chassis_ctrl.get_position_based_power_on(action_enum)
Parameters:
- action_enum(enum):
  - rm_define.chassis_forward
  - rm_define.chassis_translation
  - rm_define.chassis_rotate

Return value:
- position(float)

19、当底盘撞击到障碍物



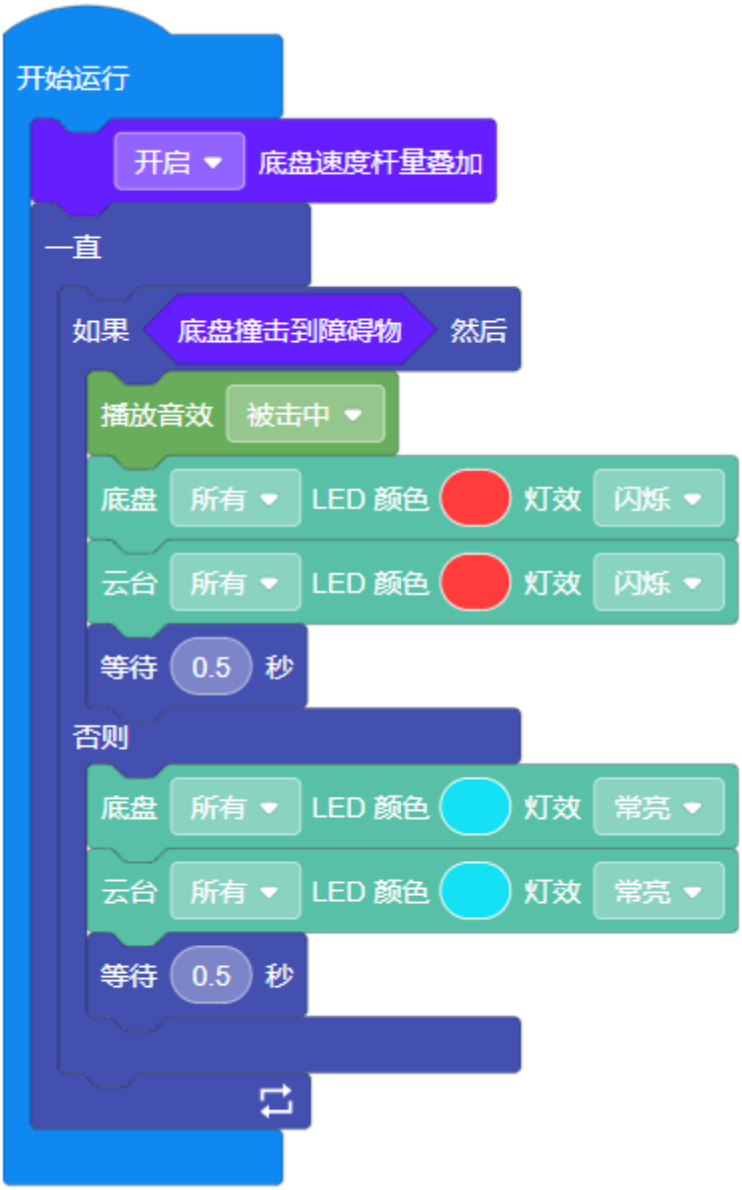
（1）含义：在行驶过程中，当底盘撞击到人、桌腿等障碍物时，运行本模块内程序
（2）类型：事件类
（3）范例：自我保护
当底盘撞击到障碍物时，启动自我保护机制——后退并停止程序运行。



Python API：
Function: def chassis_impact_detection(msg)
Type: Event callback

20、底盘撞击到障碍物

（1）含义：在行驶过程中，检测到底盘撞击到人、桌腿等障碍物时会返回“真”，否则返回“假”
（2）返回值：布尔型
（3）范例：危险警报
在 FPV 界面操控机器人行驶的过程中，如果底盘撞击到障碍物，则播放“被击中”音效，底盘和云台所有 LED 红光闪烁；否则底盘和云台以默认色常亮。



Python API：

Function:chassis_ctrl.is_impact()

Return value:

● impact_status(bool)

云台

1、（开启）云台速度杆量叠加

（1）含义：开启或关闭云台速度杆量叠加
（2）类型：设置类
（3）范例：半自动防御
在底盘持续左右旋转的同时，在 FPV 界面手动控制云台自由旋转，以便随时瞄准射击。

Python API：
Function: gimbal_ctrl.enable_stick_overlay()
gimbal_ctrl.disable_stick_overlay()

2、设置云台以（0）度跟随底盘

（1）含义：在“云台跟随底盘模式”下，当底盘左右旋转时，云台始终与底盘保持指定夹角
（2）类型：设置类
（3）范例：云台跟随底盘
底盘静止，云台向右旋转 45 度，再向左旋转到- 90 度位置后回中。

注意：
1、在“底盘跟随云台模式”或“自由模式”下，此模块不生效。
2、0 度意味着没有夹角，云台和底盘绕航向轴的运动方向始终保持一致。
3、此例程是“控制云台左右旋转”的另一种实现方法。

Python API：
Function: gimbal_ctrl.set_follow_chassis_offset(degree)
Parameters:
- degree(int): [-180, 180] °

3、设置云台旋转速率（30）度/秒

（1）含义：设置云台旋转速率，默认速率是 30 度/秒。数值越大，旋转越快。
（2）类型：设置类
（3）范例：变速旋转
底盘跟随云台一起向右旋转，且旋转得越来越快。

注意：
在为云台设定较大的旋转速率前，请确保机器人周围没有障碍物。

Python API：
Function: gimbal_ctrl.set_rotate_speed(speed)
Parameters:

- speed(float): [0, 540] °/s

4、控制云台（回中）

（1）含义：控制云台动作：

- 回中：云台回到航向轴和俯仰轴的初始位置
- 停止运动：云台停止运动，但仍处于受控状态
- 休眠：云台断电
- 唤醒：云台重新通电

（2）类型：执行类
（3）范例：云台复活、空中悬停
①云台复活
休眠时云台疲软，电机无力矩输出，可以被随意推动；唤醒后云台恢复控制，自动回中；接着云台会旋转到指定位置，最后回中。

注意：

在“云台跟随底盘模式”下，“控制云台回中” 模块不生效。

②空中悬停

云台向上旋转 0.5 秒后静止 2 秒，而后向下旋转 0.5 秒回到原位，停止运动。

Python API：

Function: gimbal_ctrl.recenter()

gimbal_ctrl.stop()

```
gimbal_ctrl.suspend()
gimbal_ctrl.resume()
```

5、控制云台（向上）旋转

（1）含义：控制云台向指定方向旋转
（2）类型：执行类
（3）范例：底盘、云台反向旋转

注意：
1）在“云台跟随底盘模式”下，“控制云台（向左/向右）旋转”不生效，“控制云台（向上/向下）旋转”不受影响。
2）此模块会控制云台向指定方向持续旋转，直到收到“控制云台停止运动”、“等待（x）秒”或其它控制云台运动的指令。

Python API：
Function: gimbal_ctrl.rotate(direction_enum)
Parameters:

- direction_enum(enum):
  - rm_define.gimbal_up
  - rm_define.gimbal_down

- rm_define.gimbal_left
- rm_define.gimbal_right

6、控制云台（向上）旋转（0）度

（1）含义：控制云台向指定方向旋转指定角度
（2）类型：执行类
（3）范例：云台转动
云台随着音效左右旋转。

注意：
在“云台跟随底盘模式”下，“控制云台（向左/向右）旋转（x）度”不生效，“控制云台（向上/向下）旋转（x）度”不受影响。

Python API：
Function: gimbal_ctrl.rotate_with_degree(direction_enum, degree)
Parameters:

- direction_enum(enum):
  - rm_define.gimbal_up
  - rm_define.gimbal_down
  - rm_define.gimbal_left
  - rm_define.gimbal_right
- degree(int): [0, 55]°

7、控制云台绕航向轴旋转到（0）度

（1）含义：控制云台绕航向轴旋转到指定位置
（2）类型：执行类
（3）范例：回马枪
底盘持续向前平移，云台在底盘后侧来回扫射。

注意：
1）在“云台跟随底盘模式”下，此模块不生效。
2）云台绕航向轴实现左右旋转，绕俯仰轴实现上下旋转。

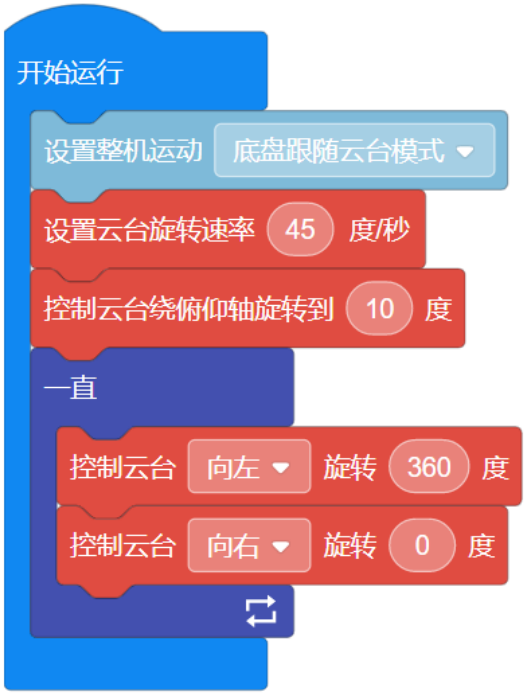
Python API：
Function: gimbal_ctrl.yaw_ctrl(degree)
Parameters:
- degree(int): [-250, 250]°

8、控制云台绕俯仰轴旋转到（0）度

（1）含义：控制云台绕俯仰轴旋转到指定位置
（2）类型：执行类
（3）范例：开机模拟、箭头标签跟随
①开机模拟

模拟机器人开机动作。

②箭头标签跟随



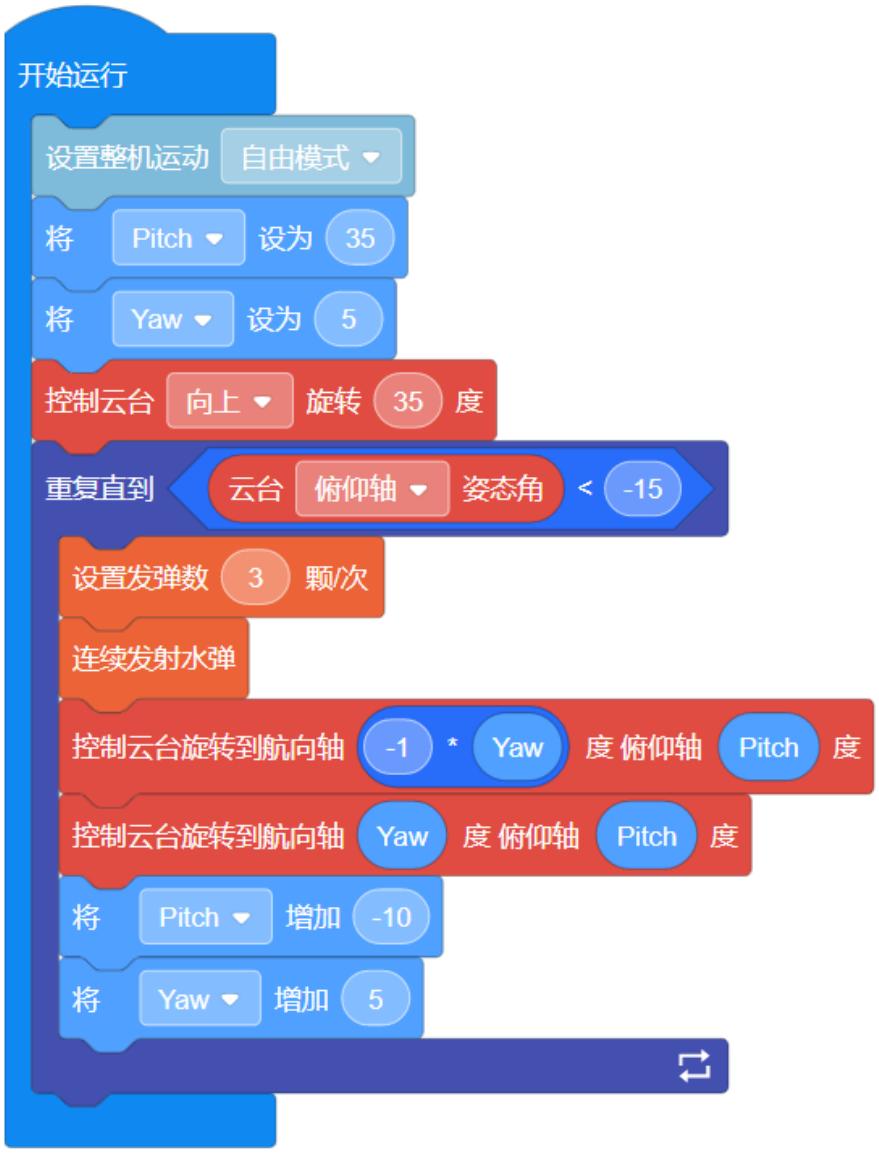
注意：
云台绕俯仰轴实现上下旋转，绕航向轴实现左右旋转。

Python API：
Function: gimbal_ctrl.pitch_ctrl(degree)
Parameters:

- degree(int): [-20, 35]°

9、控制云台旋转到航向轴（0）度 俯仰轴（0）度

（1）含义：控制云台旋转到指定角度位置
（2）类型：执行类
（3）范例：逐层扫射



注意：

1）在“云台跟随底盘模式”下，“控制云台旋转到航向轴（x）度”部分不生效，“旋转到俯仰轴（x）度”部分不受影响。

2）

“控制云台（向上/下/左/右）旋转（x）度”是相对位置，基于云台当前方位。

3）

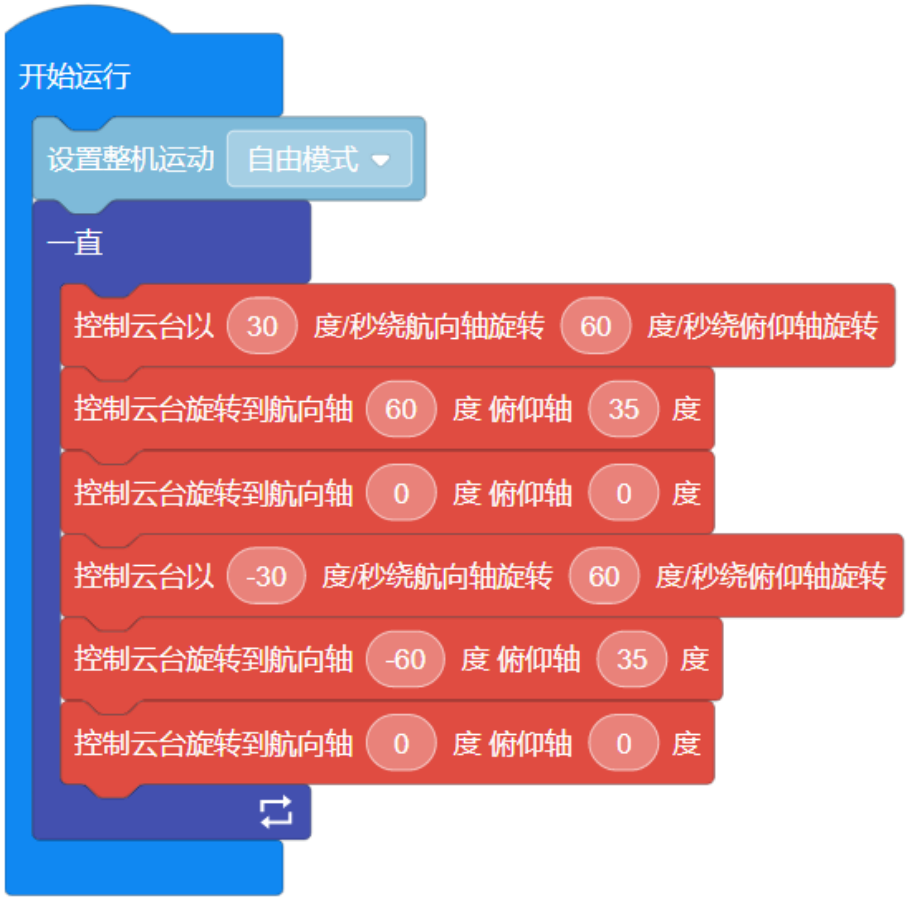
“控制云台绕航向轴旋转到（x）度”、“控制云台绕俯仰轴旋转到（x）度” 、 “控制云台旋转到航向轴（x）度 俯仰轴（x）度”是绝对位置，基于底盘当前方位。

Python API：
Function: gimbal_ctrl.angle_ctrl(yaw_degree, pitch_degree)
Parameters:

- yaw_degree (int): [-250, 250]°
- pitch_degree (int): [-20, 35]°

10、控制云台以（30）度/秒绕航向轴旋转 （30）度/秒绕俯仰轴旋转

（1）含义：控制云台以指定旋转速度同时绕航向轴、俯仰轴旋转
（2）类型：执行类
（3）范例：灵动的脖子



注意：
速度输入值的正负号代表着云台转动方向。
对航向轴而言：正值意味着向右转动，负值意味着向左转动。
对俯仰轴而言：正值意味着向上转动，负值意味着向下转动。

Python API：
Function: gimbal_ctrl.rotate_with_speed(yaw_speed, pitch_speed)
Parameters:

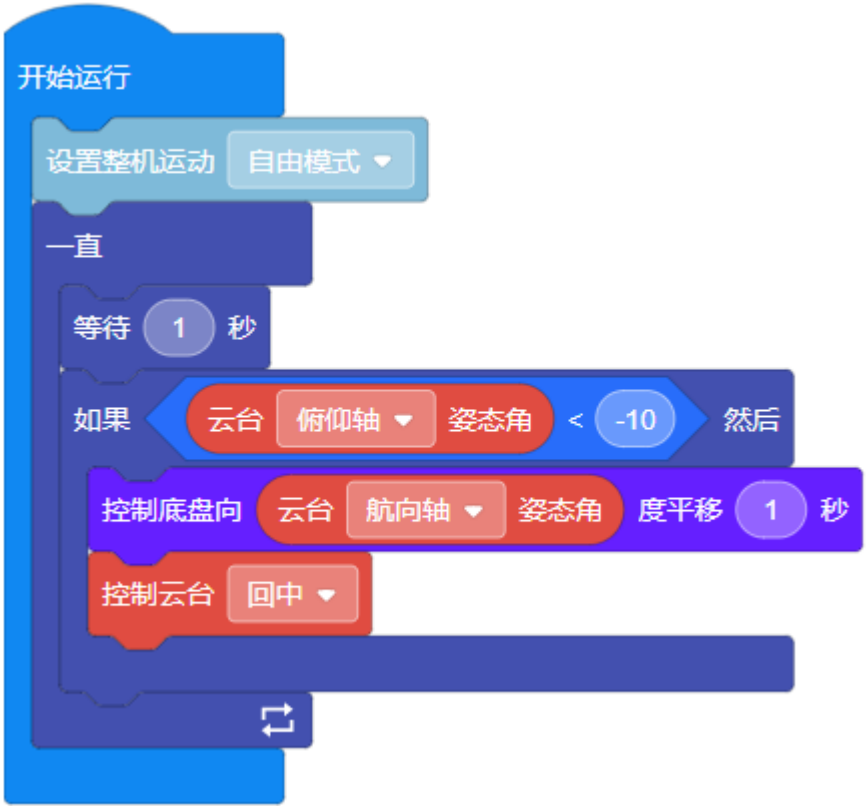
- yaw_speed(float): [-360, 360]°/s

- pitch_speed(float): [-360, 360]°/s

11、云台（航向轴）姿态角

（1）含义：获取云台当前在航向轴或俯仰轴上的姿态角值
（2）类型：信息类（变量型数据）
（3）范例：朝云台方向运动

拨动云台指向目标方向，按下云台，机器人将向目标方向前进 1 秒。



Python API：
Function: gimbal_ctrl.get_axis_angle(axis_enum)
Parameters:
- axis_enum (enum):
  - rm_define.gimbal_axis_yaw
  - rm_define.gimbal_axis_pitch

Return value:
- degree(int)

发射器

1、设置发弹数（1）颗/次

（1）含义：设置发弹数，即每次射出的水弹颗数
（2）类型：设置类
（3）范例：连发 6 颗

注意：
每次最多发射 8 颗水弹。

Python API：
Function: gun_ctrl.set_fire_count(count)
Parameters:

- count(int): [1, 8]

2、单次发射水弹

（1）含义：控制发射器只发射一次水弹
（2）类型：执行类、阻塞型
（3）范例：单次喷射

注意：

1）默认状态是一次发弹一颗。

2）搭配“设置发弹数（x）颗/次”模块可以设置一次连发几颗。

Python API：

Function: gun_ctrl.fire_once()

3、连续发射水弹

（1）含义：控制发射器持续发射水弹

（2）类型：执行类、非阻塞型

（3）范例：环绕扫射

控制云台绕航向轴旋转扫射。

注意：

1）默认频率是每秒发射一次，每次发弹一颗。

2）连续发射水弹是非阻塞型模块，也就是它会持续发射，直到遇到“停止发射”指令或例程结束的情况。

3）单次发射水弹和连续发射水弹的区别是：

5 秒内不间断地发射水弹

发射一次水弹后安静等待 5 秒

Python API：
Function: gun_ctrl.fire_continuous()

4、停止发射水弹

（1）含义：停止发射水弹
（2）类型：执行类
（3）范例：停止射击

注意：
1）因为“单次发射水弹”模块是阻塞型的，所以“停止发射水弹”指令对它不起作用。
2）“停止发射水弹”模块只对“连续发射水弹”有限制。

Python API：
Function: gun_ctrl.stop()

5、设置红外发射频率（1）次/秒

（1）含义：设置红外光束的发射频率，即每秒射出的红外光束次数。
（2）类型：设置类

（3）范例：连射 8 次红外光束

控制发射器在两秒内持续发射红外光束，每秒发射 4 次。

注意：

1）每秒最多发射 8 次红外光束。

2）此模块对“单次发射红外光束”模块不起作用。

3）在实验室编程时，发射类操作不受系统设置影响。比如当前设置-控制里"射击类型"选择的是红外光束，但编程时使用的是发射水弹模块，程序运行时依然会发射水弹。而在实验室之外与系统设置不一样的设计将不生效。

6、单次发射红外光束

（1）含义：控制发射器只发射一次红外光束。

（2）类型：执行类、阻塞型

（3）范例：

在发射一次红外光束的同时亮起弹道灯。

注意：

使用“单次发射红外光束”、“连续发射红外光束”模块发射红外光束时，本身没有任何可见效果。

7、连续发射红外光束

（1）含义：控制发射器持续发射红外光束。

（2）类型：执行类、非阻塞型

（3）范例：红外扫射

云台绕航向轴旋转，发射器持续扫射红外光束。

注意：

1）默认频率是每秒发射一次红外光束。

2）“连续发射红外光束”是非阻塞型模块，也就是它会连续发射，直到遇到“停止发射红外光束”指令或程序结束的情况。

3）“单次发射红外光束”和“连续发射红外光束”的区别是：

发射一次红外光束后安静等待 5 秒

5 秒内不间断地发射五次红外光束

8、停止发射红外光束

（1）含义：停止发射红外光束。
（2）类型：执行类
（3）范例：停止射击
在发射器连续发射 5 秒红外光束后停止。

注意：
1）因为“单次发射红外光束”模块是阻塞型的，所以“停止发射红外光束”指令对它不起作用。
2）“停止发射红外光束”只对“连续发射红外光束”有限制。

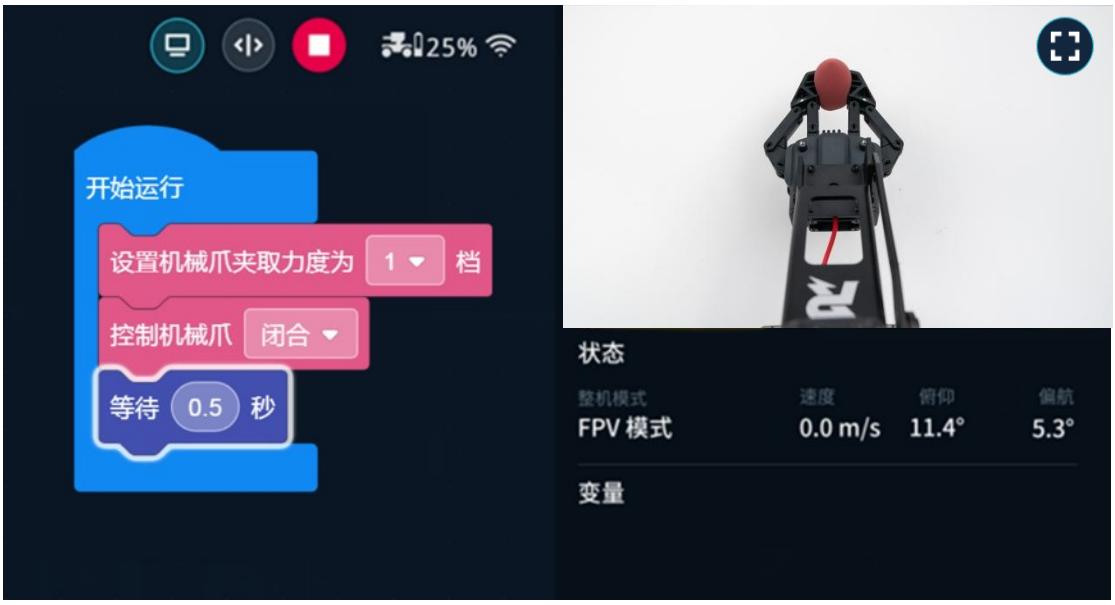
拓展机构

1、设置机械爪夹取力度为（1）档

（1）含义：设置机械爪夹取力度，档位越高，夹取力越大。
（2）类型：设置类
（3）范例：夹取形变
在机械爪里放置一个软体物质，不断调大机械爪夹取力度，被夹取物的形变程度会越来越明显。可以通过 FPV 观察。

开始运行前，被夹取物无形变：

夹取力度为 1 档时，被夹取物略有形变：



夹取力度为 4 档时，被夹取物形变明显：

注意：

1）设置机械爪夹取力度的档位越高，夹取力越大，执行速度也越大。

2）机械爪夹取不同物品所需要的力度不同，比如夹取鸡蛋壳的力度建议设置为低档位，夹取水杯的力度建议为高档位，请根据实际需要和效果进行调试。

2、控制机械爪（张开）

（1）含义：控制机械爪张开、闭合或停止运动。

（2）类型：执行类

（3）范例：收放自如

控制机械爪先张开后闭合。

注意：

1）“控制机械爪（张开、闭合）”是趋势运动，所以需要搭配“等待（1）秒”等时间控制模块使用。

2）机械爪运动达到限位后将保持不变。

3、机械爪已完全（张开）

（1）含义：机械爪当前状态满足条件时返回“真”，否则返回“假”。

（2）返回值：布尔型

（3）范例：机械爪张开用时

获知机械爪从完全闭合到完全张开用时。

可以在 FPV 窗口观察到大概用时是 2.1 秒。

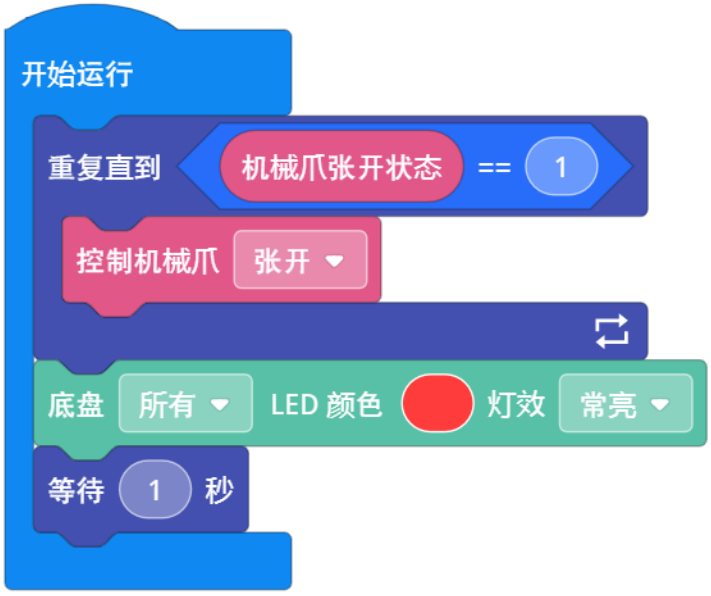
4、机械爪闭合状态

（1）含义：获知机械爪是否为闭合状态，返回值“1”表示已闭合，“0”表示未完全闭合。

（2）类型：信息类
（3）范例：机械爪闭合
根据机械爪当前状态控制其运动到完全闭合。

5、机械爪张开状态

（1）含义：获知机械爪是否为张开状态，返回值“1”表示已张开，“0”表示未完全张开。
（2）类型：信息类
（3）范例：机械爪张开
控制机械爪运动到完全张开状态，底盘红灯常亮一秒示意。



6、控制机械臂向（前）移动（50）毫米

（1）含义：以机器人机体坐标系为参照系，控制机械臂向指定方向移动设定距离。
（2）类型：执行类
（3）范例：机械臂舞
控制机械臂前、后、上、下灵活运动。

注意：
1）此机器人的机械臂属于并联机械臂，结构较复杂但承载力大、刚性强、精度高，由两个安装在机械臂底部的舵机驱动。
2）机器人机械臂的末端执行器是机械爪，通过控制机械臂的连杆结构，使其沿着机器人前进方向前后运动、沿机体垂直方向上下运动，从而将机械爪灵活、准确地送达指定位置。

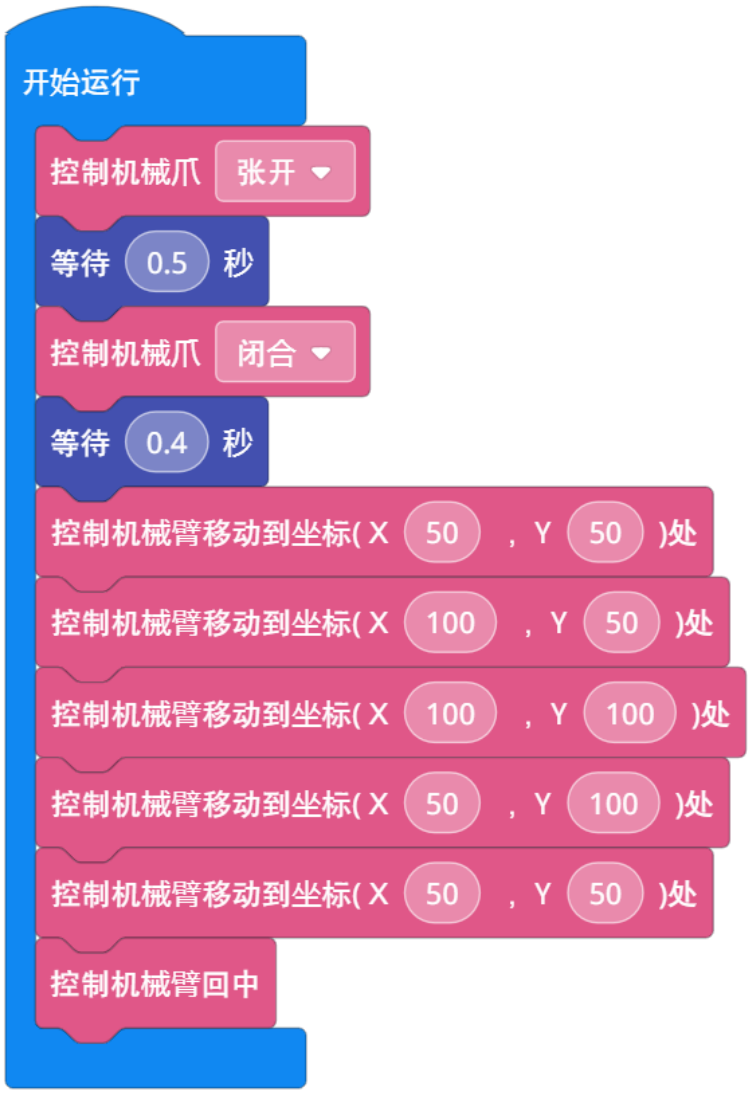
7、控制机械臂移动到坐标(X (50),Y (50))处

（1）含义：控制机械臂移动到指定位置处，沿机体前后方向为 X，沿机体垂直方向为 Y，单位为毫米。
（2）类型：执行类
（3）范例：取物
在机器人前方放置一个水弹瓶，控制机器臂准确到达，机械爪配合抓取，而后整机掉头。

注意：
1）机械臂接入时需要标定原点，请按照说明书上的指示进行。
2）灵活操控机械臂、机械爪和移动底盘，可以使机器人实现物品夹取、升降、转移、输送等功能。

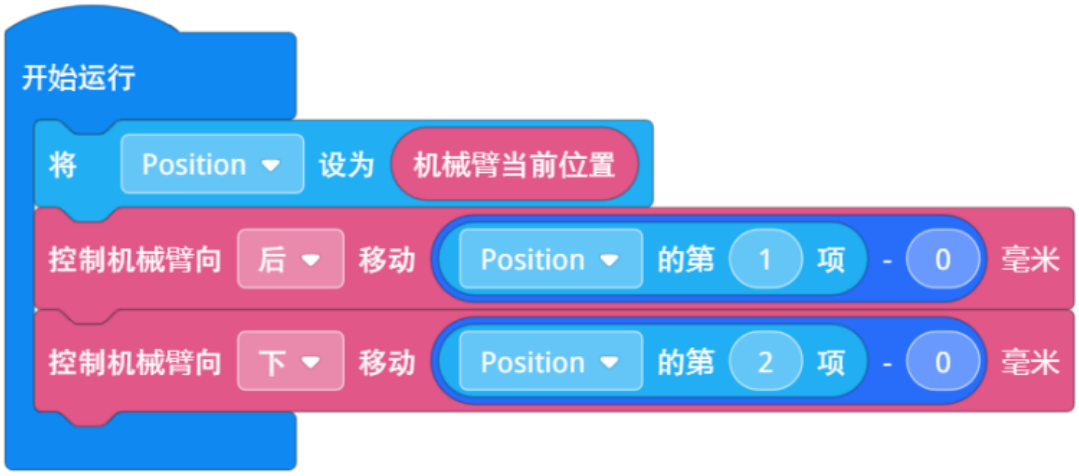
8、控制机械臂回中

（1）含义：控制机械臂回到初始位置，即机体坐标系（0,0）点。
（2）类型：执行类
（3）范例：写“口”字
控制机械爪夹紧毛笔，利用机械臂的移动完成“口”字撰写，而后回中。



9、机械臂当前位置

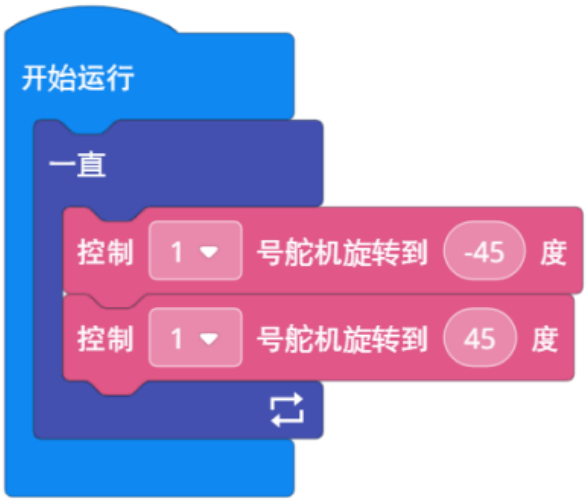
（1）含义：获取机械臂当前的位置信息，参数为（X（横坐标）,Y（纵坐标）），单位为毫米。
（2）类型：执行类
（3）范例：控制机械臂回中
通过计算机械臂当前横纵坐标和原点位置间的差值，控制机械臂回中。



可以通过 FPV 窗口观察：

10、控制（1）号舵机旋转到（90）度

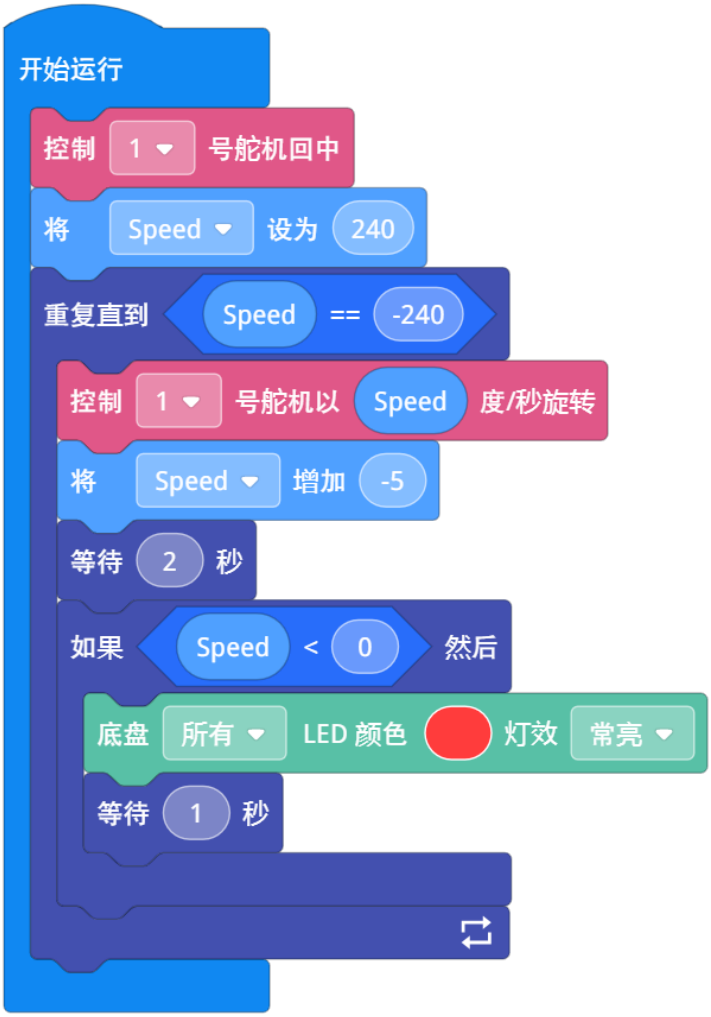
（1）含义：控制指定序号的舵机旋转到设定角度。正数为顺时针旋转，负数为逆时针旋转。
（2）类型：执行类
（3）范例：摆钟
利用舵机、舵盘和胶纸制作摆钟，控制舵机在 -45 度到 45 度间来回旋转。



注意：
1）舵机支持设定角度和调节速度两种控制，此模块适用于对舵机旋转角度控制要求高的情况。
2）当舵机被装配在机械臂上时，不支持使用“拓展机构”大类里的舵机相关模块对其进行控制。

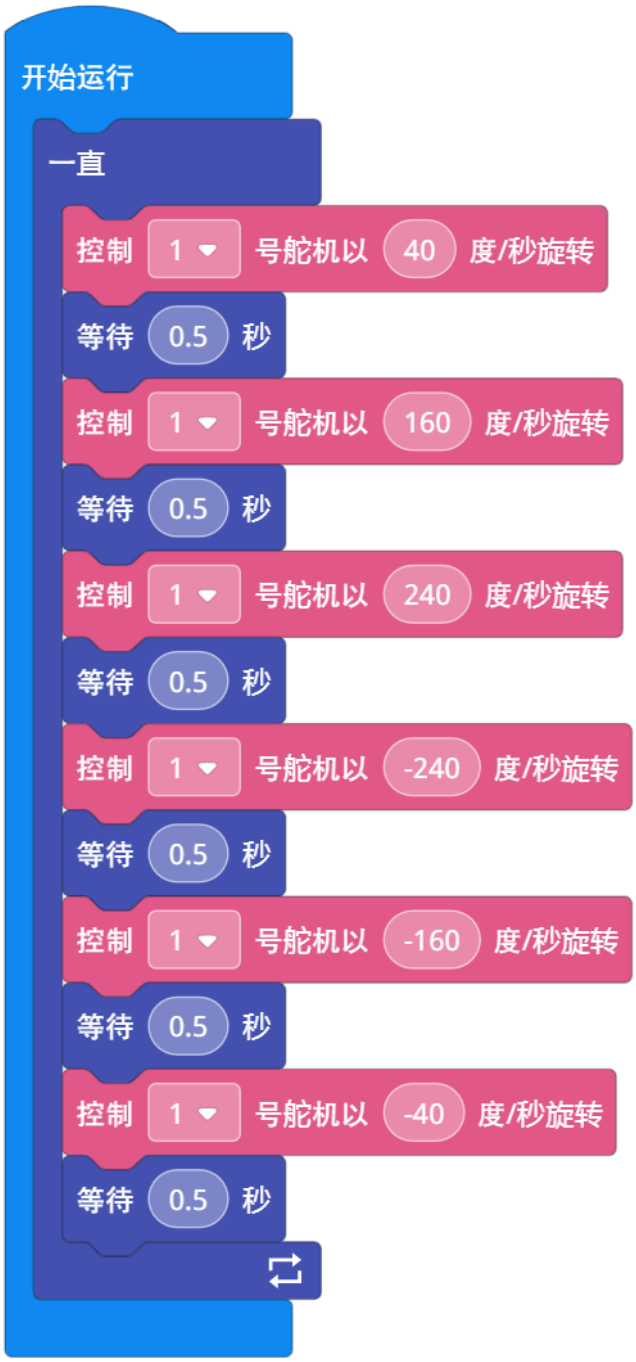
11、控制（1）号舵机回中

（1）含义：控制指定序号的舵机回到初始位置。
（2）类型：执行类
（3）范例：检测舵机性能
为了检测 1 号舵机性能，控制舵机从以最大速度顺时针旋转逐渐减速至停止，而后逆时针旋转并不断加速。在舵机由顺时针切换为逆时针旋转的时候，底盘红灯常亮提示。



12、控制（1）号舵机以（10）度/秒旋转

（1）含义：控制指定序号的舵机以设定速度旋转。正数为顺时针方向，负数为逆时针方向。
（2）类型：执行类
（3）范例：舵机转向变速
控制舵机顺时针旋转时不断加速，逆时针旋转时不断减速。



注意：
舵机支持设定角度和调节速度两种控制，此模块适用于对舵机进行速度控制的情况，舵机会持续旋转，无限位。

13、（1）号舵机的角度

（1）含义：获取指定序号舵机的角度。

（2）类型：信息类

（3）范例：舵机角度

在舵机不断旋转的过程中，获知舵机角度值的变化。

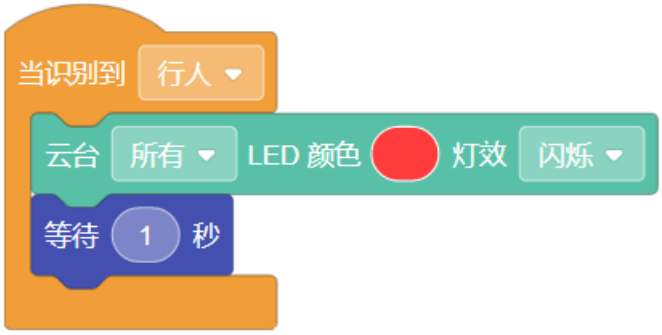
可以通过 FPV 窗口观察。

智能

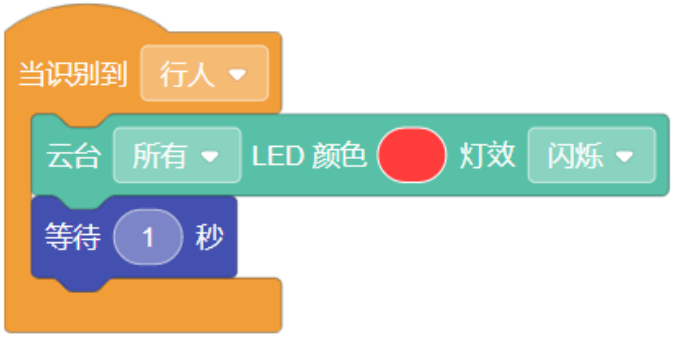
1、（开启）（视觉标签）识别

（1）含义：开启或关闭 RoboMaster EP 对视觉标签、姿势、行人或同类 EP 机器人的视觉识别功能
（2）类型：设置类
（3）范例：V 字模仿
RoboMaster EP 识别到“V 字”手势后，云台划出 V 形。

注意：
RoboMaster EP 的智能识别功能默认处于关闭状态。所以如果未先开启识别功能，机器人对相应的可识别信息就不会有反应，比如：



或



在这两个例程中，RoboMaster EP 不会识别到人，云台 LED 不会闪烁。

Python API：

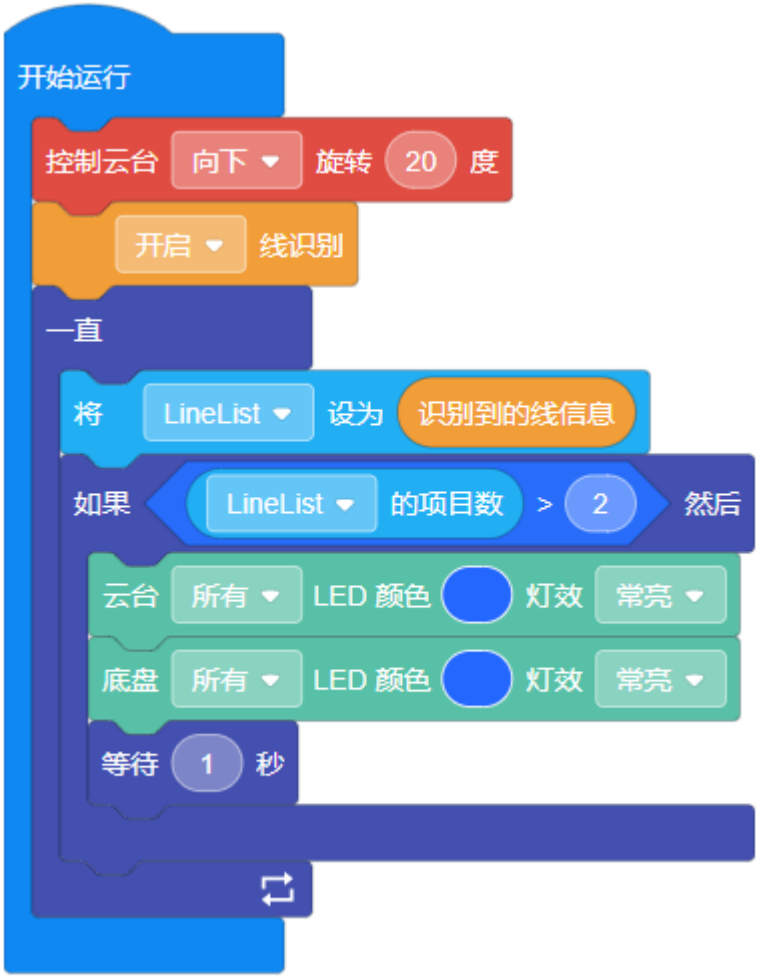
Function: vision_ctrl.enable_detection(function_enum)
vision_ctrl.disable_detection(function_enum)

Parameters:

- function_enum(enum):
  - rm_define.vision_detection_marker
  - rm_define.vision_detection_pose
  - rm_define.vision_detection_car
  - rm_define.vision_detection_people

2、（开启）线识别

（1）含义：开启或关闭线识别功能
（2）类型：设置类
（3）范例：蓝线识别示意
如果机器人识别到线，则底盘和云台所有 LED 常亮蓝光示意。

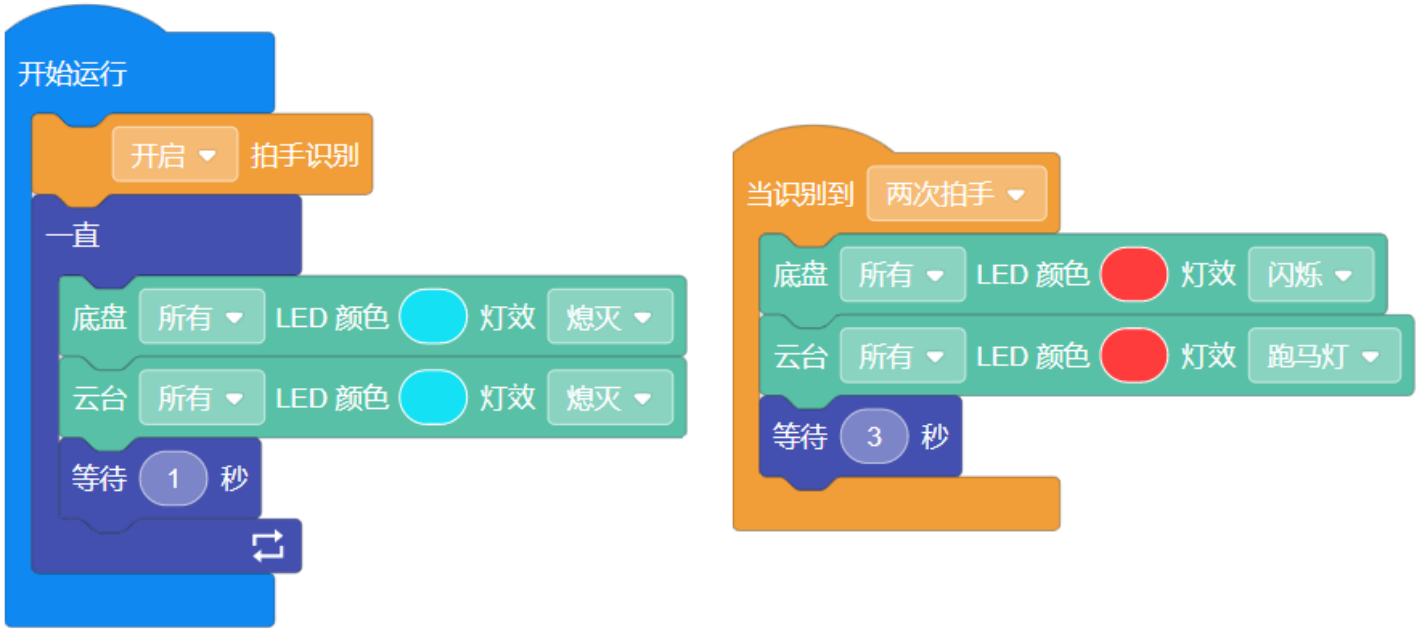


注意：
1）默认状态下，线识别功能处于关闭状态。所以如果未先“开启线识别”，RoboMaster EP 不会响应任何巡线指令。
2）机器人默认能够识别的线颜色是蓝色。

Python API：
Function: vision_ctrl.enable_detection(function_enum)
vision_ctrl.disable_detection(function_enum)
Parameters:
- function_enum(enum):
  - rm_define.vision_detection_line

3、（开启）拍手识别

（1）含义：开启或关闭掌声识别功能
（2）类型：设置类
（3）范例：拍手唤醒
底盘和云台 LED 保持熄灭状态。当识别到两次拍手后，底盘所有 LED 红光闪烁，云台 LED 呈红色跑马灯特效。



注意：
默认状态下掌声识别功能处于关闭状态。
所以如果未先开启拍手识别，则 RoboMaster EP 不会响应任何拍手指令。

Python API：
Function: media_ctrl.enable_sound_recognition(function_enum)
media_ctrl.disable_sound_recognition(function_enum)
Parameters:
- function_enum(enum):
  - rm_define.sound_detection_applause

4、设置视觉标签的可识别距离为（1）米内

（1）含义：设置视觉标签的有效识别距离，超出限制则不会识别
（2）类型：设置类
（3）范例：可识别距离
机器人识别到前进箭头后向前平移 1 米。

注意：

1）对此例程来说，如果把视觉标签放置在 1 米开外（比如说离机器人 1.5 米或 2 米远）的地方，机器人将无法识别。

2）此识别距离只对官方提供的标准尺寸的视觉标签而言，如果是自行打印的非标准尺寸视觉标签，有效识别距离会有差异。

Python API：

Function: vision_ctrl.set_marker_detection_distance(distance)

Parameters:

- distance(float): [0.5, 3]

5、设置视觉标签识别颜色为（红）

（1）含义：设置机器人能够识别的视觉标签颜色。

（2）类型：设置类

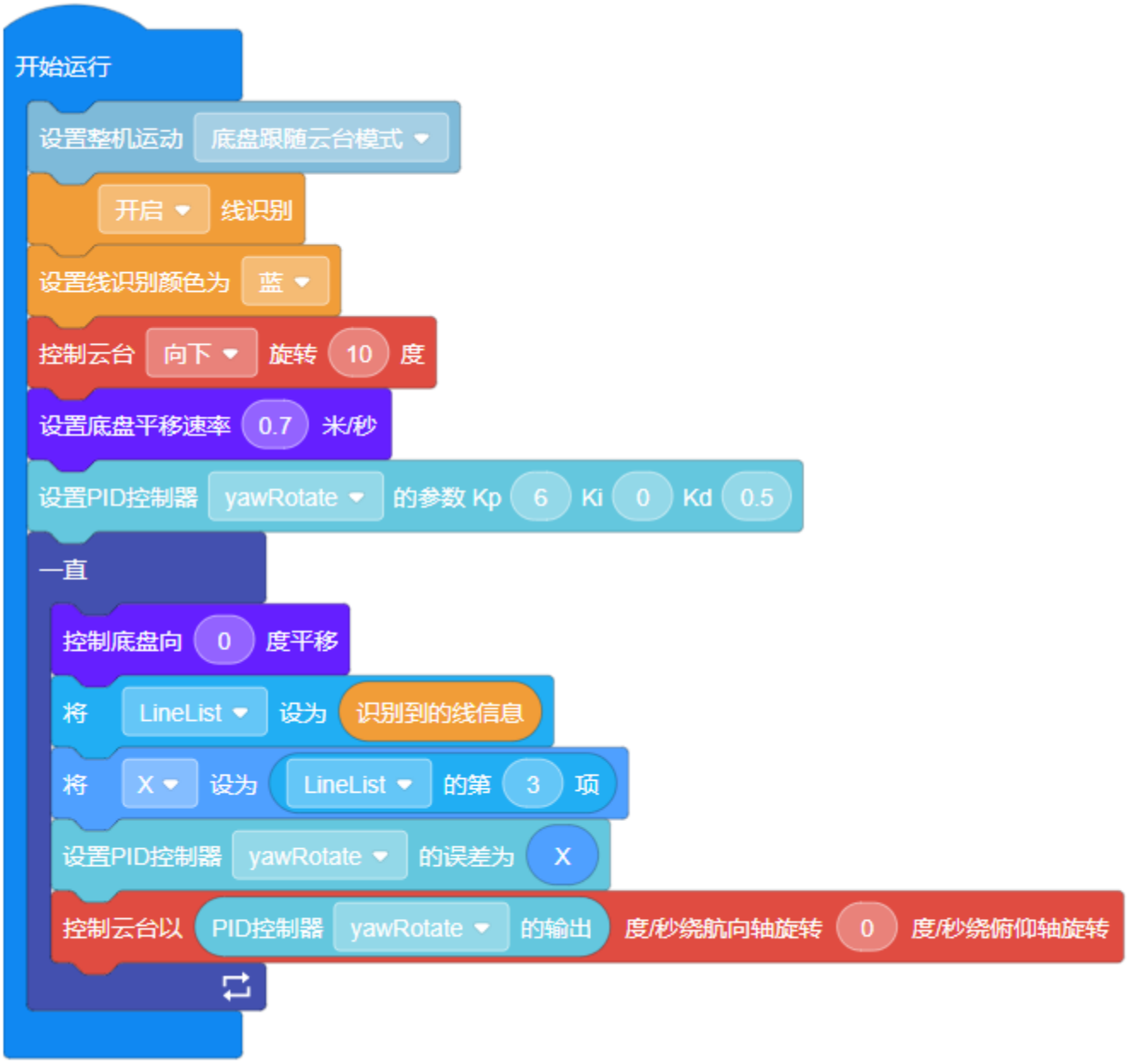
（3）范例：不同颜色的视觉标签识别切换

待机器人识别到红色标签数字 1 后，底盘红色 LED 光常亮；识别到绿色标签数字 1 后，底盘绿色 LED 光常亮。

注意：
机器人默认能够识别的视觉标签颜色是红色。

6、设置线识别颜色为（蓝）

（1）含义：设置机器人能够识别的线颜色
（2）类型：设置类
（3）范例：巡蓝线
在地面上贴好蓝色胶带，EP 机器人将沿蓝线行驶。



注意：
机器人默认能够识别的线颜色是蓝色。

Python API：
Function: vision_ctrl.line_follow_color_set(color_enum)
Parameters:

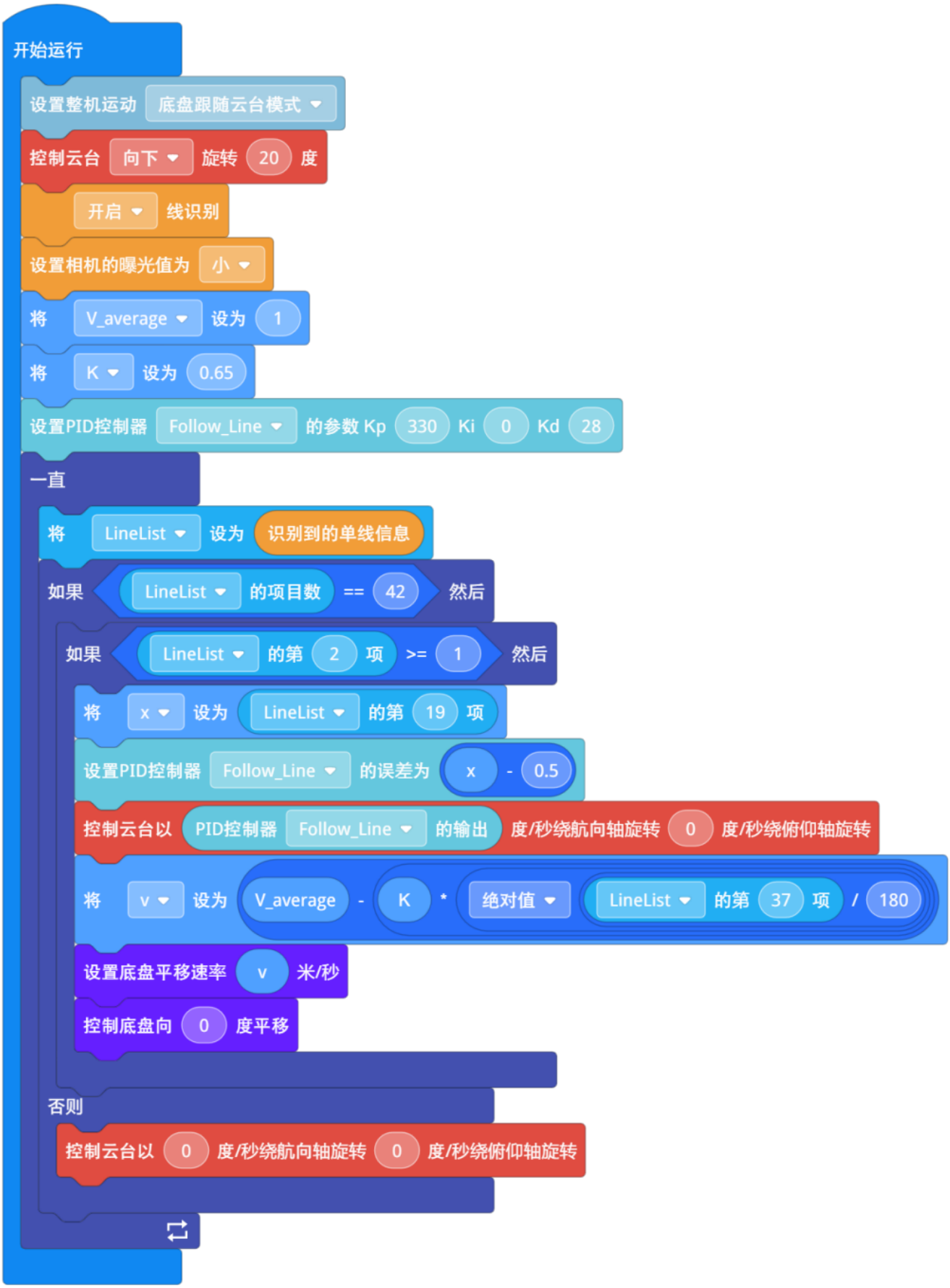
- color_enum(enum):
  - rm_define.line_follow_color_blue
  - rm_define.line_follow_color_red
  - rm_define.line_follow_color_green

7、设置相机的曝光值为（大）

（1）含义：在巡线时减小曝光值，可以缓解快速转向造成的视野模糊，让识别更稳定。

（2）类型：设置类

（3）范例：EP 巡线



Python API：

Function: media_ctrl.exposure_value_update(exposure_value_enum)

Parameters:

- exposure_value_enum(enum):
  - rm_define.exposure_value_large
  - rm_define.exposure_value_medium
  - rm_define.exposure_value_small

8、识别到（红心）并瞄准

（1）含义：机器人识别并瞄准对应视觉标签的中心位置

（2）类型：执行类

（3）范例：瞄准红心

注意：

1、请先“开启视觉标签识别”功能，否则不会识别。

2、使用此模块时，当视野中有红心出现就会瞄准，5 秒内没有识别到红心的话会超时退出，运行后面的程序。

Python API：

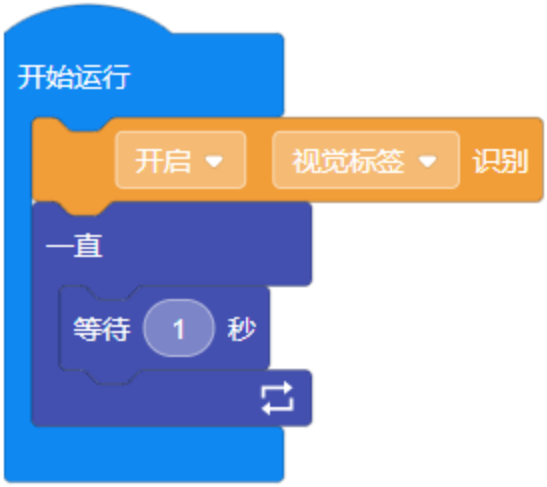
Function: vision_ctrl.detect_marker_and_aim(marker_enum)

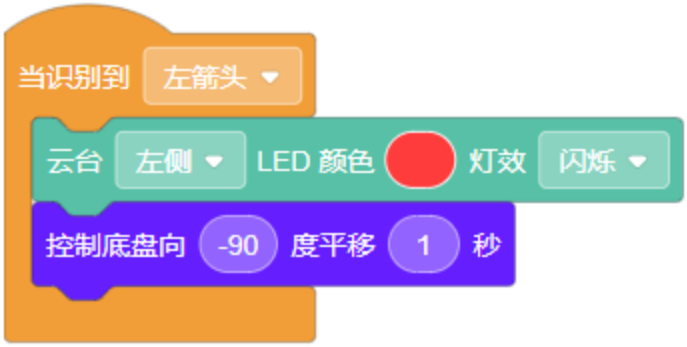
Parameters:

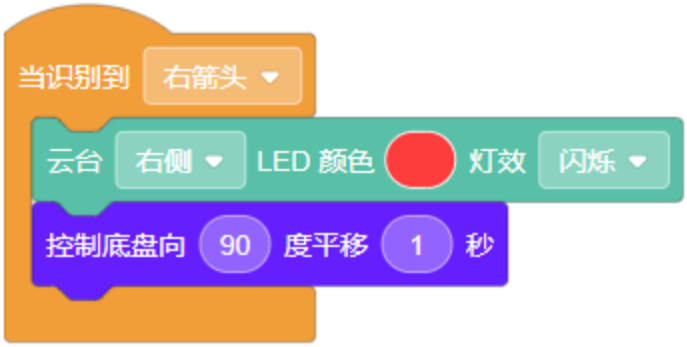
- rm_define.marker_trans_red_heart
- rm_define.marker_trans_target
- rm_define.marker_trans_dice
- rm_define.marker_number_[zero,..., nine]
- rm_define.marker_letter_[A,..., Z]

9、当识别到（行人）

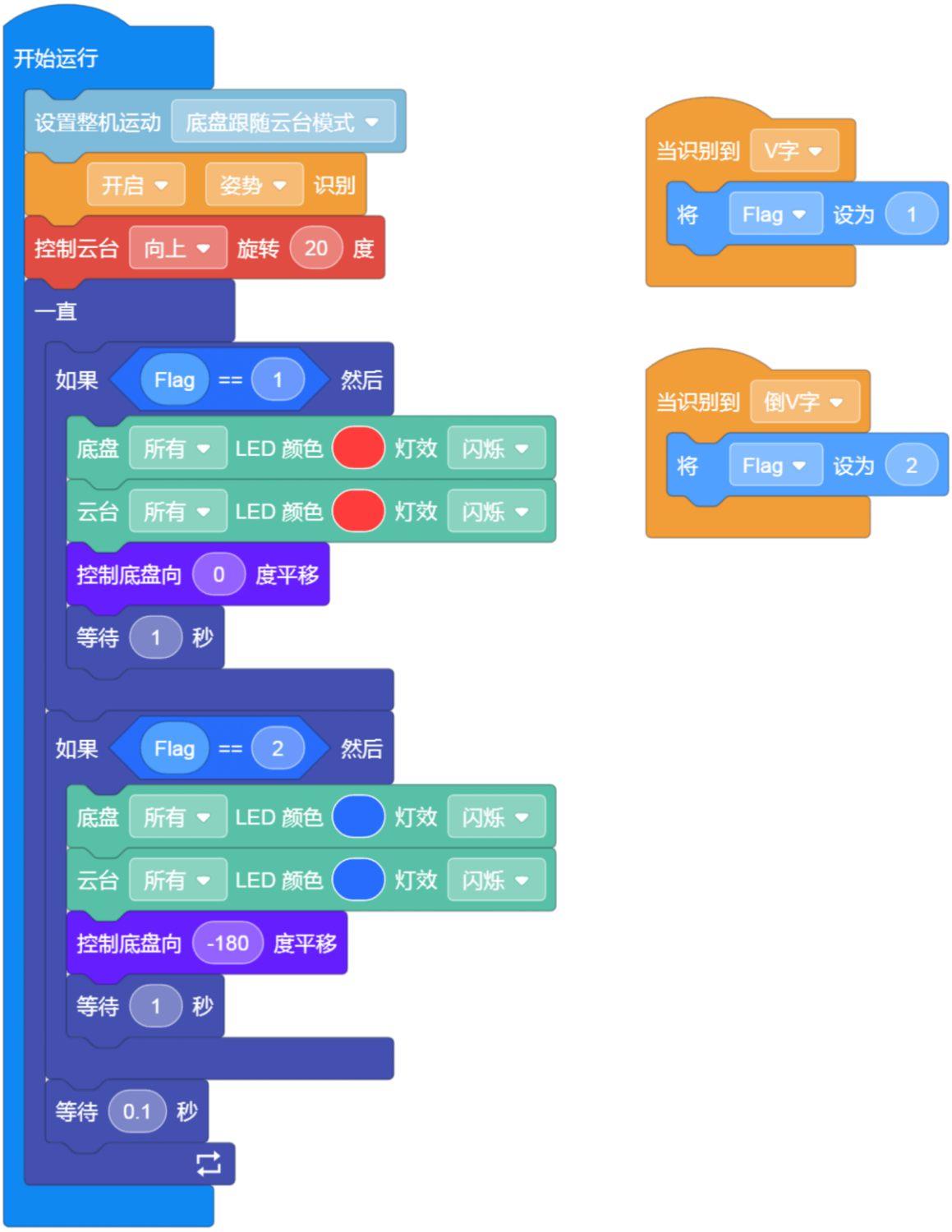
（1）含义：当识别到物体、视觉标签、姿势等对应信息时运行本模块内的程序
（2）类型：事件类
（3）范例：分叉路指示、姿势控制
①分叉路指示
当识别到左箭头，云台左侧 LED 红光闪烁，机器人向左平移 1 秒；当识别到右箭头，云台右侧 LED 红光闪烁，底盘向右平移 1 秒。







②姿势控制
行人站在机器人前 1 米处，用姿势控制机器人运动——行人作出 V 字姿势时机器人前进，倒 V 字姿势时机器人后退。



注意：

1）事件触发模块优先级高，也就是说无论主函数运行到哪一步，只要满足事件触发条件，就会立刻跳出主函数，开始运行事件类模块内的程序。

2）需识别行人时，请控制云台适度向上旋转，让行人站在机器人前 1 米处，不要蹲着，确保行人完整出现在机器人视野中。

3）V 字、倒 V 字的姿势指令需要用手臂完成，而非手指。

Python API：

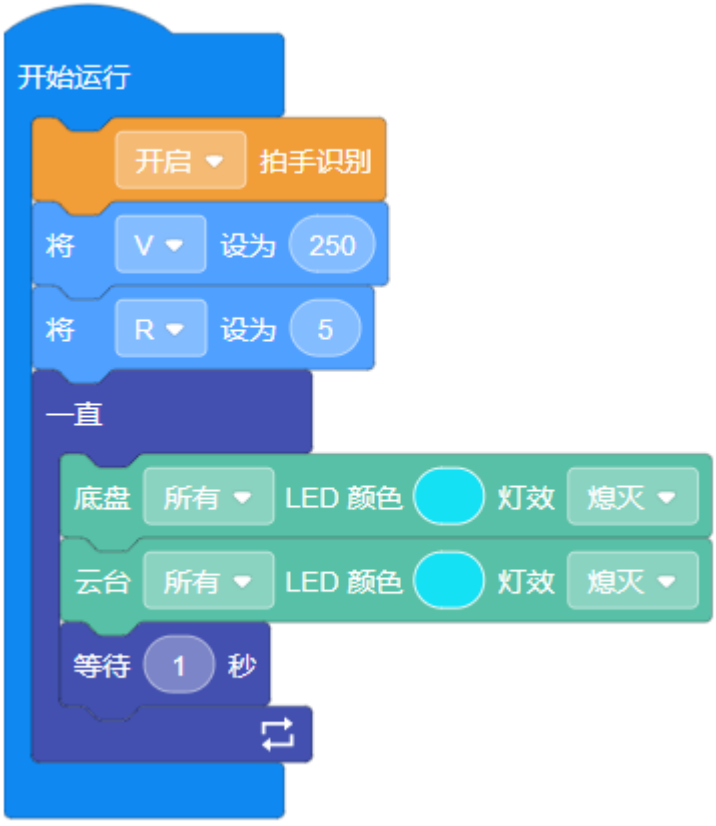
Function: def vision_recognized_people(msg)
def vision_recognized_car(msg)
def vision_recognized_pose_all(msg)
def vision_recognized_pose_victory(msg)

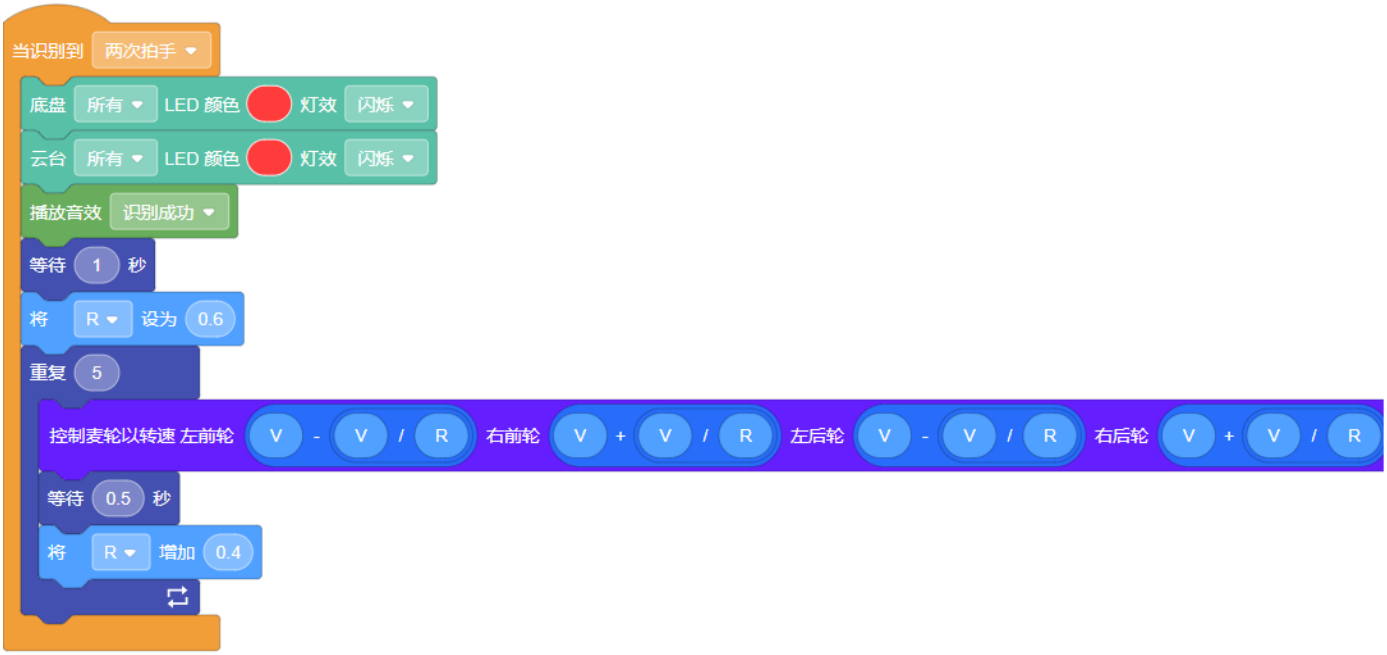
def vision_recognized_pose_give_in(msg)
def vision_recognized_pose_capture(msg)
def vision_recognized_marker_trans_all(msg)
def vision_recognized_marker_trans_left(msg)
def vision_recognized_marker_trans_right(msg)
def vision_recognized_marker_trans_stop(msg)
def vision_recognized_marker_trans_forward(msg)
def vision_recognized_marker_trans_red_heart(msg)
def vision_recognized_marker_trans_target(msg)
def vision_recognized_marker_trans_dice(msg)
def vision_recognized_marker_number_all(msg)
def vision_recognized_marker_number_[one, ..., nine](msg)
def vision_recognized_marker_letter_all(msg)
def vision_recognized_marker_letter_[A, ..., Z](msg)

Type: Event callback

10、当识别到（两次拍手）

（1）含义：当识别到对应的拍手指令时将运行本模块内的程序
（2）类型：事件类
（3）范例：螺旋线
底盘和云台所有 LED 熄灭，当识别到两次拍手后，所有 LED 红光闪烁，机器人做螺旋线运动。





Python API:

Function: def sound_recognized_applause_twice(msg)

def sound_recognized_applause_thrice(msg)

Type: Event callback

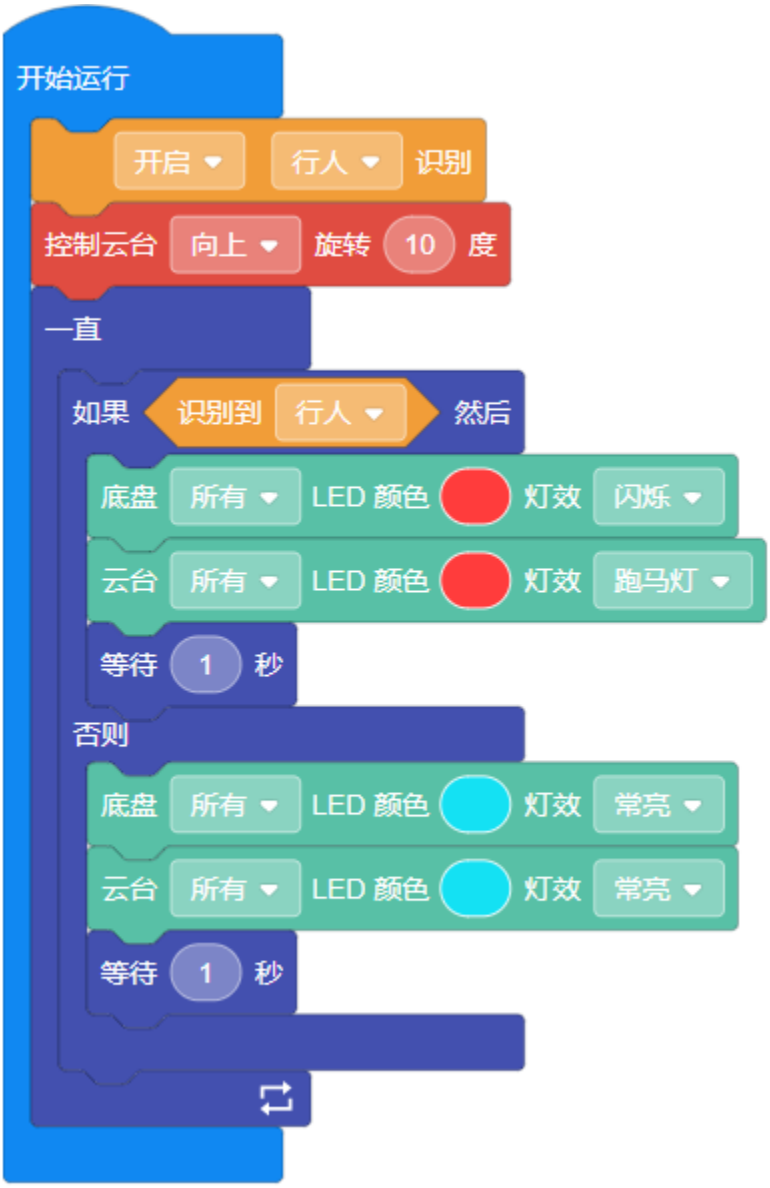
11、识别到（行人）

（1）含义：识别到物体、视觉标签、姿势等对应信息时返回“真”，否则将返回“假”

（2）返回值：布尔型

（3）范例：行人识别

如果机器人视野中识别到行人，底盘所有 LED 红光闪烁，云台所有 LED 呈红色跑马灯特效；否则，底盘和云台所有 LED 以默认色常亮。



注意：
多和条件类模块连用。

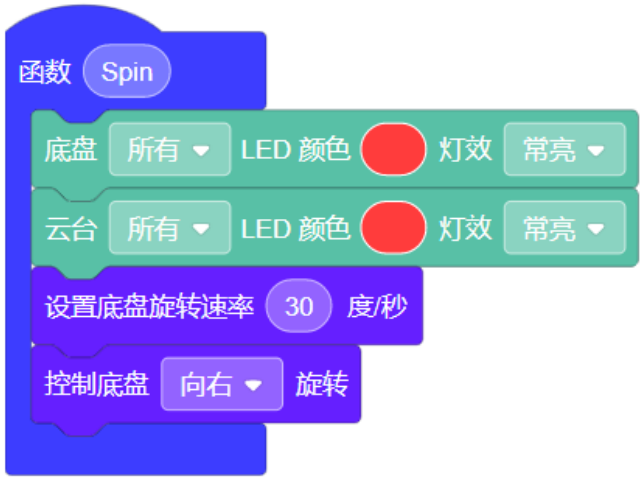
Python API：
Function: vision_ctrl.check_condition(condition_enum)
Parameters:

- condition_enum(enum):
  - rm_define.cond_recognized_people
  - rm_define.cond_recognized_car
  - rm_define.cond_recognized_marker_trans_all
  - rm_define.cond_recognized_marker_trans_left
  - rm_define.cond_recognized_marker_trans_right
  - rm_define.cond_recognized_marker_trans_forward
  - rm_define.cond_recognized_marker_trans_stop
  - rm_define.cond_recognized_marker_trans_red_heart
  - rm_define.cond_recognized_marker_trans_target
  - rm_define.cond_recognized_marker_trans_dice
  - rm_define.cond_recognized_marker_number_all
  - rm_define.cond_recognized_marker_number_[zero,..., nine]

- rm_define.cond_recognized_marker_letter_all
- rm_define.cond_recognized_marker_letter_[A,..., Z]
- rm_define.cond_recognized_pose_all
- rm_define.cond_recognized_pose_victory
- rm_define.cond_recognized_give_in
- rm_define.cond_recognized_capture

12、识别到（两次拍手）

（1）含义：识别到对应的拍手指令时返回“真”，否则返回“假”
（2）返回值：布尔型
（3）范例：拍手叫停
机器人向右旋转，如果识别到三次拍手则停止所有运动。



Python API：
Function: vision_ctrl.check_condition(condition_enum)
Parameters:

- condition_enum(enum):
  - rm_define.cond_sound_recognized_applause_twice
  - rm_define.cond_sound_recognized_applause_thrice

13、等待识别到（行人）

（1）含义：待机器人识别到物体、视觉标签、姿势等对应信息时将继续执行，否则将持续等待
（2）类型：执行类、阻塞型
（3）范例：打靶

注意：
若未识别到靶标签，程序会在此模块上停留等待，此时模块保持高亮：

Python API：
Function: vision_ctrl.cond_wait(condition_enum)
Parameters:

- condition_enum(enum):
  - rm_define.cond_recognized_people

- rm_define.cond_recognized_car
- rm_define.cond_recognized_marker_trans_all
- rm_define.cond_recognized_marker_trans_left
- rm_define.cond_recognized_marker_trans_right
- rm_define.cond_recognized_marker_trans_forward
- rm_define.cond_recognized_marker_trans_stop
- rm_define.cond_recognized_marker_trans_red_heart
- rm_define.cond_recognized_marker_trans_target
- rm_define.cond_recognized_marker_trans_dice
- rm_define.cond_recognized_marker_number_all
- rm_define.cond_recognized_marker_number_[zero,..., nine]
- rm_define.cond_recognized_marker_letter_all
- rm_define.cond_recognized_marker_letter_[A,..., Z]
- rm_define.cond_recognized_pose_all
- rm_define.cond_recognized_pose_victory
- rm_define.cond_recognized_give_in
- rm_define.cond_recognized_capture

14、等待识别到（两次拍手）

（1）含义：待机器人识别到对应的拍手指令时将继续执行，否则将持续等待
（2）类型：执行类、阻塞型
（3）范例：拍手录制
两次拍手后开始视频录制，三次拍手后结束视频录制。

Python API：
Function: vision_ctrl.cond_wait(condition_enum)

Parameters:

- condition_enum(enum):
  - rm_define.cond_sound_recognized_applause_twice
  - rm_define.cond_sound_recognized_applause_thrice

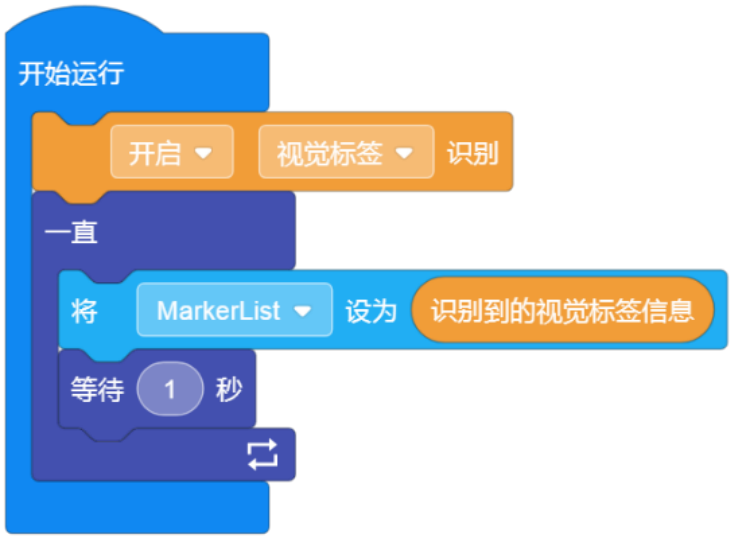
15、识别到的视觉标签信息

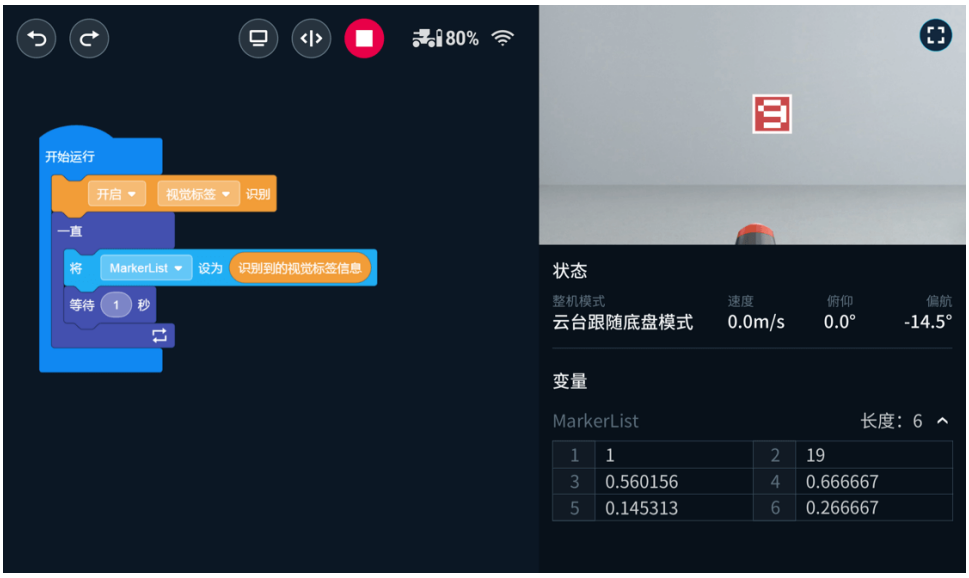
（1）含义：获取识别到的视觉标签信息，参数为 N , ID , X , Y , W, H

（2）类型：信息类（列表型数据）

（3）范例：视觉标签信息

在机器人视野中放置 1 个视觉标签，在 FPV 窗口打印获取到的视觉标签信息。移动视觉标签，观察这 6 项数据的变化。





注意：

1）视觉标签信息的格式为：

第 1 项为识别到的视觉标签数量 N，后续项以 5 个数据为一组，分别是第一个视觉标签的 ID，标签中心点在机器人视野中位置的横坐标 X、纵坐标 Y、宽度 W 和高度 H；第二个视觉标签的 ID、标签中心点在机器人视野中位置的横坐标、纵坐标、宽度和高度；第三个…

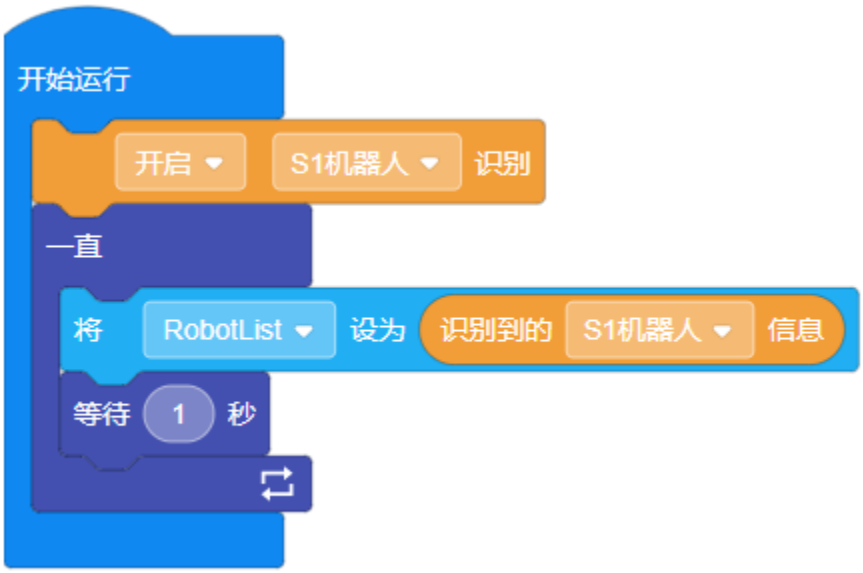
2）返回的 ID 值释义：
ID 值为 1 代表识别到的是：停止
ID 值为 2 代表识别到的是：骰子
ID 值为 3 代表识别到的是：靶
ID 值为 4 代表识别到的是：左箭头
ID 值为 5 代表识别到的是：右箭头
ID 值为 6 代表识别到的是：前进箭头
ID 值为 8 代表识别到的是：红心
ID 值为 10-19 代表识别到的是：数字 0-9
ID 值为 20-45 代表识别到的是：字母 A-Z

Python API：
Function: vision_ctrl.get_marker_detection_info()
Return value:

- detection_info(list)

16、识别到的（行人）信息

（1）含义：获取识别到的行人或 S1 机器人信息，参数为 N，X，Y，W，H
（2）类型：信息类（列表型数据）
（3）范例：S1 机器人信息
在机器人视野中出现一台同类的 S1 机器人，在 FPV 窗口打印获取到的机器人信息。移动视野中的机器人，观察这 5 项数据的变化。



注意：

物体信息的格式为：

第 1 项为识别到的物体数量 N，后续项以 4 个数据为一组，分别是第一个物体中心点在机器人视野中位置的横坐标 X、纵坐标 Y、宽度 W 和高度 H；第二个物体中心在机器人视野中位置的横坐标、纵坐标、宽度和高度；第三个……

Python API：

Function: vision_ctrl.get_people_detection_info()

vision_ctrl.get_car_detection_info()

Return value:

- detection_info(list)

17、识别到的姿势信息

（1）含义：获取识别到的姿势信息，参数为 N，ID，X，Y，W，H

（2）类型：信息类（列表型数据）

（3）范例：姿势信息

可以在 FPV 窗口观察姿势信息的变化。

注意：

1）姿势信息的格式为：

第 1 项为识别到的姿势数量 N，后续项以 5 个数据为一组，分别是第一个姿势的 ID 值，姿势中心点在机器人视野中的横坐标 X、纵坐标 Y，宽度 W 和高度 H；第二个姿势返回的 ID 值，中心点在机器人视野中的横坐标、纵坐标，宽度和高度；第三个……

2）ID 值释义：

ID 值为 4 对应姿势：正 V

ID 值为 5 对应姿势：倒 V 字

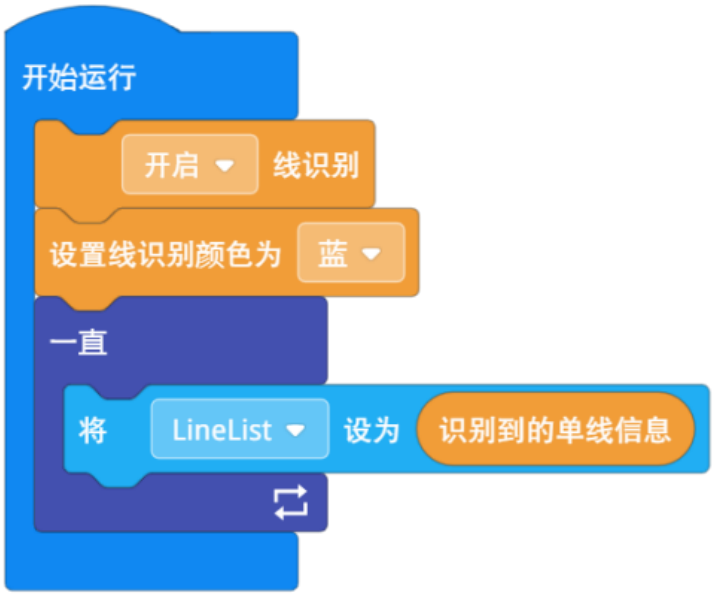
ID 值为 6 对应姿势：拍照手势

Python API：
Function: vision_ctrl.get_pose_detection_info()
Return value:

- detection_info(list)

18、识别到的单线信息

（1）含义：获取识别到的单条线信息，参数为 N，Info，X，Y， θ ，C
（2）类型：信息类（列表型数据）
（3）范例：单线信息
机器人视野中出现蓝线后，获取返回的线信息。



可以通过 FPV 窗口实时观察线路数据变动。

状态

整机模式	速度	俯仰	偏航
云台跟随底盘模式	2.4m/s	-20.0°	0.0°

变量

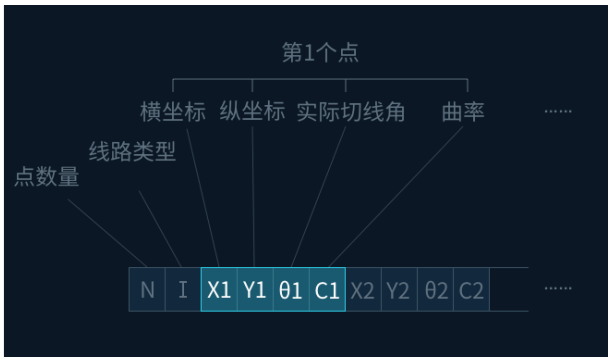
LineList 长度：42

1	10	2	1
3	0.503125	4	0.794444
5	-3.598248	6	0
7	0.503125	8	0.766667
9	-4.580367	10	-0.032737
11	0.5	12	0.738889
13	-0.700007	14	-0.700007
15	0.5	16	0.711111
17	-0.700007	18	0.002858
19	0.5	20	0.683333
21	-5.397032	22	-0.156504
23	0.496875	24	0.655556

注意：

1）机器人识别到的单线信息格式为：

第一项 N 为识别到的线上点的数量，第二项 Info 为线路类型，后续项以四个数据为一组：分别是线上由近及远等距提取的十个点的（横坐标 X，纵坐标 Y，实际切线角 θ， 曲率 C），一共 42 个数据。

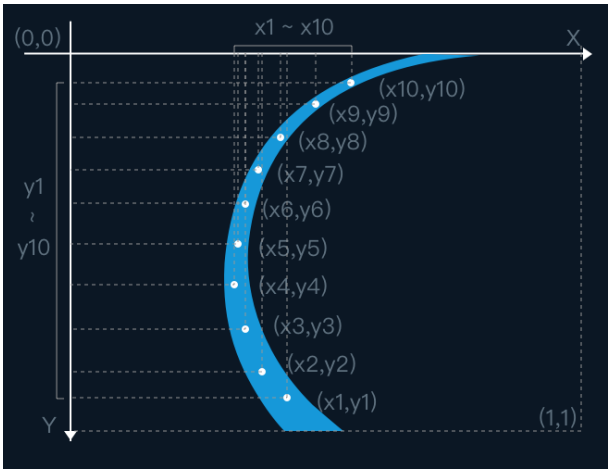


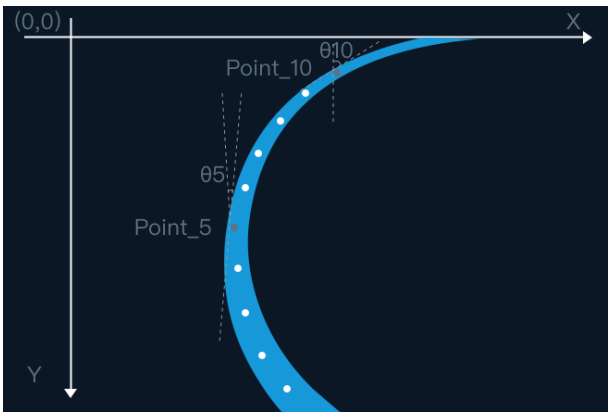
第一项数据 N：点数量。固定值 10 或者 0，“10”表示识别到了线，“0”表示没有识别到线。

第二项数据 I：线路类型。四种情况——“0”表示没有识别到线，“1”代表视野内一条线，“2”表示 Y 字路口，“3”表示十字路口。

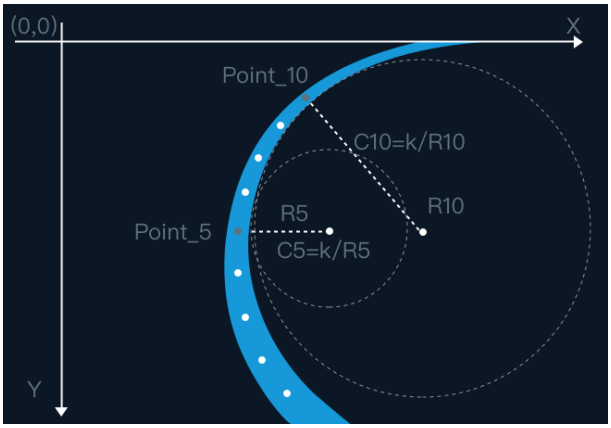
第三、第四项数据（X1，Y1）表示的是线上第一个点的坐标信息。

第一个点位于视野最下端，也就是最接近机器人的点。





第五项数据 θ1：第一个点的切线角。当切线角数值为 0 时，意味着此点切线与竖直方向的夹角为 90 度；切线角为 90 时，证明此点所在的线非常弯曲。

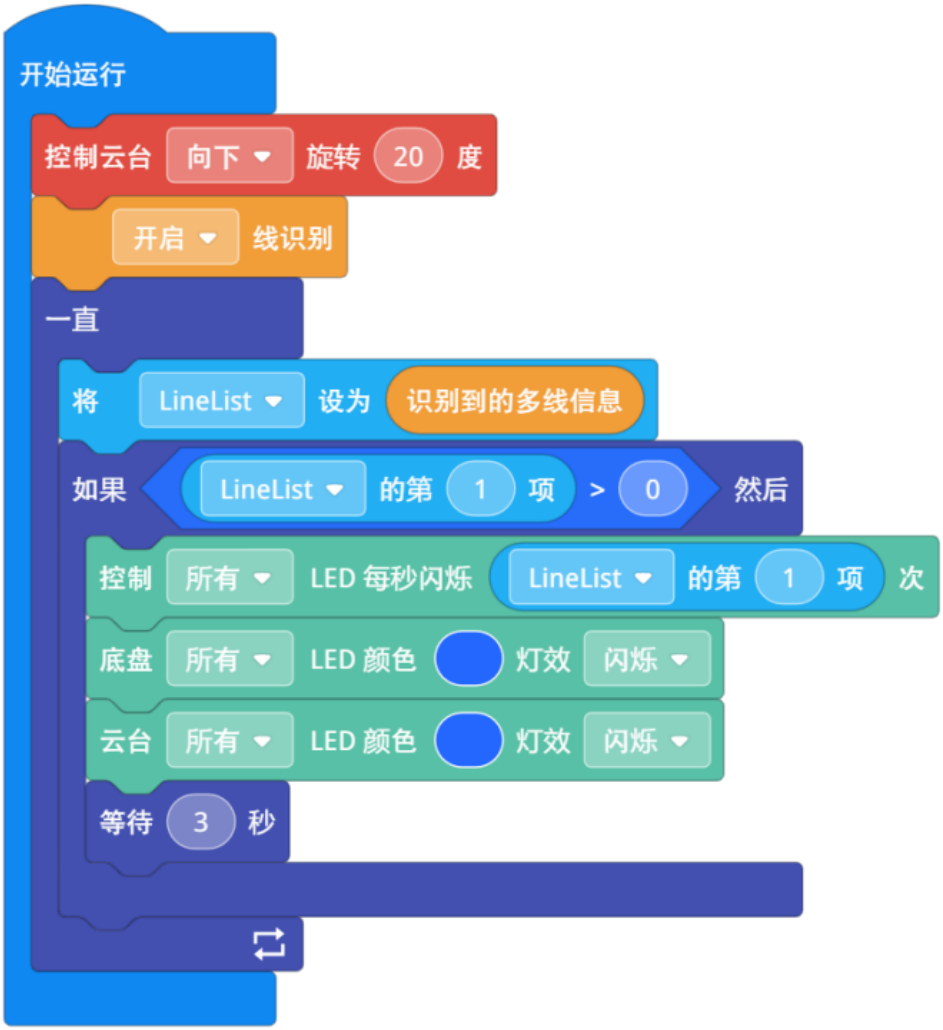


第六项数据 C1：第一个点的曲率。C=k/R，取值范围为 0 到 10，0 为纯直线，从 1 到 10 线弧度不断增大。当外接圆越大，证明此点附近的线弧度越大。

2）如果机器人没有识别到线，返回的线信息数据长度为 2，第一项和第二项都为 0。

19、识别到的多线信息

（1）含义：获取识别到的多条线信息，参数为 n，X，Y， θ，C
（2）类型：信息类（列表型数据）
（3）范例：多线闪烁示意
机器人识别到几条线，底盘和云台所有 LED 就蓝光闪烁几次。



注意：

机器人识别到的多线信息格式为：

n（线的条数），顺时针第一条线上由近及远等距提取的十个点的（横坐标 X，纵坐标 Y，实际切线角 θ，曲率 C），第二条线上十个点的(X,Y,θ,C) … 第 n 条线上十个点的(X,Y,θ,C)，一共有 40n+1 个数据。

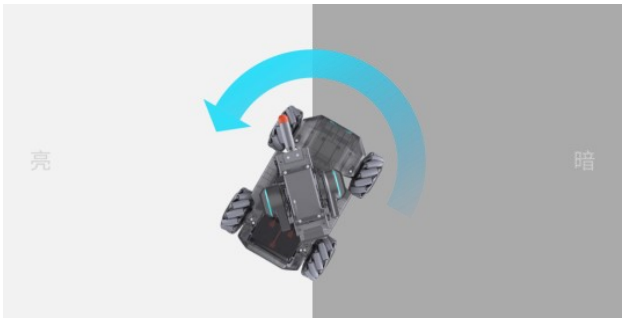
机器人“识别到的多线信息”比“识别到的单线信息”模块主要多了一个线的条数的反馈。

20、当前场景亮度

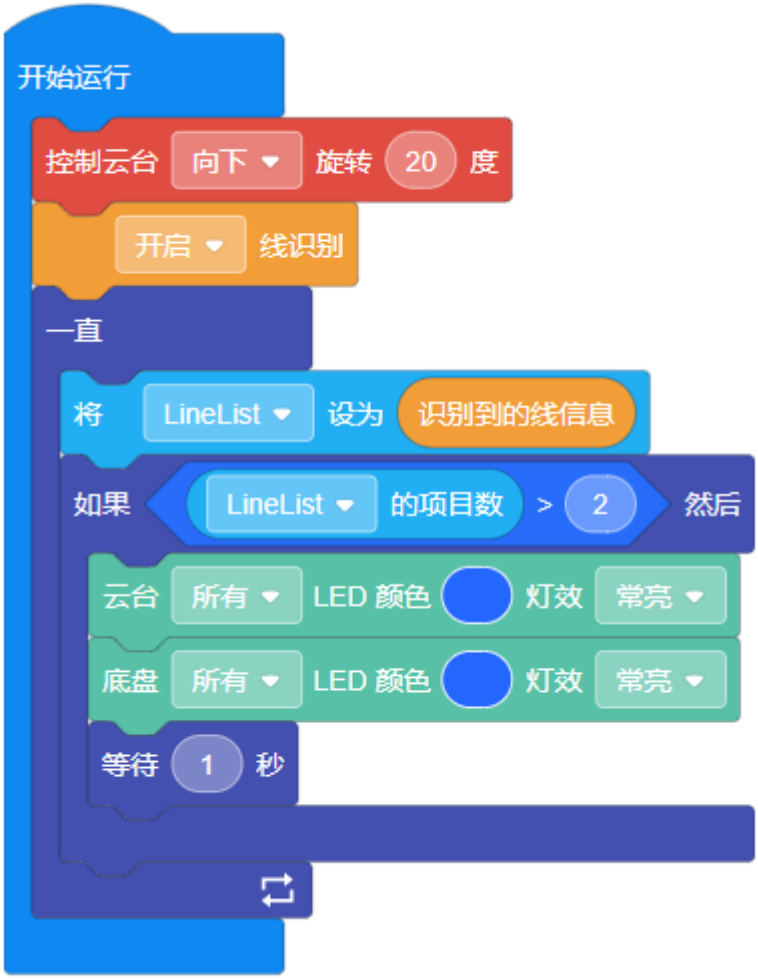
（1）含义：获取当前场景的亮度信息，返回数值 0-10。数值越大，代表当前场景越亮。

（2）类型：信息类（变量型数据）

（3）范例：趋光



如果视野前方较暗，EP 打开弹道灯照明，并转向亮处



Python API：

Function: vision_ctrl.get_env_brightness()

Return value:

- brightness_value(int)

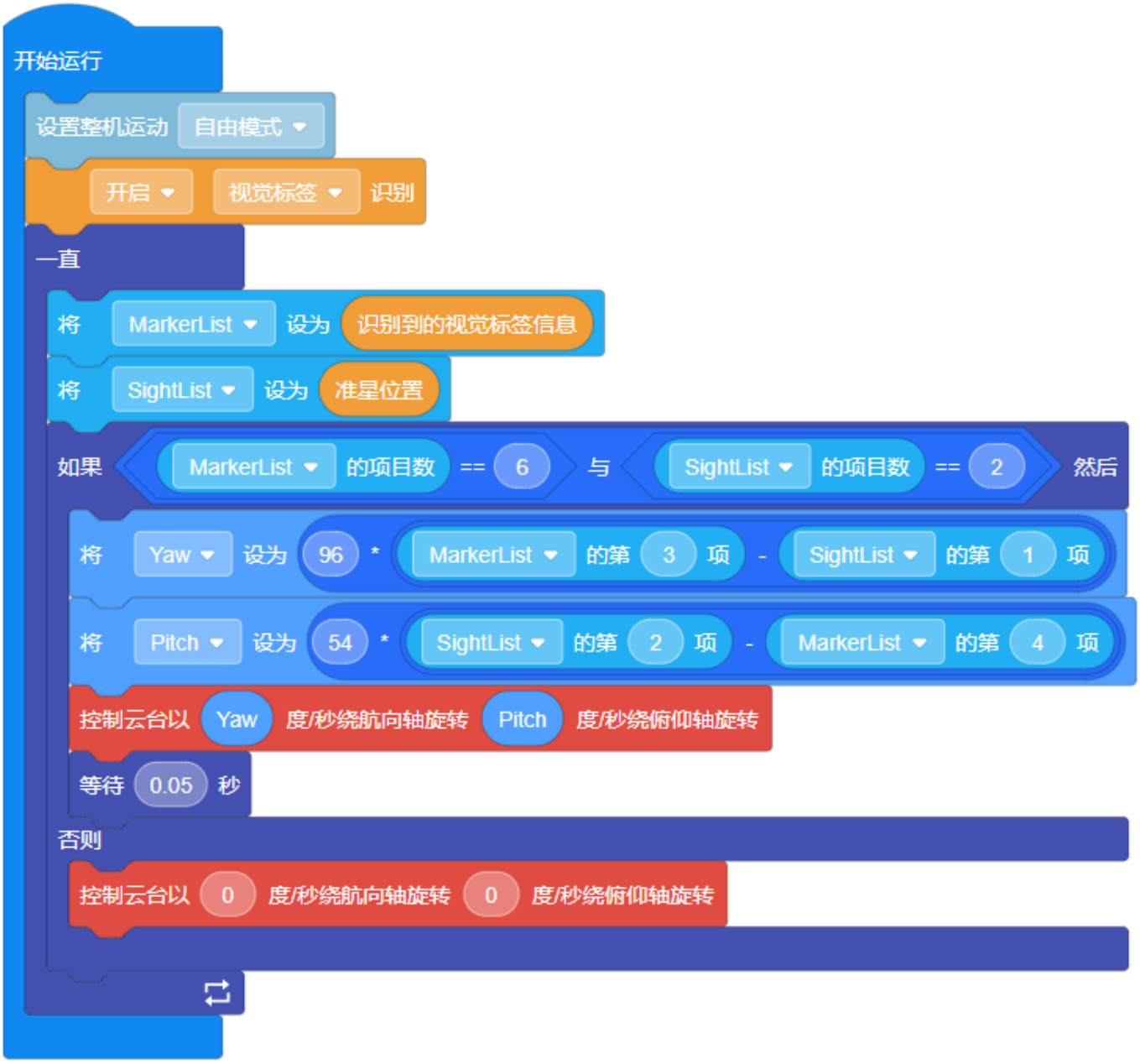
21、准星位置

（1）含义：获取准星位置信息，参数为 X，Y

（2）类型：信息类（列表型数据）

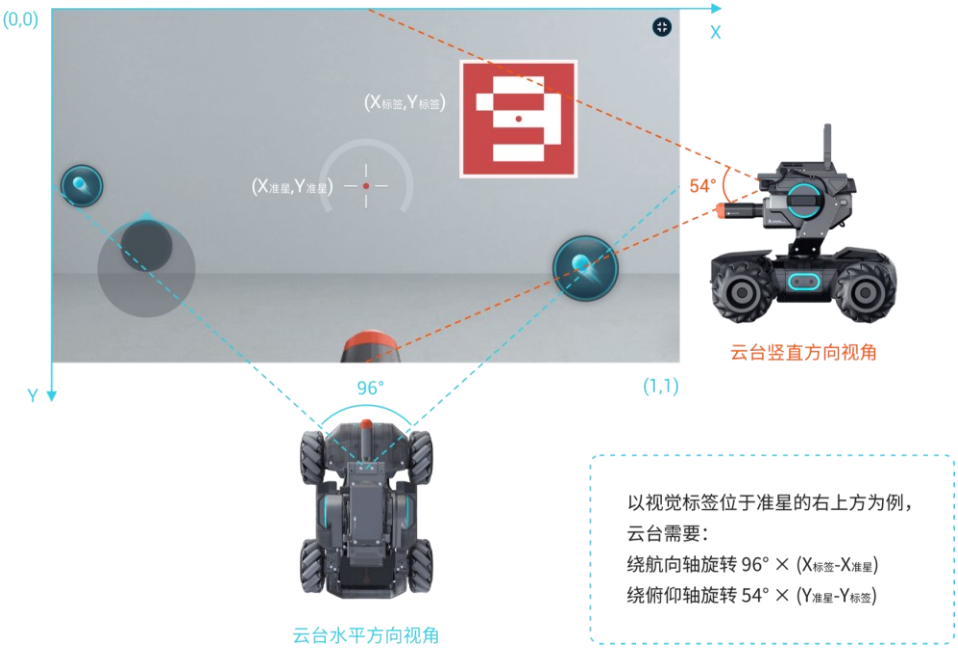
（3）范例：视觉标签跟随瞄准

将准星和视觉标签中心点在云台视野中位置的差值转换为云台旋转角度值，控制云台跟随视觉标签移动。



注意：

准星数据格式为：横坐标 X、纵坐标 Y，共 2 项。



Python API：

Function: media_ctrl.get_sight_bead_position()

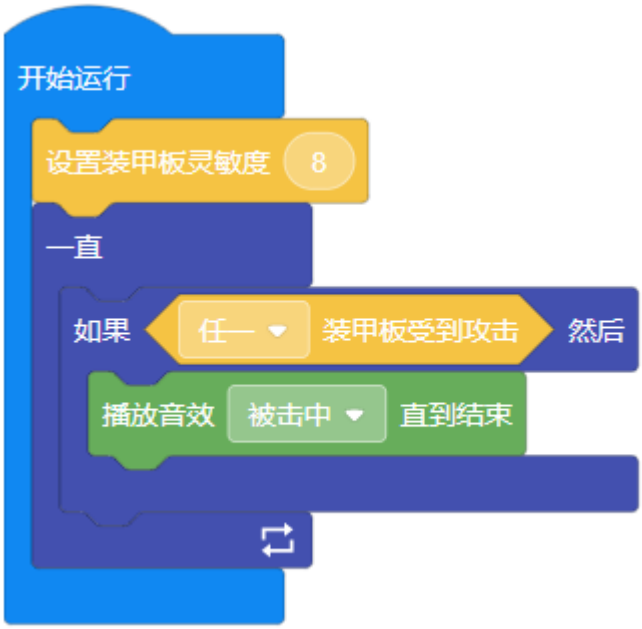
Return value:

- sight_bead_position(list)

装甲板

1、设置装甲板灵敏度（5）

（1）含义：数值越大，装甲板感应灵敏度越高。硬物敲击时建议灵敏度设为 6，指关节叩击时设为 8。
（2）类型：设置类
（3）范例：叩击响应
用手的指关节叩击机器人任一装甲板，播放“被击中”音效。



注意：
灵敏度设置只在实验室环境中生效，对战中装甲板的灵敏度都会恢复为默认值。

Python API：
Function: armor_ctrl.set_hit_sensitivity(value)
Parameters:
- value(int): [0, 10]

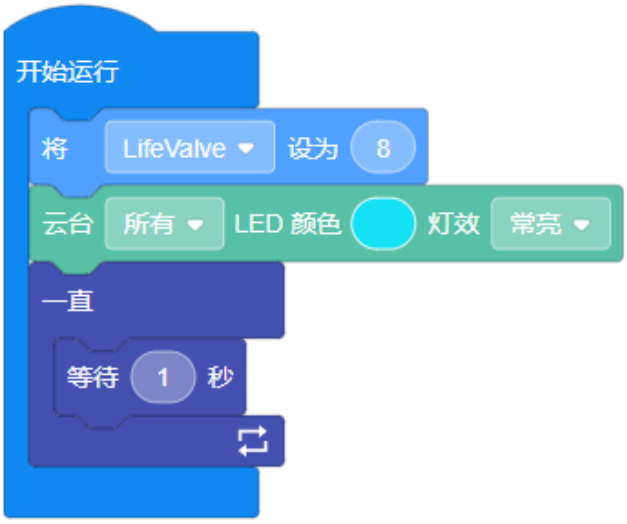
2、当（任一）装甲板受到攻击

（1）含义：当检测到指定位置的装甲板受到攻击时，运行本模块内的程序
（2）类型：事件类

（3）范例：血条灯

云台 LED 可以示意机器人当前的生命值，满血为 8。

当任一装甲板受到攻击，机器人掉 1 血；当生命值为 0 时，所有 LED 熄灭，所有运动停止。



注意:

事件触发模块优先级高，也就是说无论主函数运行到哪一步，只要满足触发条件，就会立刻跳出主函数，开始运行事件类模块内的程序。

Python API:

Function: def armor_hit_detection_all(msg)
def armor_hit_detection_bottom_right(msg)

def armor_hit_detection_bottom_left(msg)
def armor_hit_detection_bottom_front(msg)
def armor_hit_detection_bottom_back(msg)
def armor_hit_detection_top_right(msg)
def armor_hit_detection_top_left(msg)

Type: Event callback

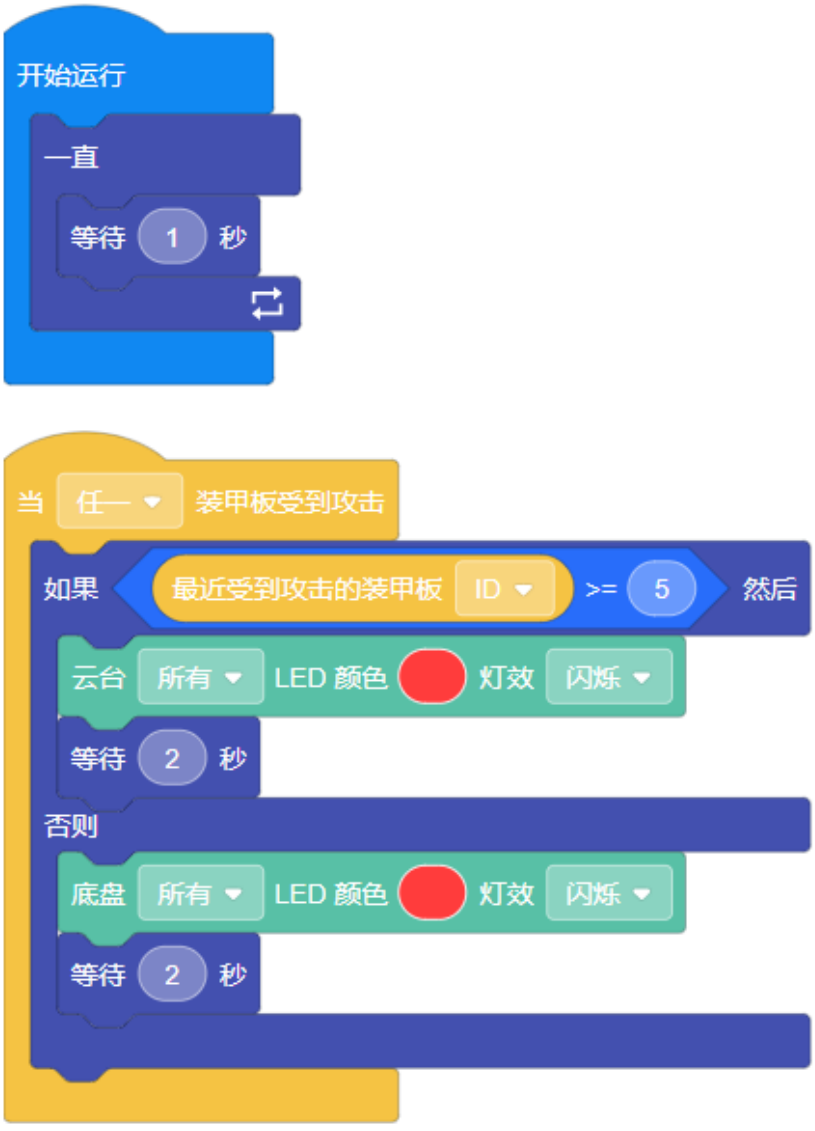
3、最近受到攻击的装甲板（ID）

（1）含义：获取最近受到攻击的装甲板信息，ID 值反馈出装甲板位置，通过整机运行时间计算可以获知受攻击的时间点

（2）类型：信息类（变量型数据）

（3）范例：挨打示意

如果最近受到攻击的是云台的装甲板，则云台所有 LED 红光闪烁；如果是底盘的装甲板，则底盘所有 LED 红光闪烁示意。



注意：

返回的 ID 值释义：

ID 值为 1 对应受到攻击的装甲板在：底盘后侧
ID 值为 2 对应受到攻击的装甲板在：底盘前侧
ID 值为 3 对应受到攻击的装甲板在：底盘左侧
ID 值为 4 对应受到攻击的装甲板在：底盘右侧
ID 值为 5 对应受到攻击的装甲板在：云台左侧
ID 值为 6 对应受到攻击的装甲板在：云台右侧

Python API：
Function: armor_ctrl.get_last_hit_index()
Return value:
- index(int)

Function: armor_ctrl.get_last_hit_time()
Return value:
- time(float)

4、（任一）装甲板受到攻击

（1）含义：持续检测指定装甲板是否受到攻击，被攻击会返回“真”，否则返回“假”
（2）返回值：布尔型
（3）范例：快速撤退
如果底盘左侧装甲板受到攻击，机器人向右平移撤退；如果底盘前侧装甲板受到攻击，机器人向后平移撤退。

Python API：

Function: armor_ctrl.check_condition(condition_enum)

Parameters:

- condition_enum(enum):
  - rm_define.cond_armor_hit
  - rm_define.cond_armor_bottom_front_hit
  - rm_define.cond_armor_bottom_back_hit
  - rm_define.cond_armor_bottom_left_hit
  - rm_define.cond_armor_bottom_right_hit
  - rm_define.cond_armor_top_left_hit
  - rm_define.cond_armor_top_right_hit

5、等待（任一）装甲板受到攻击

（1）含义：待指定位置的装甲板受到攻击后才会执行下一条指令，否则持续等待
（2）类型：执行类、阻塞型
（3）范例：挨打反击

注意：
对此例程来说，若底盘左侧装甲板未受到攻击，程序会在此模块上停留等待，此时模块保持高亮。

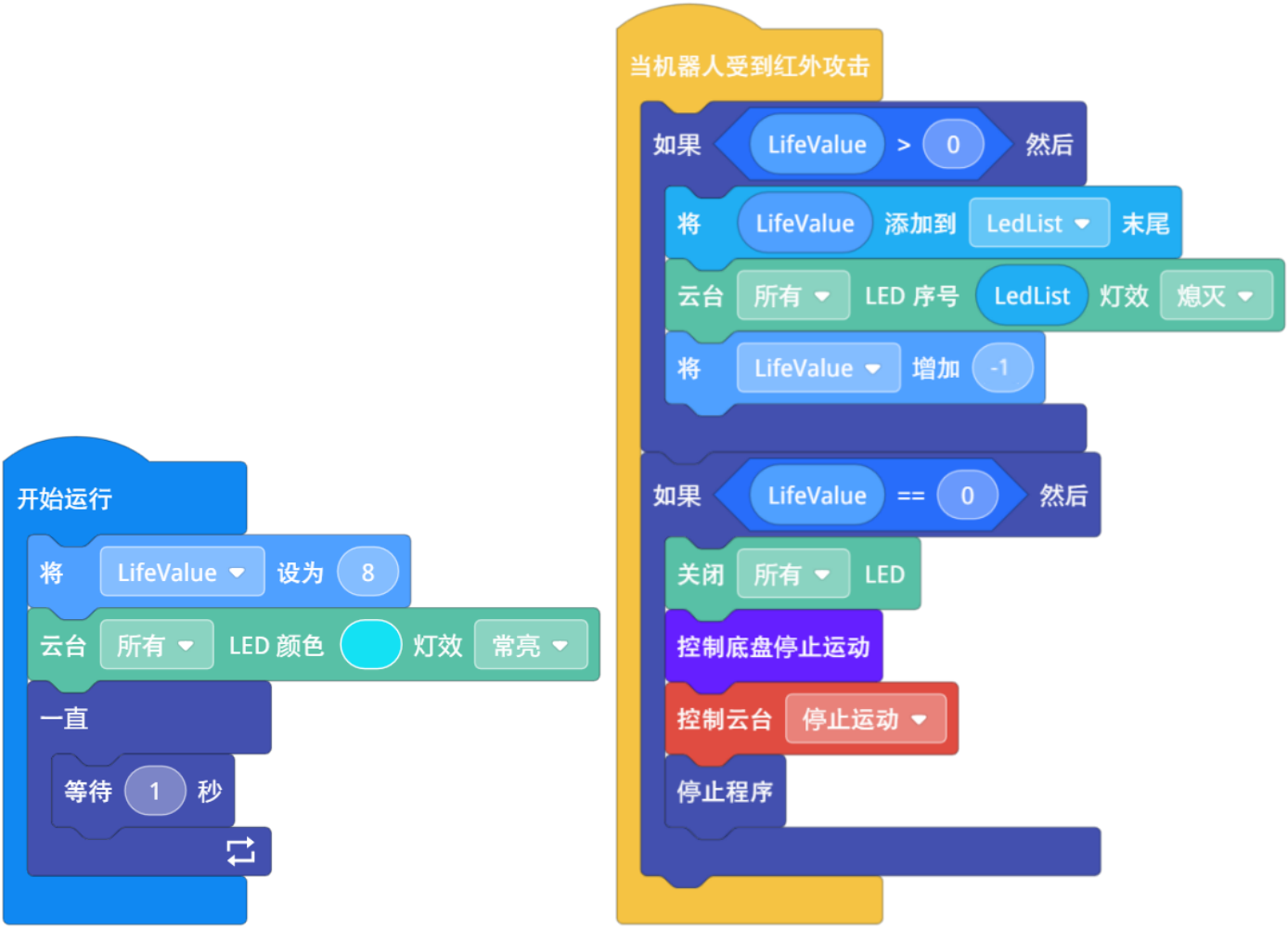
Python API：
Function: armor_ctrl.cond_wait(condition_enum)
Parameters:

- condition_enum(enum):
  - rm_define.cond_armor_hit
  - rm_define.cond_armor_bottom_front_hit
  - rm_define.cond_armor_bottom_back_hit

- rm_define.cond_armor_bottom_left_hit
- rm_define.cond_armor_bottom_right_hit
- rm_define.cond_armor_top_left_hit
- rm_define.cond_armor_top_right_hit

5、当机器人受到红外攻击

（1）含义：当机器人云台两侧的红外传感器受到红外光束攻击时，运行本模块内的程序。
（2）类型：执行类
（3）范例：血条灯
云台 LED 可以示意机器人当前的生命值，满血为 8。
当机器人受到红外攻击，机器人掉 1 血；当生命值为 0 时，所有 LED 熄灭，运动停止。



注意：
事件触发模块优先级高，也就是说无论主函数运行到哪一步，只要满足触发条件，就会立刻跳出主函数，开始运行事件类模块内的程序。

6、等待机器人受到红外攻击

（1）含义：待机器人云台两侧的红外传感器受到红外光束攻击后才会执行下一条指令，否则将持续等待。
（2）类型：执行类、阻塞型
（3）范例：自旋保护
机器人受到红外光束攻击后，原地旋转以降低被击中概率，并进行反击。

注意：
对此例程来说，若机器人云台上的红外传感器未检测到红外光束攻击，程序会在此模块上停留等待，此时模块保持高亮。

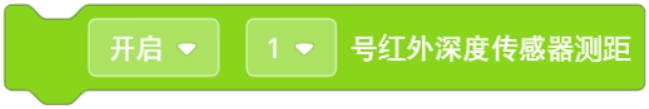
7、机器人受到红外攻击

（1）含义：检测机器人云台两侧的红外传感器是否受到红外光束攻击，受到攻击会返回“真”，否则返回“假”。
（2）返回值：布尔型
（3）范例：被击次数
如果机器人受到红外光束攻击，播放“被击中”音效，并累计被击中次数。

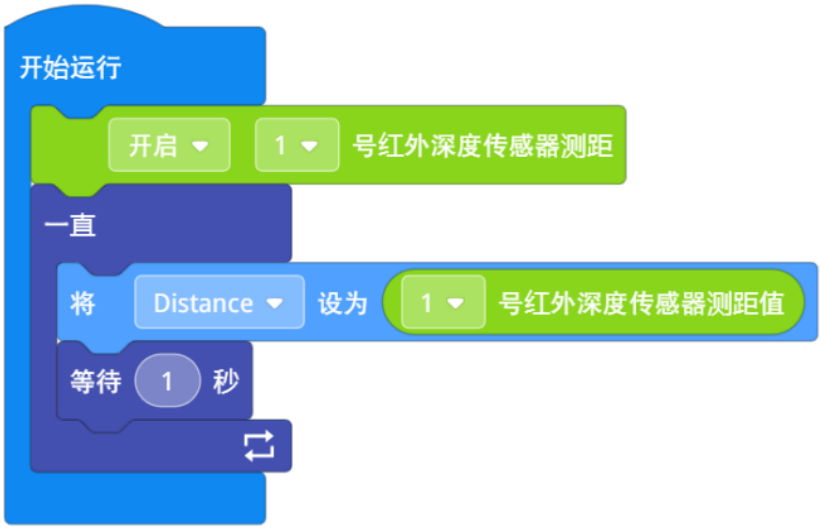

开始运行
将
Times
设为
0
一直
如果
机器人受到红外攻击
然后
播放音效
被击中
将
Times
增加
1

传感器

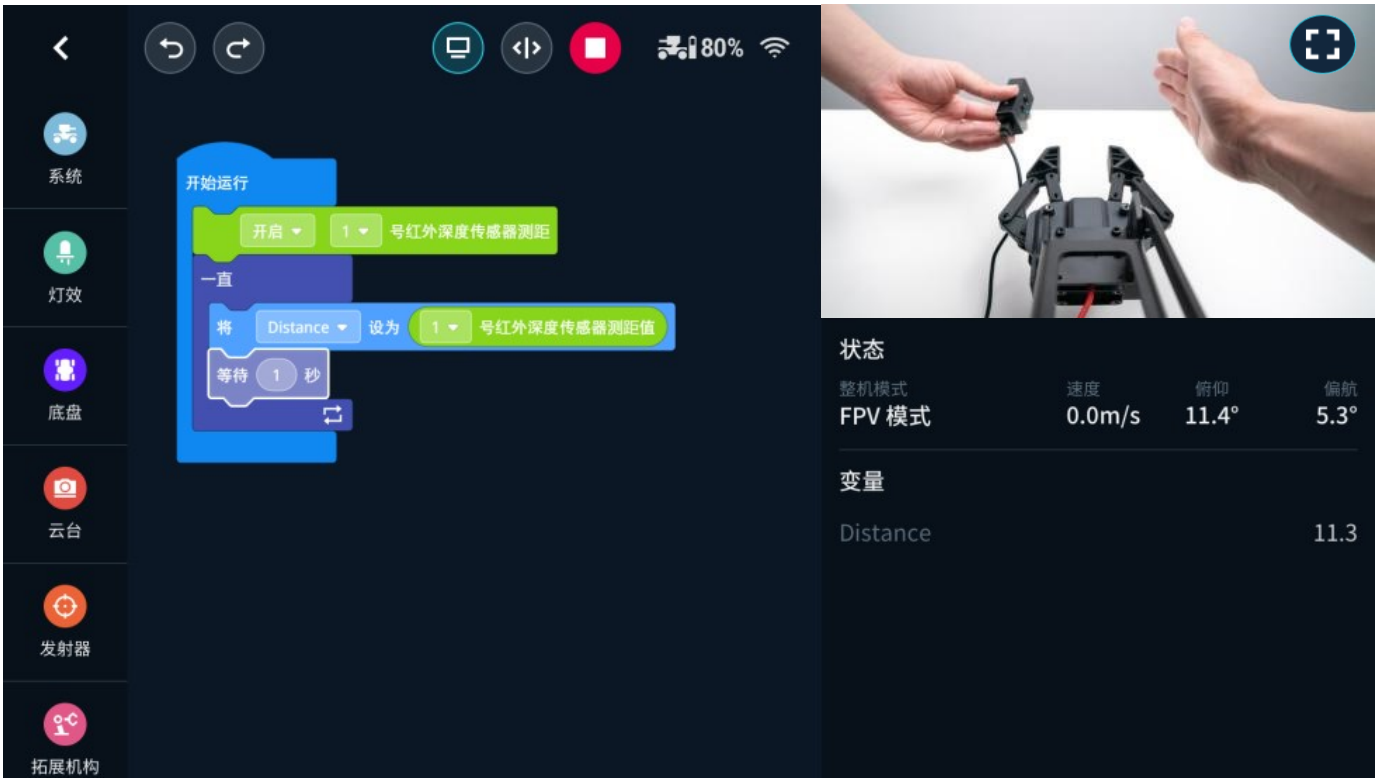
1、（开启）（1）号红外深度传感器测距



（1）含义：开启或关闭指定序号的红外深度传感器的测距功能。
（2）类型：设置类
（3）范例：红外测距
测算红外深度传感器和手掌之间的距离。



不断地前后移动手掌，可以在 FPV 窗口实时观察到距离值的变动。

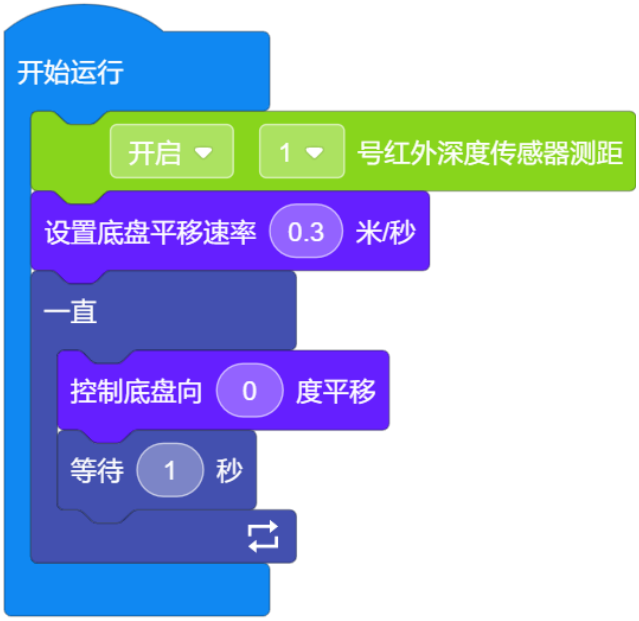


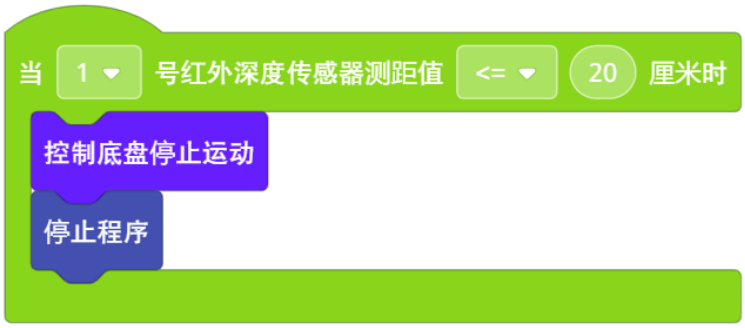
注意：

红外深度传感器的测距功能主要依靠 ToF 相机实现，ToF 是飞行时间（Time of Flight）技术的缩写，也就是传感器发出经过调制的光脉冲（如红外深度传感器发射红外光束），遇到物体后反射，传感器通过计算光线从发射到反射回来的时间差（飞行往返时间），从而换算出传感器和被测物体之间的距离。

2、当（1）号红外深度传感器测距值（≥）（10）厘米时

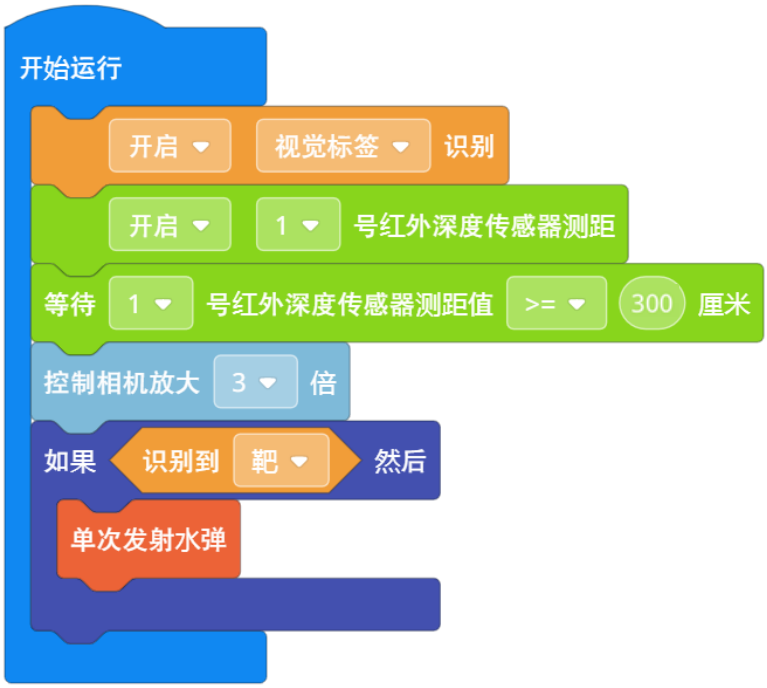
（1）含义：当指定序号的红外深度传感器测算出的距离值满足条件时，运行本模块内的程序。
（2）类型：事件类
（3）范例：不断逼近
控制机器人移动底盘不断平移向墙壁靠近，当红外深度传感器测算出的距离值 ≤ 20 厘米时，停止一切运动。





3、等待（1）号红外深度传感器测距值（≥）（10）厘米

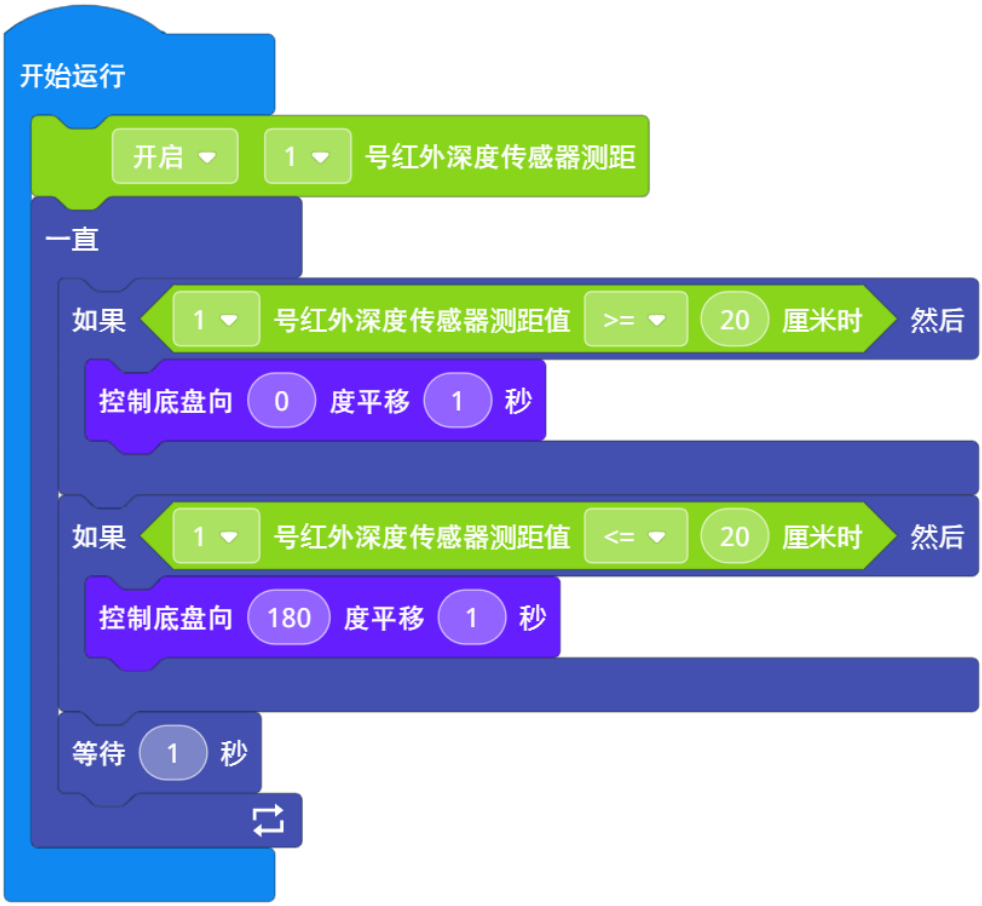
（1）含义：待指定序号的红外深度传感器测距值满足条件时会执行下一条指令，否则将持续等待。
（2）类型：执行类、阻塞型
（3）范例：远距离打靶
手持视觉标签不断远离机器人，待红外深度传感器检测到“靶”已距离机器人达 3 米之远，将通过放大相机倍镜远程射击。



4、（1）号红外深度传感器测距值（≥）（10）厘米时

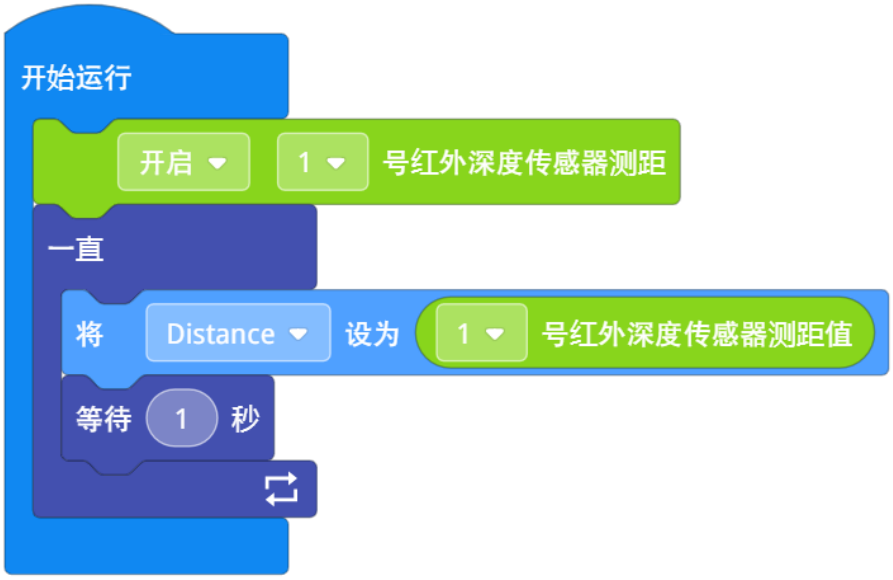
（1）含义：指定序号红外深度传感器测距值满足条件时返回“真”，否则返回“假”。
（2）返回值：布尔型
（3）范例：你进我退

机器人为了保证和手掌之间保持着合适的距离，熟练掌握了“你进我退、你退我进”策略：如果红外深度传感器测算到和手之间的距离值 ≤ 20 厘米，机器人向后平移；如果距离值 ≥ 20 厘米，机器人向前平移。

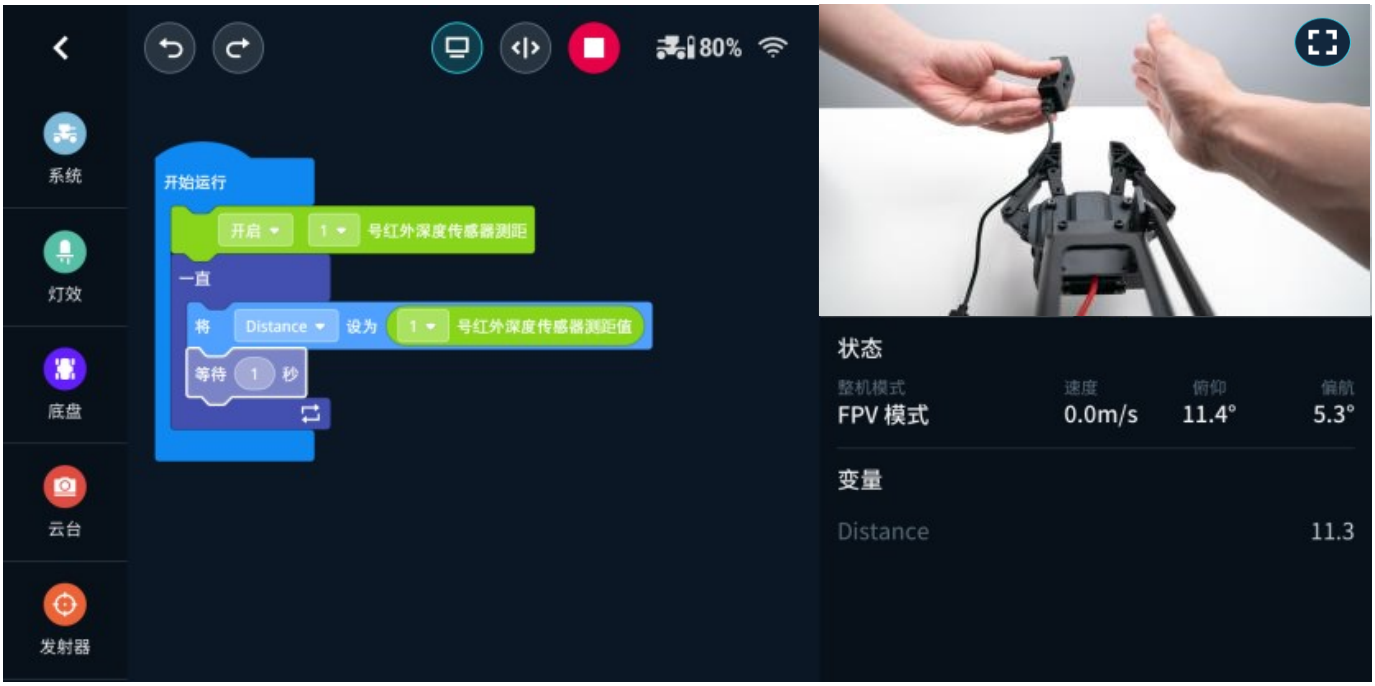


5、（1）号红外深度传感器测距值

（1）含义：获取指定序号红外深度传感器测算出的距离数值，单位为厘米。
（2）类型：信息类
（3）范例：红外测距
测算红外深度传感器和手掌之间的距离。



不断地前后移动手掌，可以在 FPV 窗口实时观察到距离值的变动。



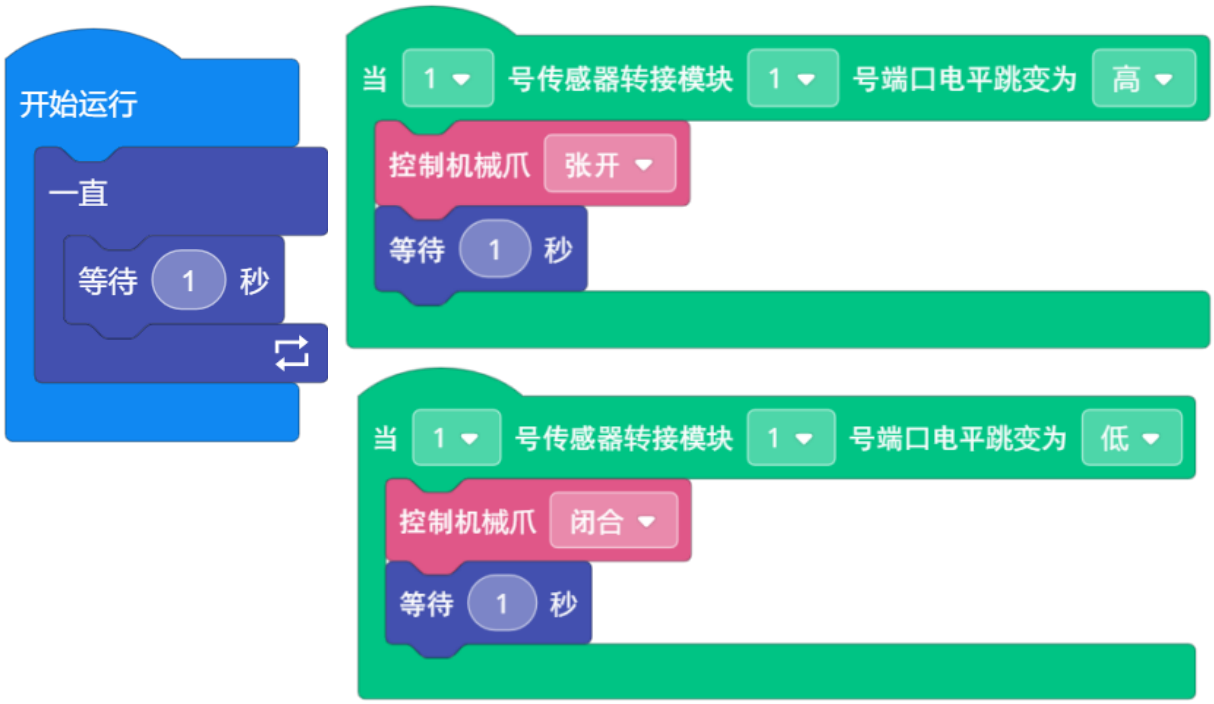
注意：

红外深度传感器可读取距离为 10 厘米~10 米。

转接模块

1、当（1）号传感器转接模块（1）号端口电平跳变为（高）

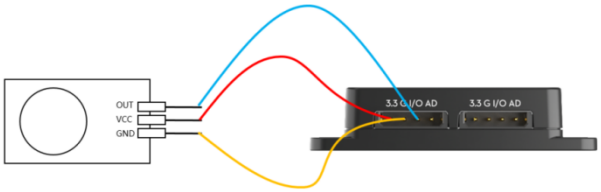
（1）含义：当指定传感器转接模块指定端口的 I/O 引脚电平满足条件时，执行本模块内的程序。
（2）类型：事件类
（3）范例：按钮控制机械爪开合
在 1 号传感器转接模块处接入按钮。当按住按钮时，机器人机械爪张开；当松开按钮时，机械爪闭合。



注意：

1）按钮输出电平有高、低之分，按钮在被“按下”和“松开”的时候，输出的电平会发生跳变，传感器转接模块的 I/O 引脚接收到该电平信号的变化从而控制机械爪实现张开、闭合两种动作。
2）按钮上的三个引脚需要通过杜邦线分别连接至传感器转接模块指定位置：

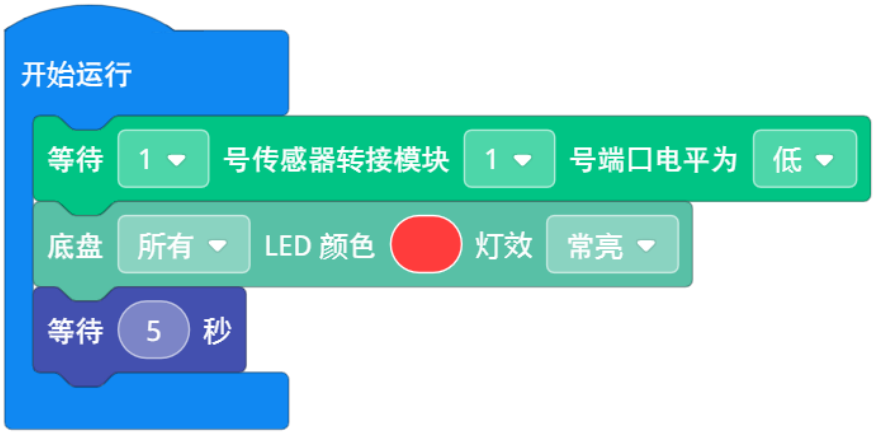
按钮引脚	传感器转接模块接口
OUT（输出）	I/O（输入/输出）
VCC（供电）	3.3（3.3 V 供电）
GND（接地）	G（接地）



3）机器人最多支持同时接入 6 个传感器转接模块。

2、等待（1）号传感器转接模块（1）号端口电平为 （高）

（1）含义：待指定传感器转接模块指定端口的 I/O 引脚电平满足条件时执行后续指令，否则将持续等待。
（2）类型：执行类、阻塞型
（3）范例：分贝过高提示
在 1 号传感器转接模块处接入声音传感器，待检测到的声音过大后，机器人底盘红灯常亮。



注意：
1）此例程设计时使用的声音传感器接收到的声音信号若达不到传感器的设定阈值会输出高电平，超过阈值输出低电平，但不同类型的声音传感器情况不一。
2）像红外避障传感器、触摸传感器等输出只有高低电平，故可通过转接模块里检测高低电平的 I/O 口获取功能区采集。

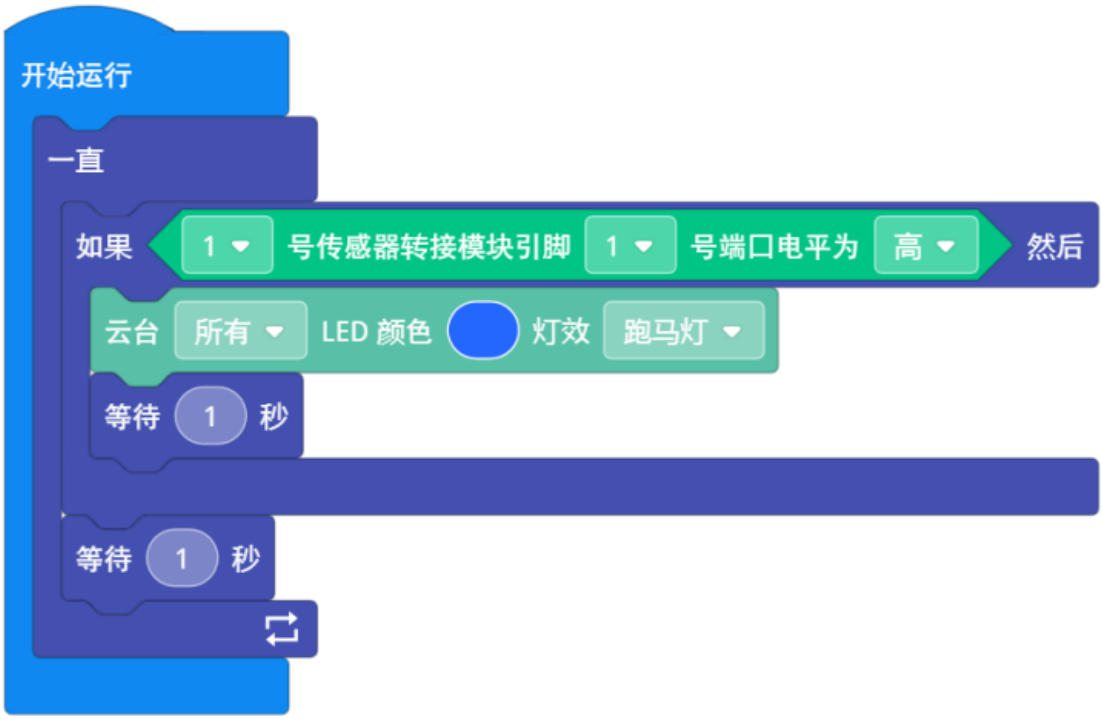
触发型			
红外避障传感器		VCC(5V) GND OUT(TTL)	高低电平
震动传感器		VCC(5V) GND OUT(TTL)	高低电平
倾斜传感器		VCC(5V) GND OUT(TTL)	高低电平
声音传感器		VCC(5V) GND OUT(TTL)	高低电平
码盘计数		VCC(5V) GND OUT(TTL)	高低电平
触摸传感器		VCC(5V) GND OUT(TTL)	高低电平

3、（1）号传感器转接模块（1）号端口电平为 （高）

（1）含义：指定传感器转接模块的指定端口 I/O 引脚电平满足条件时返回“真”，否则返回“假”。
（2）返回值：布尔型

（3）范例：触摸灯效

在 1 号传感器转接模块处接入触摸传感器，手指触摸传感器后云台以蓝色跑马灯灯效示意。



注意：

此例程设计时使用的触摸传感器若手指和其接触会输出高电平，没有触摸时为低电平，但不同类型的触摸传感器情况不一。

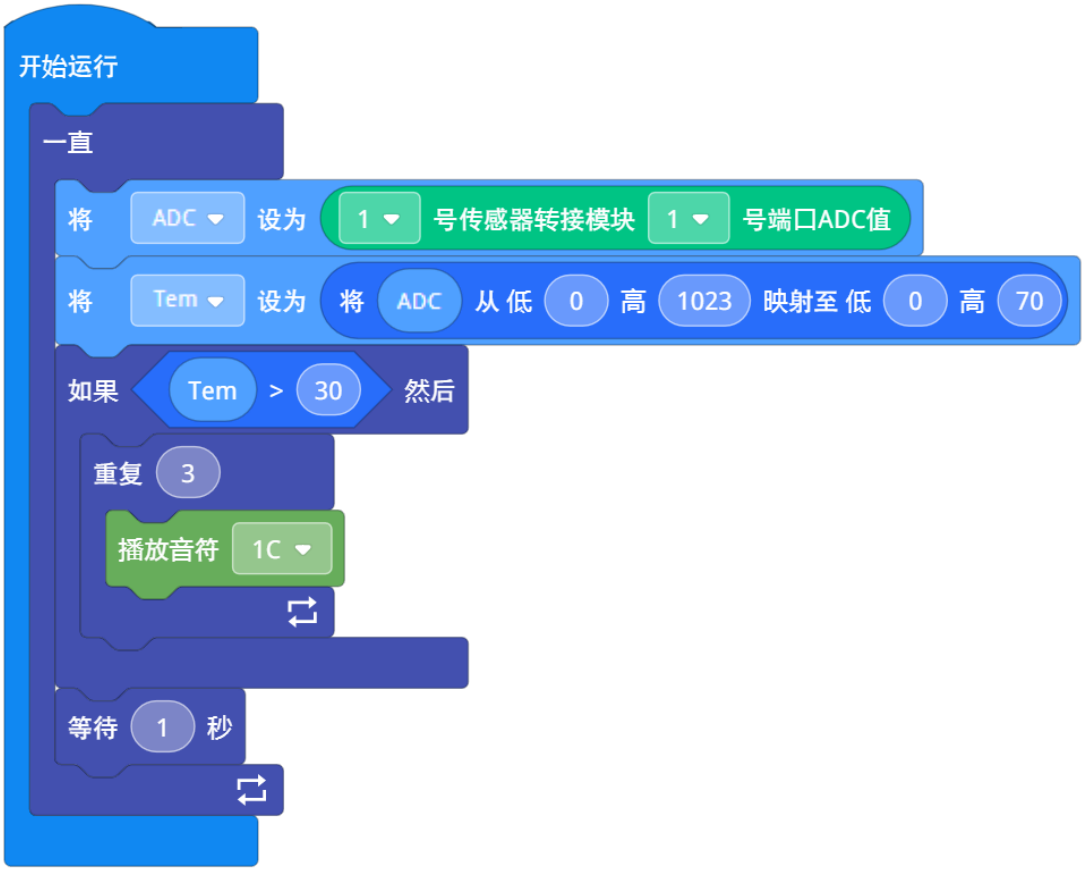
4、（1）号传感器转接模块（1）号端口 ADC 值

（1）含义：读取指定序号的传感器转接模块指定端口的 ADC 引脚数值。

（2）返回值：信息类

（3）范例：温度报警器

在 1 号传感器转接模块处接入温度传感器。当温度传感器靠近装满热水的杯子时播放音符警示。



注意：

1）传感器转接模块 ADC 取样位数为 10 Bit，精度单位 1024 级，因此取值范围是 0-1023。

2）ADC 采集的是电压值，像压力传感器、电位器等都有 ADC 值输出。

触发型+ADC 型

光敏电阻传感器模块		VCC(5V) GND OUT(TTL) OUT(AD)	高低电平+ADC
水位传感器		VCC(5V) GND OUT(AD)	ADC
霍尔传感器		VCC(5V) GND OUT(AD) OUT(TTL)	高低电平+ADC
火焰传感器模块		VCC(5V) GND OUT(AD) OUT(TTL)	高低电平+ADC

寻迹传感器		VCC(5V) GND OUT(AD) OUT(TTL)	高低电平+ADC
烟雾传感器		VCC(5V) GND OUT(AD) OUT(TTL)	高低电平+ADC

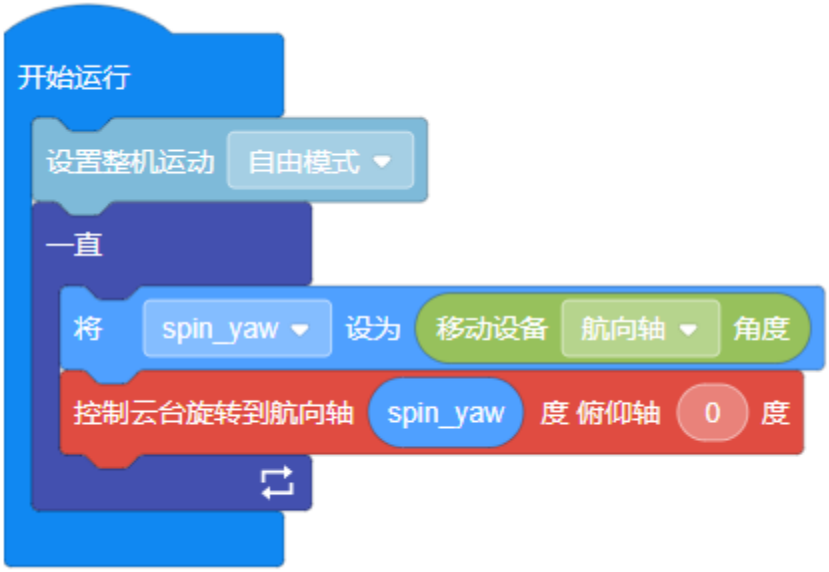
5、（1）号传感器转接模块（1）号端口脉冲持续时间

（1）含义：获取指定传感器转接模块指定端口 I/O 引脚脉冲从发生跳变到获取时刻的时间，单位毫秒。
（2）返回值：信息类
（3）范例：拍照
在 1 号传感器转接模块接入按钮。如果按下按钮超过 3 秒，开始拍照。

移动设备

1、移动设备（航向轴）角度

（1）含义：获取移动设备当前角度值
（2）类型：信息类（变量型数据）
（3）范例：体感控制旋转
左右转动手中的移动设备，观察 RoboMaster EP 云台是否会跟随旋转。



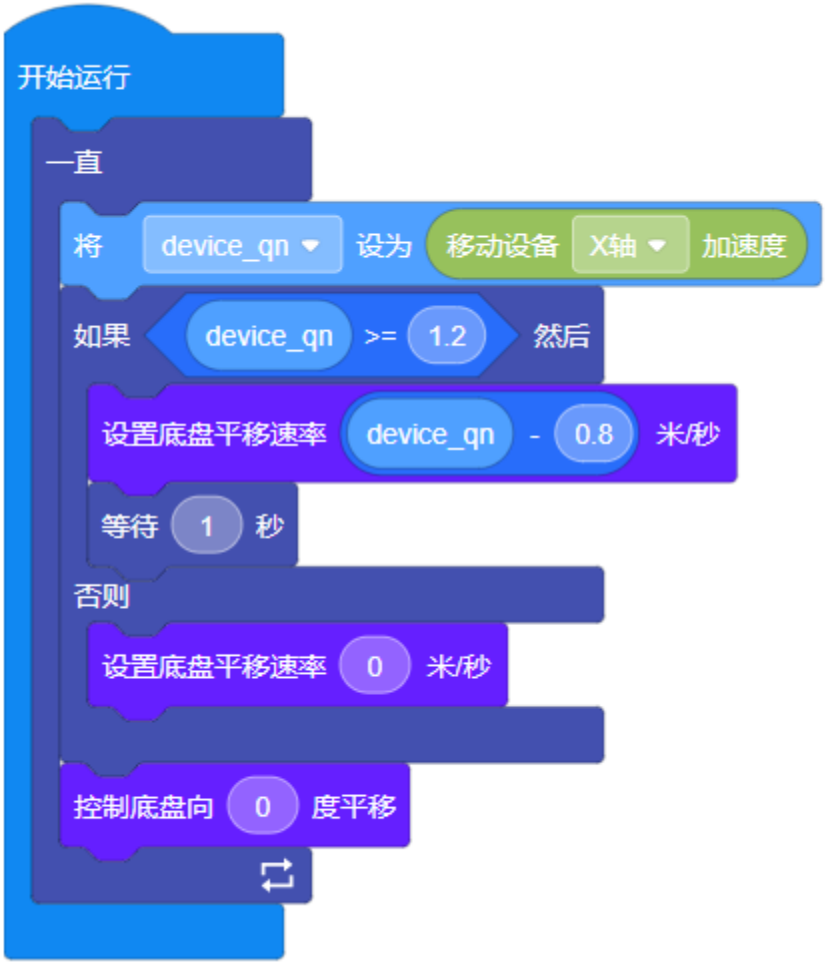
注意：
1）RoboMaster EP 的默认运动模式是“云台跟随底盘模式”，所以如果想单独控制云台绕航向轴旋转，需要设置为“自由模式”。
2）移动设备是指手机、平板等。
3）移动设备姿态角范围是 -180~180°，
对航向轴来说：向右为正（0-180）
对俯仰轴来说：向上为正（0-180）
对翻滚轴来说：右下为正（0-180）

Python API：
Function: mobile_ctrl.get_attitude(attitude_enum)
Parameters:
- attitude_enum(enum):
  - rm_define.mobile_atti_pitch
  - rm_define.mobile_atti_roll
  - rm_define.mobile_atti_yaw

2、移动设备（X 轴）加速度

（1）含义：获取移动设备加速度单元的测量值
（2）类型：信息类（变量型数据）
（3）范例：体感控制前进

上下晃动手中的移动设备，控制 RoboMaster EP 前进。



注意：
(1）速度变化越快，获取到的加速度测量值越大。
(2）移动设备是指手机、平板等。

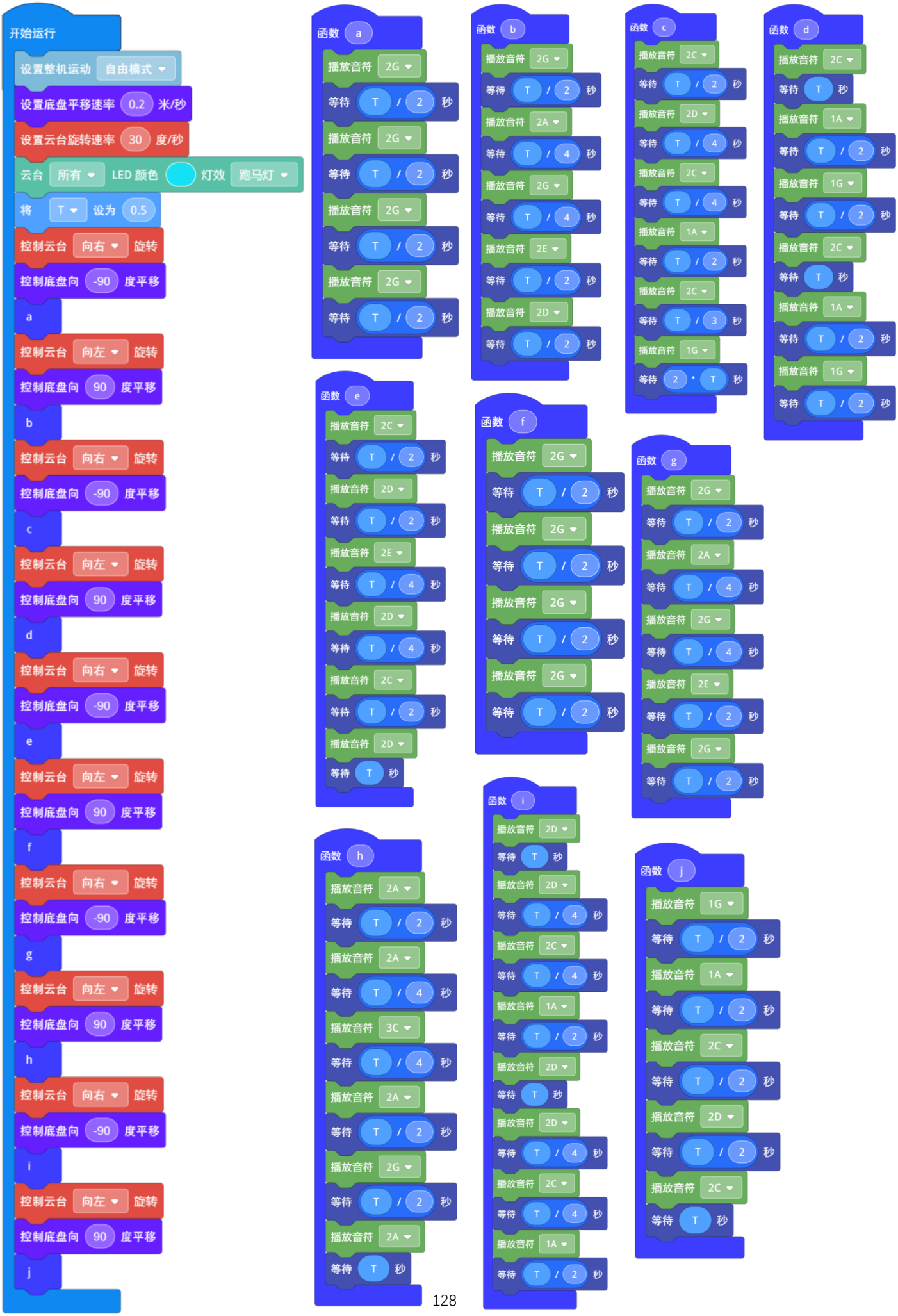
Python API:
Function: mobile_ctrl.get_accel(axis_enum)
Parameters:

- axis_enum(enum):
  - rm_define.mobile_accel_x
  - rm_define.mobile_accel_y
  - rm_define.mobile_accel_z

多媒体

1、播放音符（1C）

（1）含义：选择音符并以钢琴音色播放。
（2）类型：执行类、非阻塞型
（3）范例：粉红色的回忆
控制机器人播放《粉红色的回忆》旋律。


开始运行
设置整机运动 自由模式
设置底盘平移速率 0.2 米/秒
设置云台旋转速率 30 度/秒
云台 所有 LED 颜色 灯效 跑马灯
将 T 设为 0.5
控制云台 向右 旋转
控制底盘向 -90 度平移
a
控制云台 向左 旋转
控制底盘向 90 度平移
b
控制云台 向右 旋转
控制底盘向 -90 度平移
c
控制云台 向左 旋转
控制底盘向 90 度平移
d
控制云台 向右 旋转
控制底盘向 -90 度平移
e
控制云台 向左 旋转
控制底盘向 90 度平移
f
控制云台 向右 旋转
控制底盘向 -90 度平移
g
控制云台 向左 旋转
控制底盘向 90 度平移
h
控制云台 向右 旋转
控制底盘向 -90 度平移
i
控制云台 向左 旋转
控制底盘向 90 度平移
j
函数 a
播放音符 2G
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
函数 b
播放音符 2G
等待 T / 2 秒
播放音符 2A
等待 T / 4 秒
播放音符 2G
等待 T / 4 秒
播放音符 2E
等待 T / 2 秒
播放音符 2D
等待 T / 2 秒
函数 c
播放音符 2C
等待 T / 2 秒
播放音符 2D
等待 T / 4 秒
播放音符 2C
等待 T / 4 秒
播放音符 1A
等待 T / 2 秒
播放音符 2C
等待 T / 3 秒
播放音符 1G
等待 2 * T 秒
函数 d
播放音符 2C
等待 T 秒
播放音符 1A
等待 T / 2 秒
播放音符 1G
等待 T / 2 秒
播放音符 2C
等待 T 秒
播放音符 1A
等待 T / 2 秒
播放音符 1G
等待 T / 2 秒
函数 e
播放音符 2C
等待 T / 2 秒
播放音符 2D
等待 T / 2 秒
播放音符 2E
等待 T / 4 秒
播放音符 2D
等待 T / 4 秒
播放音符 2C
等待 T / 2 秒
播放音符 2D
等待 T 秒
函数 f
播放音符 2G
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
函数 g
播放音符 2G
等待 T / 2 秒
播放音符 2A
等待 T / 4 秒
播放音符 2G
等待 T / 4 秒
播放音符 2E
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
函数 h
播放音符 2A
等待 T / 2 秒
播放音符 2A
等待 T / 4 秒
播放音符 3C
等待 T / 4 秒
播放音符 2A
等待 T / 2 秒
播放音符 2G
等待 T / 2 秒
播放音符 2A
等待 T 秒
函数 i
播放音符 2D
等待 T 秒
播放音符 2D
等待 T / 4 秒
播放音符 2C
等待 T / 4 秒
播放音符 1A
等待 T / 2 秒
播放音符 2D
等待 T 秒
播放音符 2D
等待 T / 4 秒
播放音符 2C
等待 T / 4 秒
播放音符 1A
等待 T / 2 秒
函数 j
播放音符 1G
等待 T / 2 秒
播放音符 1A
等待 T / 2 秒
播放音符 2C
等待 T / 2 秒
播放音符 2D
等待 T / 2 秒
播放音符 2C
等待 T 秒

2、播放音效（被击中）

（1）含义：播放音效的同时，立刻执行下一条指令
（2）类型：执行类、非阻塞型
（3）范例：扫描中

在底盘向前平移，云台绕航向轴旋转的过程中播放“扫描中”音效。

Python API：
Function: media_ctrl.play_sound(sound_enum, wait_complete_flag=False)
Parameters:

- sound_enum(enum):
  - rm_define.media_sound_attacked
  - rm_define.media_sound_shoot
  - rm_define.media_sound_scanning
  - rm_define.media_sound_recognize_success
  - rm_define.media_sound_gimbal_rotate
  - rm_define.media_sound_count_down

3、播放音效（被击中）直到结束

（1）含义：音效播放完毕后才会执行下一条指令
（2）类型：执行类、阻塞型
（3）范例：遥控拍照

如果 RoboMaster EP 识别到拍照手势，会在“倒计时开始”音效播放完毕后进行拍照；如果未识别到拍照手势，RoboMaster EP 会边播放“扫描中”音效边旋转搜寻手势指令。

注意：

“播放音效”是播放音效和后续指令并行，是伴奏（比如歌伴舞）。

“播放音效直到结束”是播放音效和后续指令逐个进行，是独奏（比如音效演奏完毕后才开始跳舞）。

Python API：

Function: media_ctrl.play_sound(sound_enum, wait_complete_flag=True)

Parameters:

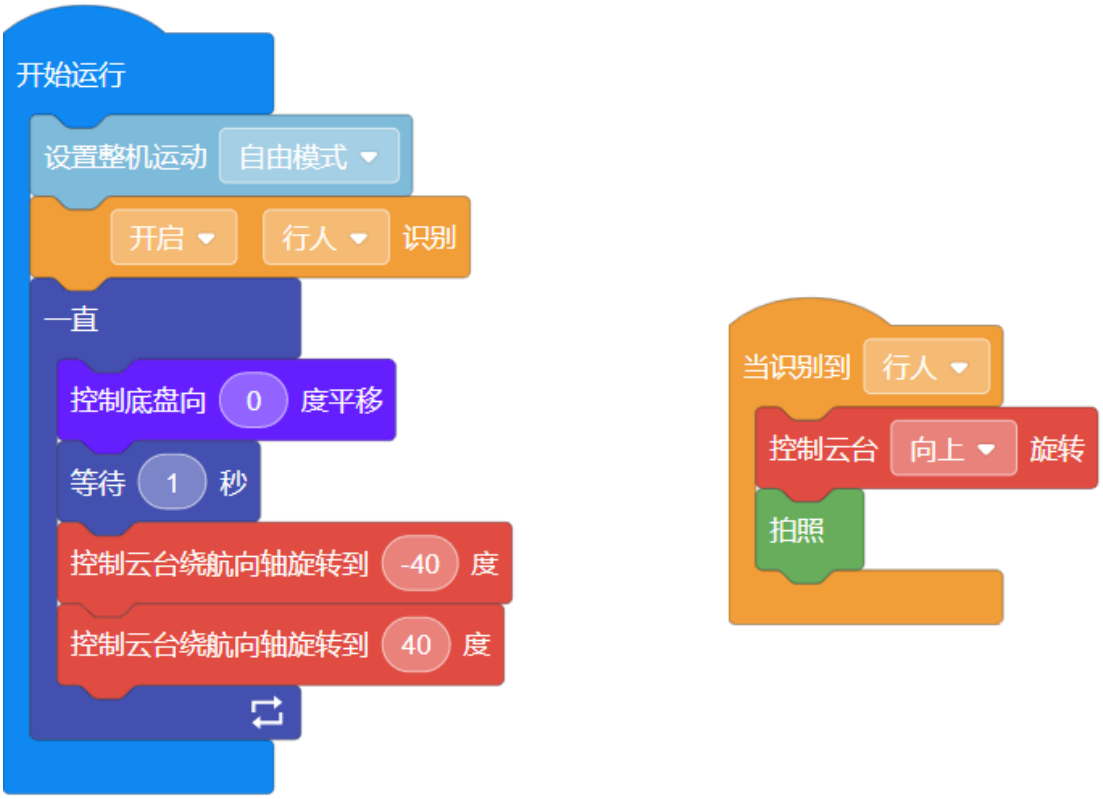
- sound_enum(enum):
  - rm_define.media_sound_attacked
  - rm_define.media_sound_shoot
  - rm_define.media_sound_scanning
  - rm_define.media_recognize_success
  - rm_define.media_gimbal_rotate
  - rm_define.media_count_down

4、拍照

（1）含义：响起快门声的同时拍摄一张照片，照片会在相册中出现。

（2）类型：执行类

（3）范例：行人抓拍



注意：

请提前在机器人中插入一张剩余用量在 2GB 以上的 SD 卡，无 SD 卡时拍照指令不生效。

Python API：

Function: media_ctrl.capture()

5、（开始）视频录制

（1）含义：开始或结束视频录制，结束后会在 APP 相册和 SD 卡中生成一段视频。
（2）类型：执行类
（3）范例：视频录制

注意：
机器人中插入的 SD 卡剩余用量需在 2GB 以上。

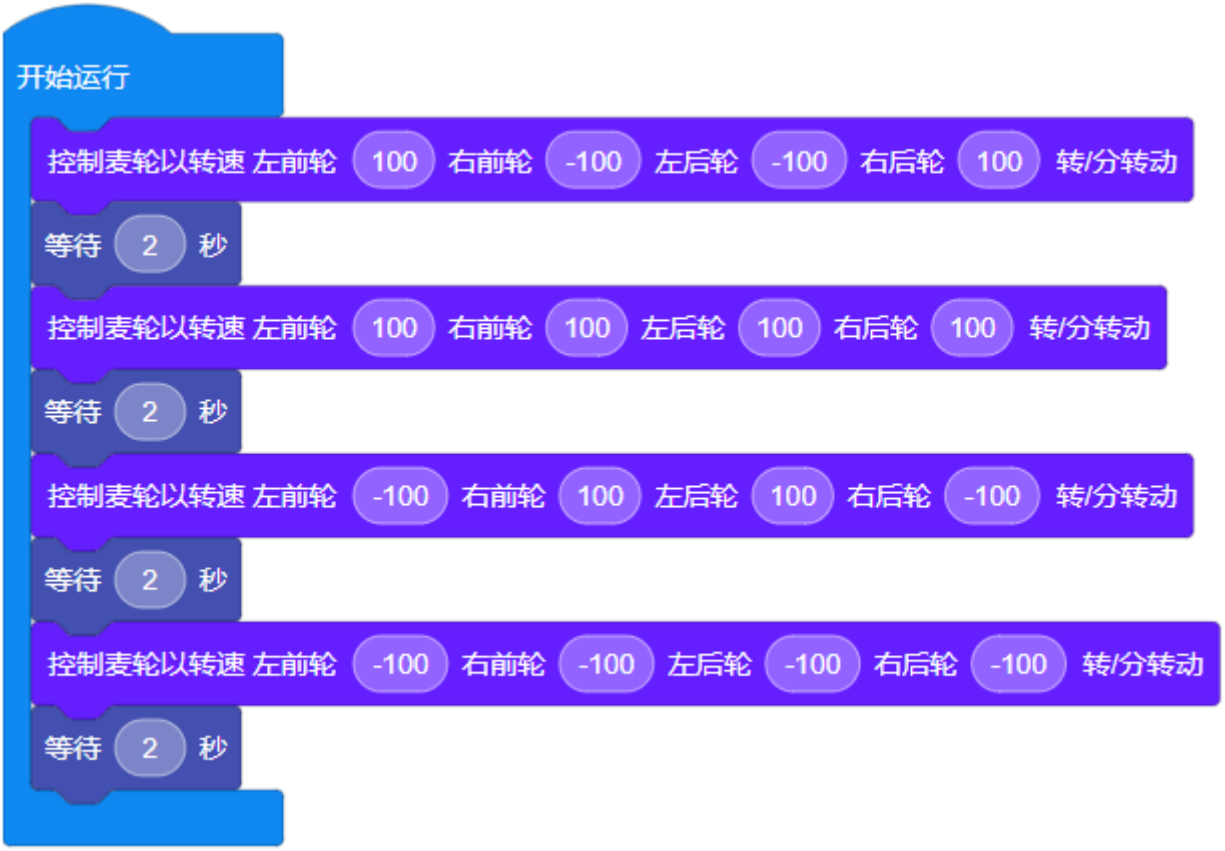
Python API：
Function: media_ctrl.record(enable_enum)
Parameters:

- enable_enum(enum):
  - 1
  - 0

控制语句

1、等待（1）秒

（1）含义：等待指定秒数，然后再执行下一条指令
（2）类型：执行类
（3）范例：方形走位
通过每隔两秒改变麦轮的转速，让 RoboMaster EP 按着“向右→向前→向左→向后”的路径平移，完成方形走位。

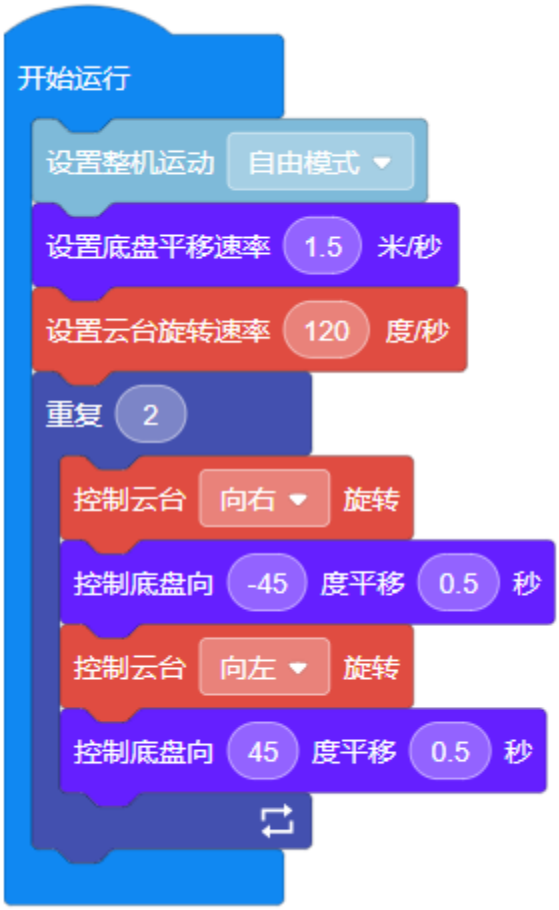


Python API
Function: time.sleep(t)
Parameters:

- t(float): [0, 3600]s

2、重复（10）

（1）含义：重复运行内部程序若干次（有限循环）
（2）类型：执行类
（3）范例：折线运动
控制机器人走出折线轨迹。



3、一直

（1）含义：持续地重复运行内部程序（无限循环）
（2）类型：执行类
（3）范例：底盘左右旋转

4、如果...然后...

（1）含义：如果条件成立，则运行内部程序
（2）类型：条件类
（3）范例：自转公转
机器人在自身旋转的过程中做环绕运动。

注意：
条件类和事件类的区别是：事件触发的优先级更高，只要满足触发条件，无论主函数现在进行到哪一步都会立刻中断，先完成事件类模块内的程序；条件类是得刚好运行到此模块时“条件满足”，才会运行模块内程序。

5、如果...然后...否则...

（1）含义：如果条件成立，运行"然后"内的程序；如果不成立，运行"否则"内的程序
（2）类型：条件类

（3）范例：限制发弹数

发弹数逐次递增，达到最大限度“8 颗/次”后又从“1 颗/次”重新开始。

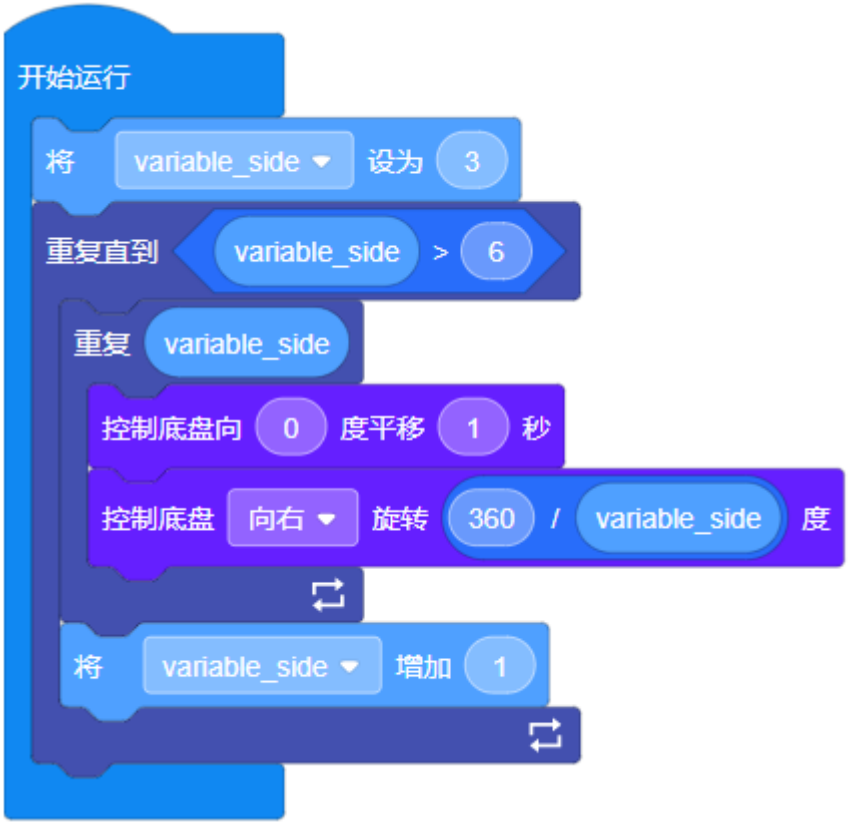
6、重复直到...

（1）含义：重复运行内部程序直到条件成立。即：条件不成立时运行内部程序；条件成立时跳出循环，执行下一条指令

（2）类型：条件类

（3）范例：多边形走位

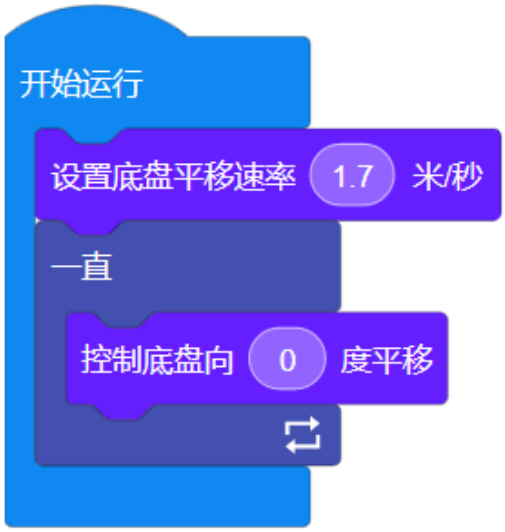
让 RoboMaster EP 依次走出正三角形、正方形、正五边形和正六边形的轨迹。

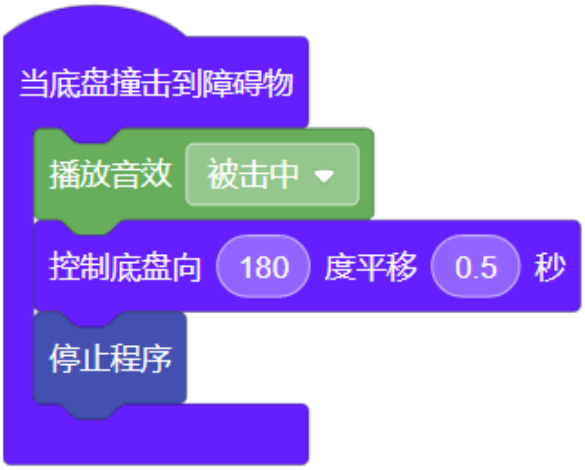


注意：
为了保证循环次数不多也不少，在填写判断条件时要考虑清楚。

7、停止程序

（1）含义：停止正在运行的所有程序
（2）类型：执行类
（3）范例：停止程序
当底盘撞击到障碍物时，后退 0.5 秒后停止正在运行的所有程序。



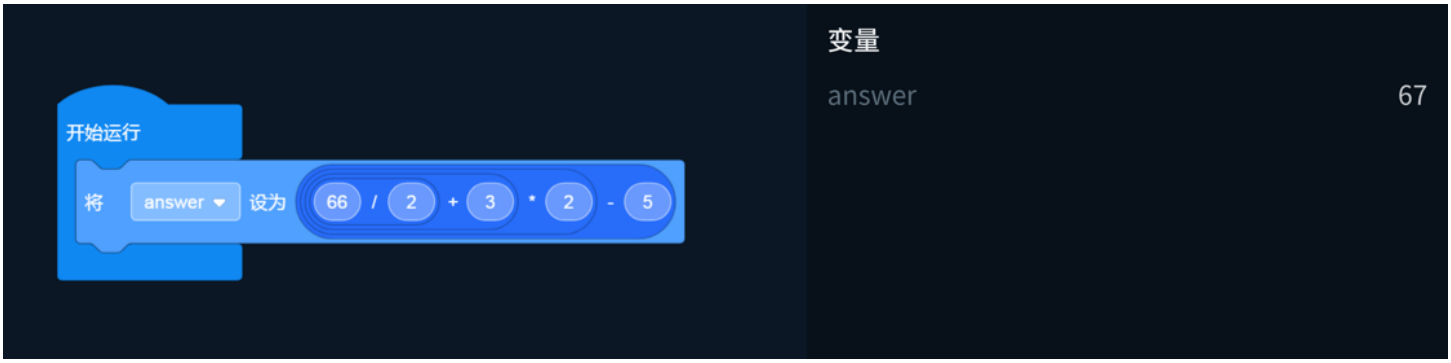


运算符

1、（O）+（O）

（1）含义：两数相加
（2）返回值：变量型数据
（3）范例：四则运算
计算【（66÷2）+3】×2-5=?

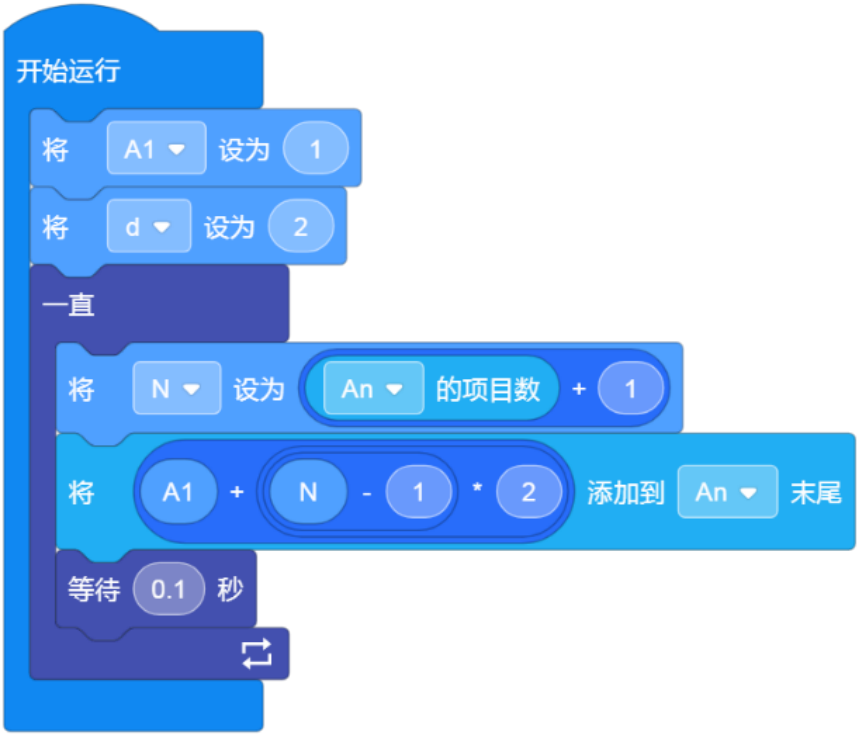
可以在 FPV 窗口观察运算结果。



注意：
不能对列表进行四则运算。

2、（O）-（O）

（1）含义：两数相减
（2）返回值：变量型数据
（3）范例：等差数列
根据等差数列公式 An=A1+（n-1）*d，计算首项为 1，公差为 2 的等差数列。



在 FPV 窗口观察：

1	1	2	3
3	5	4	7
5	9	6	11
7	13	8	15
9	17	10	19
11	21	12	23
13	25	14	27
15	29	16	31
17	33	18	35
19	37	20	39
21	41	22	43
23	45	24	47

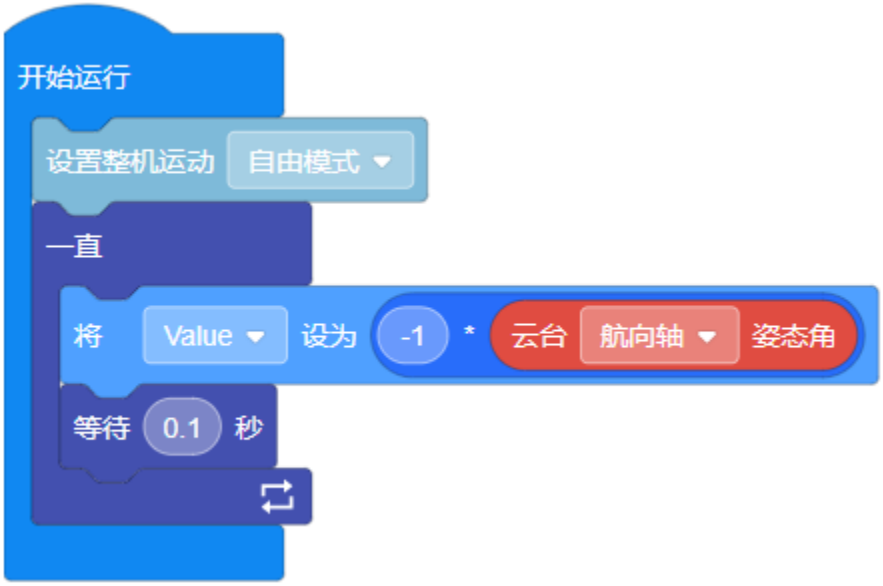
注意：
不能对列表进行四则运算。

3、（O）*（O）

（1）含义：两数相乘
（2）返回值：变量型数据
（3）范例：反向显示

用手拨动云台左右旋转，在 FPV 窗口观察 Value 的变化。

因为反向显示，所以当云台在底盘右侧旋转时，Value 为负值；当云台在底盘左侧旋转时，Value 为正值。



注意：
不能对列表进行四则运算。

4、（0）/（0）

（1）含义：两数相除
（2）返回值：变量型数据
（3）范例：漂移
控制机器人漂移停车。

注意：
不能对列表进行四则运算。

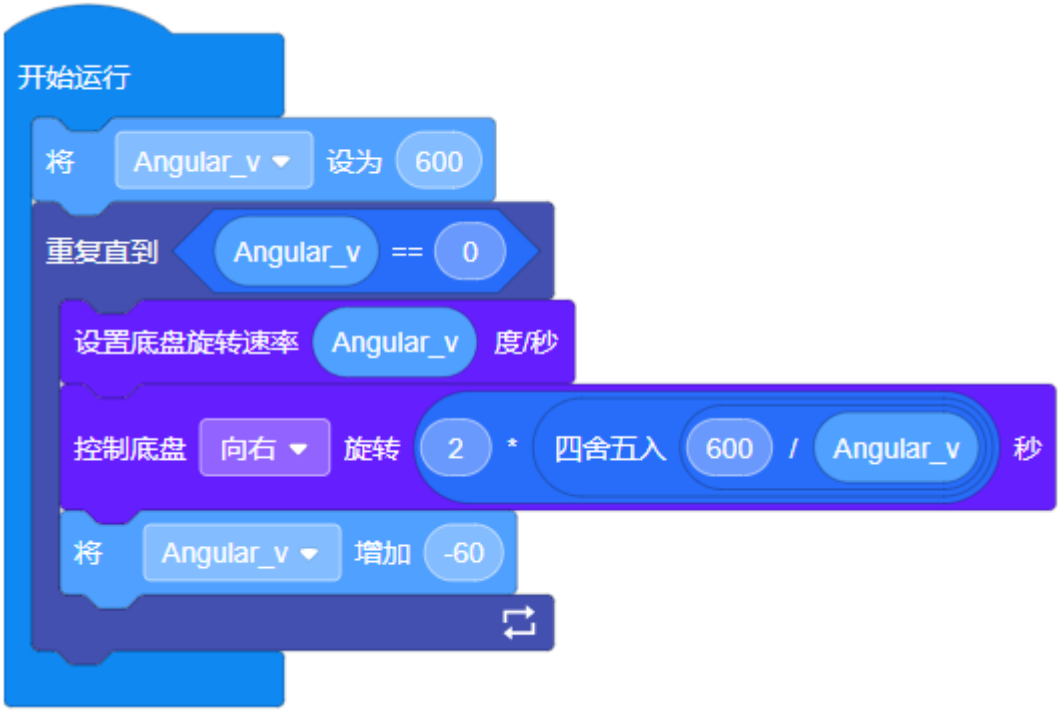
5、在（1）到（10）间随机选一个数

（1）含义：在指定范围内随机选取一个数值
（2）类型：信息类（变量型数据）
（3）范例：人机猜拳
输入你的答案，和机器人产生的随机值比较大小。如果你的答案大于或等于机器人的答案，云台以跑马灯特效庆贺；如果你的答案小于机器人的答案，云台会“垂头丧气”。

注意：
如果数值范围输入的都是整数，输出值就都是整数；
如果数值范围有一个小数（无论是起始数还是终止数），输出值就都是小数。

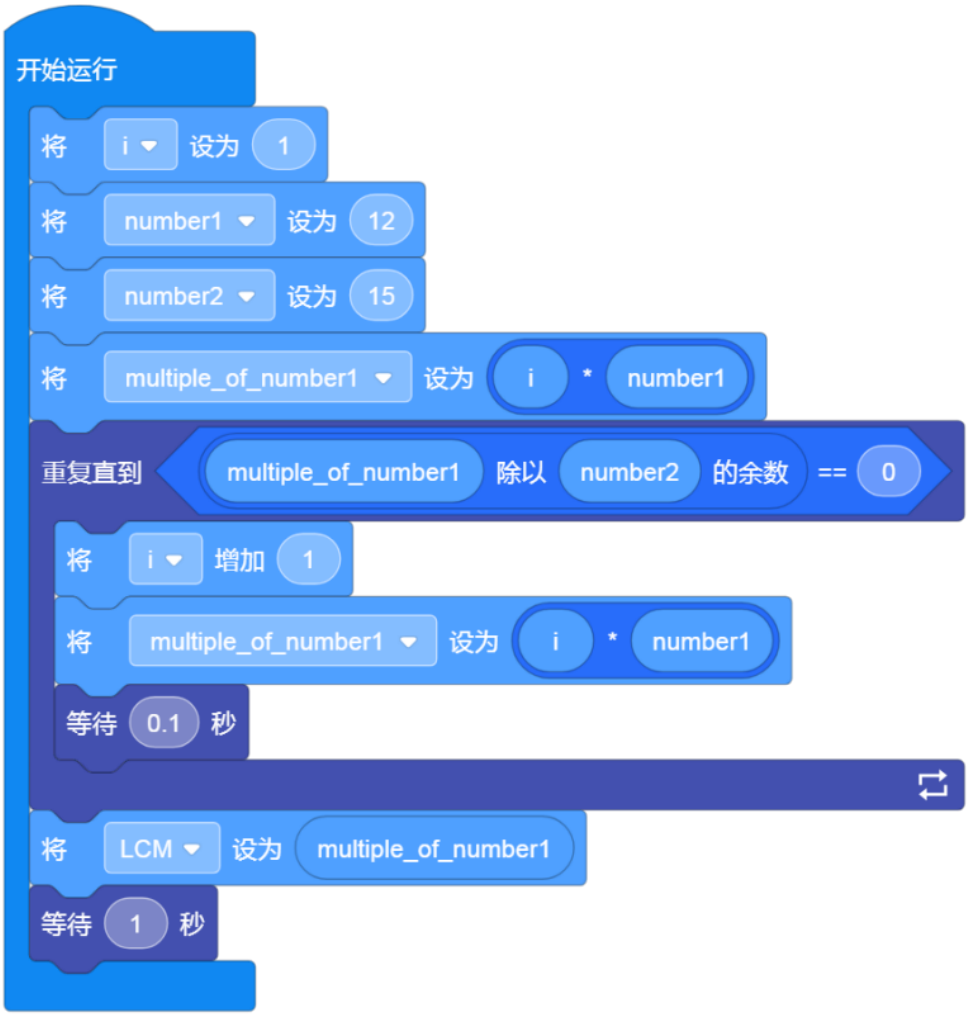
6、四舍五入（0）

（1）含义：取整，通过四舍五入获取最接近此数值的整数
（2）类型：信息类（变量型数据）
（3）范例：旋转减速
控制底盘在向右旋转的过程中，旋转速率由 600 度/秒逐渐减速到 60 度/秒。



7、（0）除以（1）的余数

（1）含义：取模，获取第一个值除以第二个值所得的余数
（2）类型：信息类（变量型数据）
（3）范例：求最小公倍数
求 12 和 15 的最小公倍数。



可以在 FPV 窗口观察运算结果，LCM=60。

注意：
LCM 是最小公倍数的缩写。

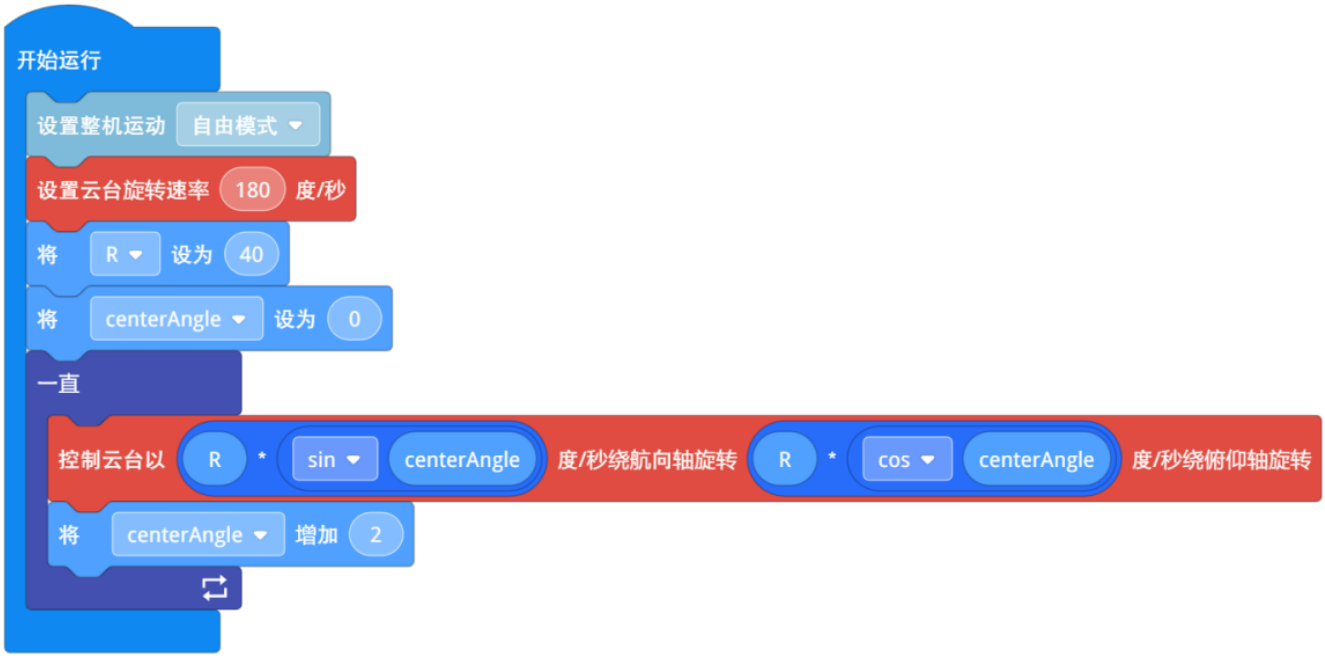
8、（绝对值）（0）

（1）含义：取绝对值、小于或等于实数的最大整数、大于或等于实数的最小整数、平方根、正弦值等
（2）类型：信息类（变量型数据）
（3）范例：数值计算、云台绕圆
①数值计算

可以在 FPV 窗口观察计算结果。

②云台绕圆

利用圆心角和半径，计算控制云台绕圆形。



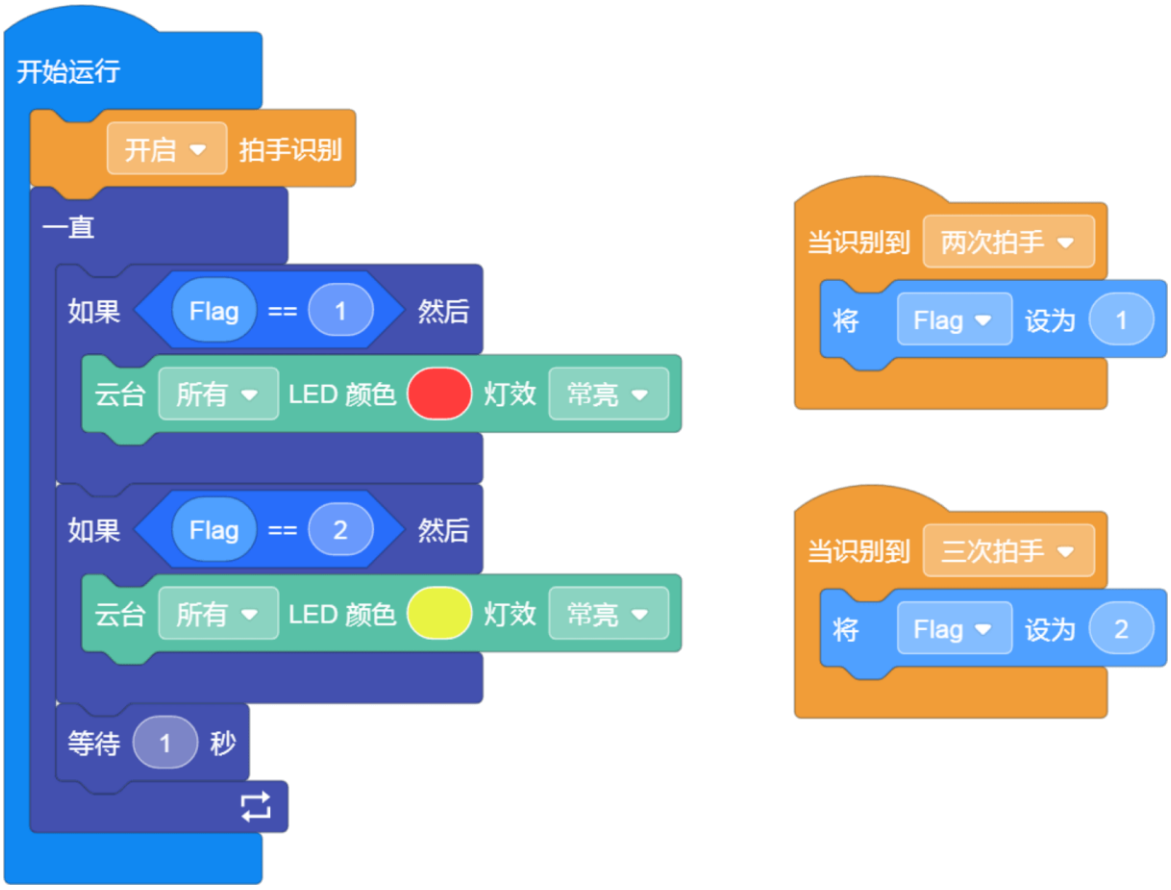
9、（0）==（0）

（1）含义：两个值相等时返回“真”，否则返回“假”

（2）返回值：布尔型

（3）范例：标志位

云台刚开始以默认色常亮。当拍两次手，云台所有 LED 红色常亮；拍三次手，云台所有 LED 切换为黄色常亮。



注意：

编程时，我们常用标志位来区分不同的状态，从而控制机器人执行不同的指令。

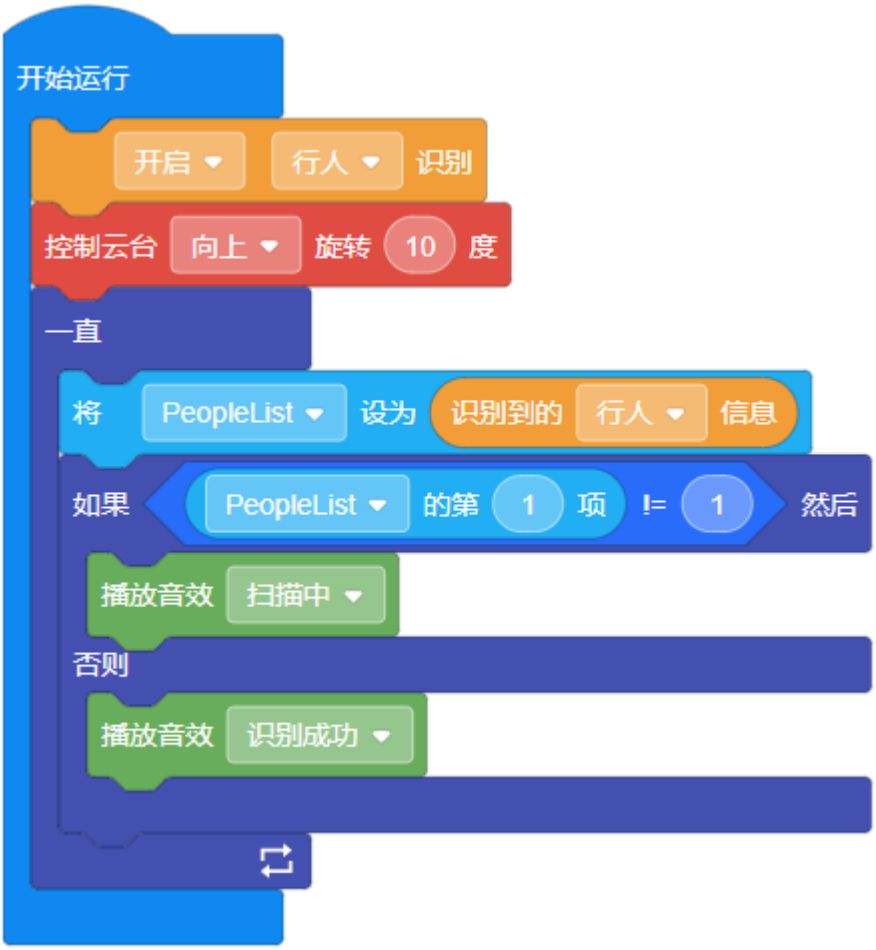
10、（0）!=（0）

（1）含义：第一个值不等于第二个值时返回“真”，否则返回“假”

（2）返回值：布尔型

（3）范例：单人识别成功

如果机器人视野中无行人或有多名行人出现，播放音效“扫描中”；如果正好有一名行人，播放“识别成功”音效。



注意：

“!=”是编程语言中的“不等于”符号，和数学中的“≠”含义一致。

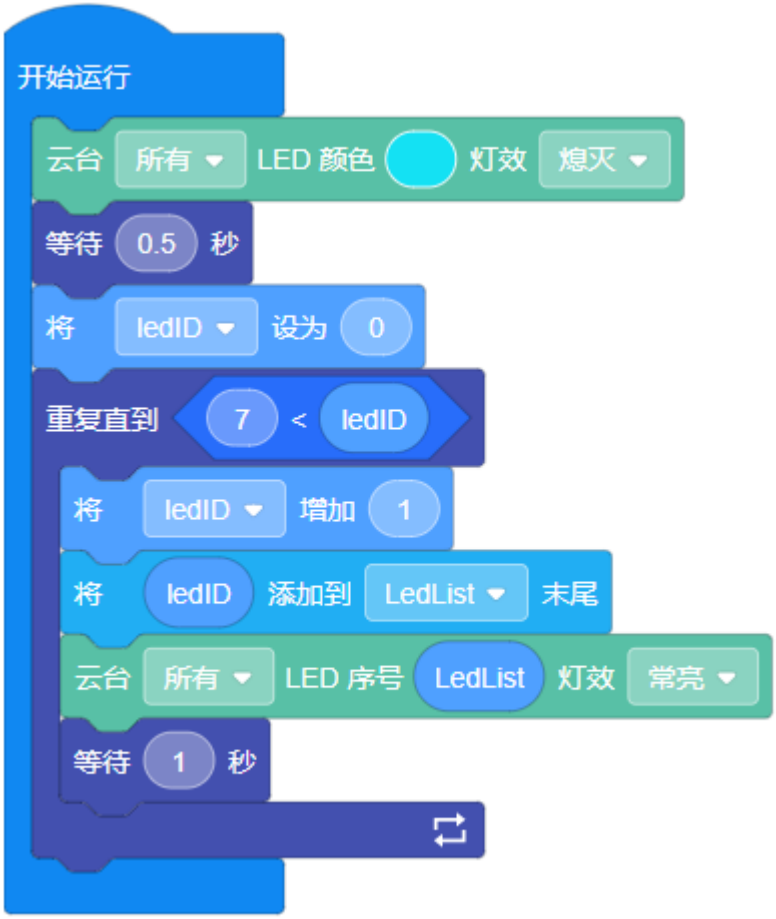
11、（0）<（0）

（1）含义：第一个值小于第二个值时返回“真”，否则返回“假”

（2）返回值：布尔型

（3）范例：依次点亮

云台所有 LED 灯按升序依次亮起。



注意：
常和条件类模块连用。

12、（0）<=（0）

（1）含义：第一个值小于或等于第二个值时返回“真”，否则返回“假”
（2）返回值：布尔型
（3）范例：加速旋转
如果底盘现在的旋转速率小于或等于 540 度/秒，那就以 60 度/秒的幅度递增，每次向右旋转 3 秒，直到达到最大值 600 度/秒后停止加速。

注意：

“＜＝”是编程语言中的“小于或等于”符号，和数学中的“≤”含义一致。

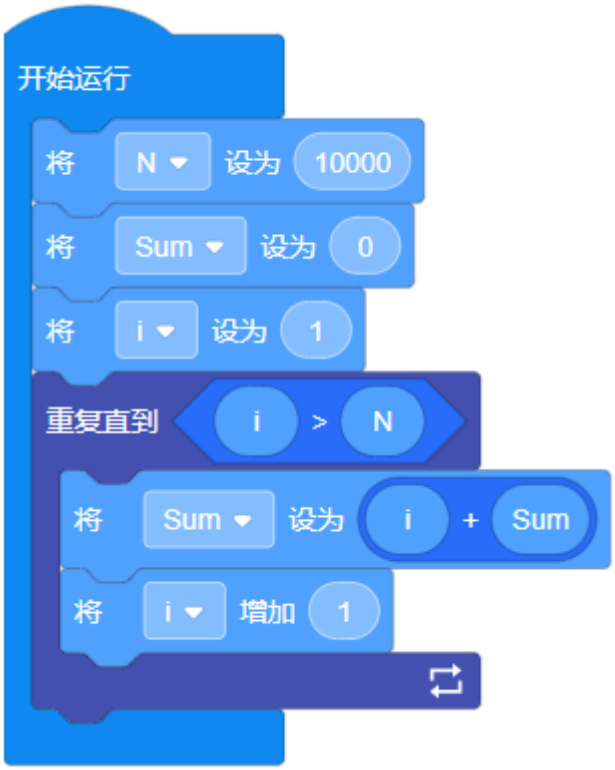
13、（0）>（0）

（1）含义：第一个值大于第二个值时返回“真”，否则返回“假”

（2）返回值：布尔型

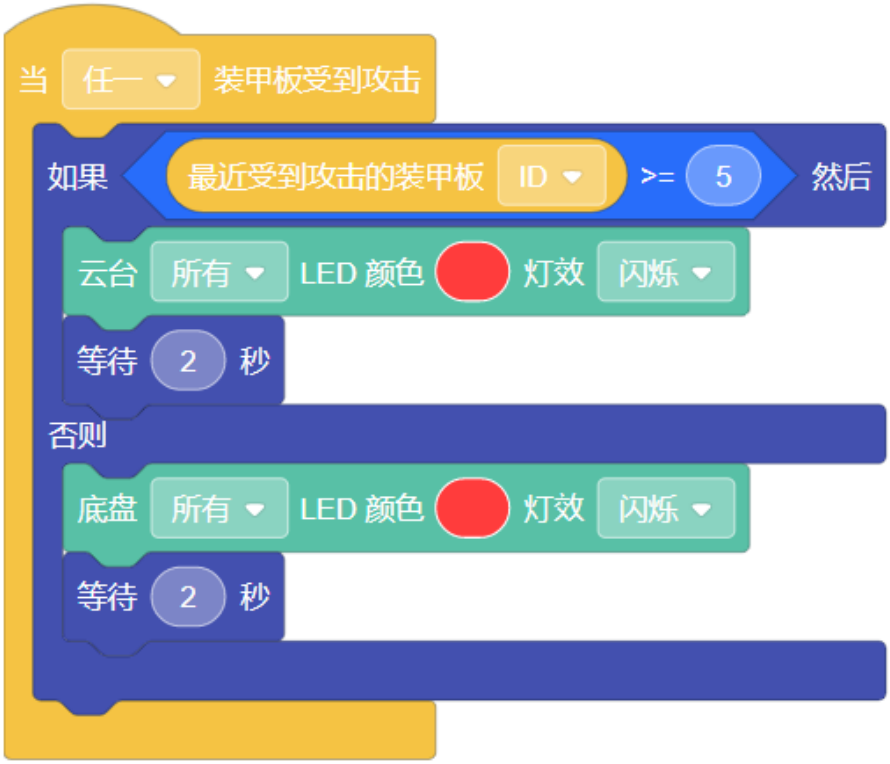
（3）范例：累加求和（1~10000）

计算 1+2+3…+10000 的值，在 FPV 窗口可以观察到 Sum=50005000。



14、（0）>=（0）

（1）含义：第一个值大于或等于第二个值时均返回“真”，否则返回“假”
（2）返回值：布尔型
（3）范例：最近受到攻击的装甲板
如果最近受到攻击的是云台的装甲板，则云台所有 LED 红光闪烁；如果是底盘的装甲板，则底盘所有 LED 红光闪烁。



注意：

“ > = ”是编程语言中的“大于或等于”符号，和数学中的“ ⩾ ”含义一致。

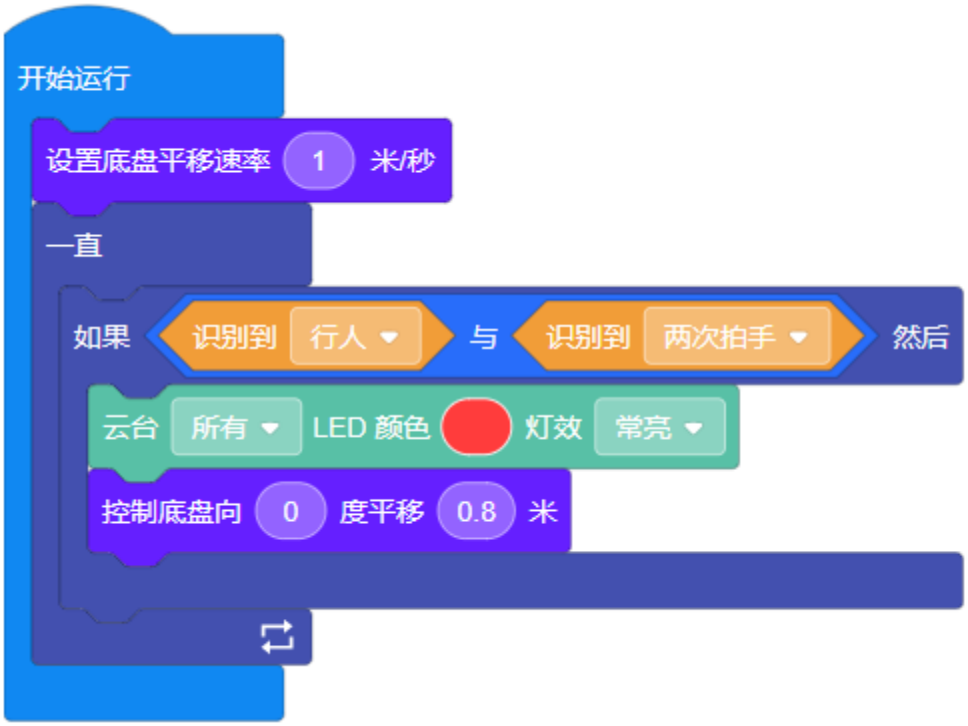
15、（ ）与（ ）

（1）含义：两个条件都成立时返回“真”，否则返回“假”

（2）返回值：布尔型

（3）范例：拍手前进

行人站在距离机器人约一米远的地方，拍手召唤机器人前来。“有行人”和“拍两次手”都必不可少，所以使用“与”运算。



注意：

“与、或、非”属于逻辑运算，返回的结果是布尔型——“真”（True）或者“假”（False）。

布尔代数真值表之“与”：

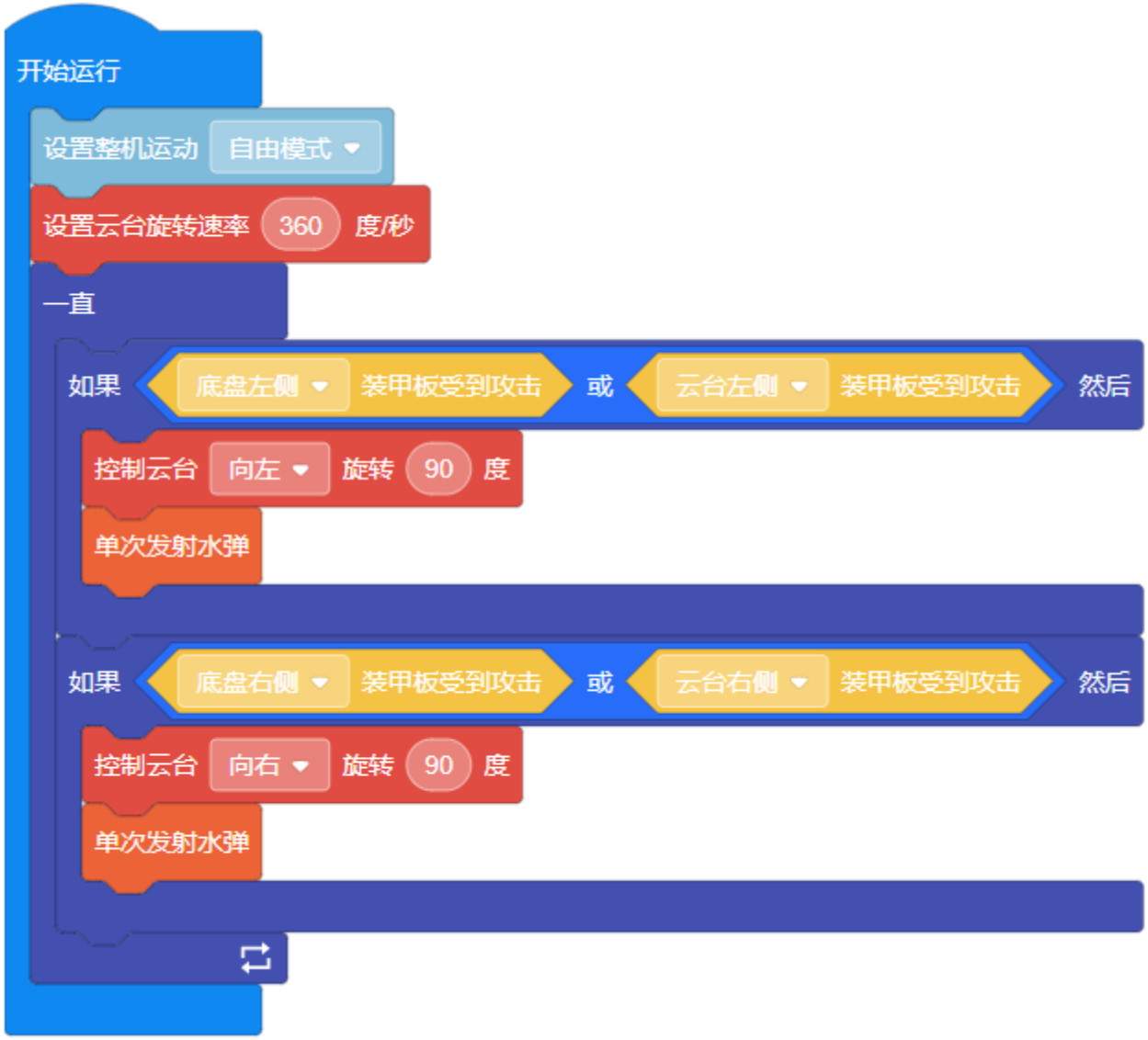
A与B		
B \ A	不成立（假）	成立（真）
不成立（假）	假	假
成立（真）	假	真

16、（ ）或（ ）

（1）含义：任一条件成立时返回“真”，两个条件都不成立时返回“假”

（2）返回值：布尔型

（3）范例：转向受攻击方向



如果左侧装甲板受到攻击，云台向左旋转反击。左侧可以是云台左侧也可以是底盘左侧，所以采用“或”运算。

注意：

“与、或、非”属于逻辑运算，返回的结果是布尔型——“真”（True）或者“假”（False）。

布尔代数真值表之“或”：

A或B		
B \ A	不成立（假）	成立（真）
不成立（假）	假	真
成立（真）	真	真

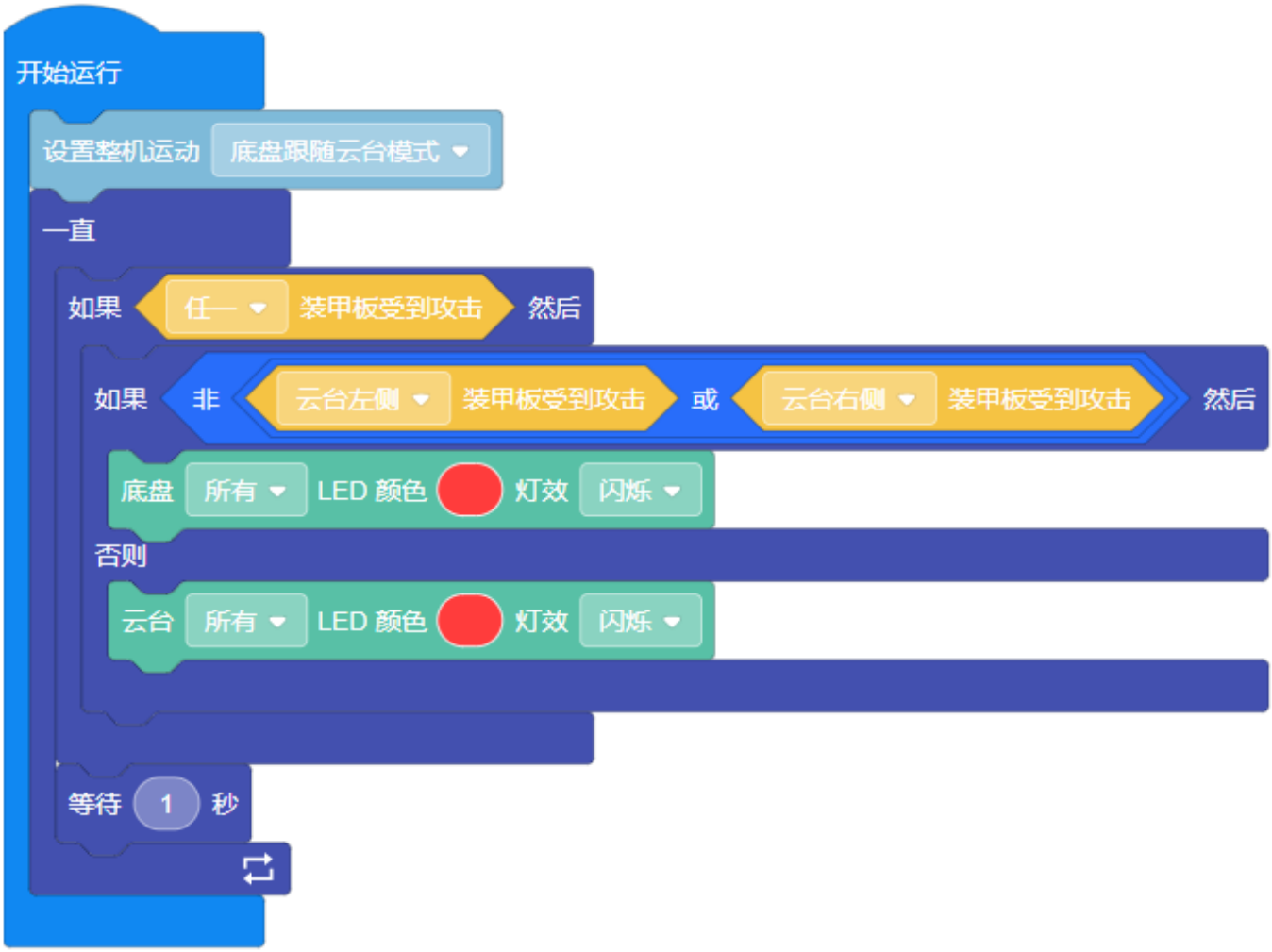
17、非（ ）

（1）含义：取反，即条件成立时返回“假”，条件不成立时返回“真”

（2）返回值：布尔型

（3）范例：非此即彼

如果云台任一装甲板受到攻击，云台所有 LED 闪烁红光；如果底盘任一装甲板受到攻击，底盘所有 LED 闪烁红光。在这里，受到攻击的装甲板“非云台侧”，说明是“底盘侧”的。



注意：

“与、或、非”属于逻辑运算，返回的结果是布尔型——“真”（True）或者“假”（False）。

布尔代数真值表之“非”：

非A		
A	不成立（假）	成立（真）
非A	真	假

18、将（0）从 低（0）高（1023）映射至 低（0）高（4）

（1）含义：根据比例关系，对设定的数值进行缩放

（2）类型：信息类

（3）范例：等比换算

传感器采集的数据实际是电压值，位数为 10 Bit，精度 1024 级，取值范围为 0 到 1023，而我们在 FPV 窗口看到的是经过映射的数值，所以通过换算，将设定范围内的数值根据比例关系缩放到另一个范围内，才能知晓真正的电压值。


开始运行
一直
将 Voltage 设为 将 1 号传感器转接模块 1 号端口ADC值 从 低 0 高 1023 映射至 低 0 高 3.3
等待 1 秒

数据对象

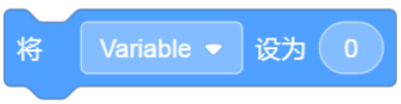
1、创建一个变量

（1）含义：创建一个变量，对变量进行命名。
（2）类型：设置类
（3）范例：新变量命名
变量名具有唯一性，最好也能够让阅读者见名知义，
所以我们常把用作标志位的变量命名为 Flag，用于存储数量的变量命名为 Number。

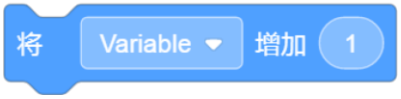
变量名需要以下划线或字母开头，只能包含数字、大小写字母和下划线。

注意：
变量创建完成后会出现 3 个相关模块，分别负责变量的调用、赋值和增减。

调用

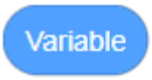


赋值



增减

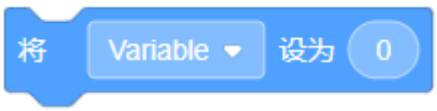
2、#变量#



（1）含义：获取变量数据
（2）类型：信息类（变量）
（3）范例：流水灯
RoboMaster EP 云台的 LED 灯全部熄灭后，流水亮起 1~8 号灯。

注意：
右键单击某变量，可以对它重命名或者删除。

3、将（#变量#）设为（0）



（1）含义：对变量进行赋值，使变量存储输入值
（2）类型：执行类
（3）范例：电子钟

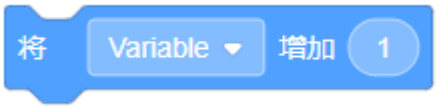
在 FPV 窗口获知当前的时间信息，时、分和秒会不断变化：

注意：

1）变量存储一个数据，列表存储一串数据，不能混用。

2）对变量来说，输入值可以是数字、变量或变量型数据；但不能是列表，不能是列表型数据。

4、将（#变量#）增加（1）



（1）含义：改变当前变量的值，正数代表增加，负数代表减小

（2）类型：执行类

（3）范例：变量值减 1、“8”字形运动

①变量值减 1

先通过赋值操作确定好变量的初始值，再设置变化值：

在 FPV 窗口，我们可以清楚地看到变量 variable_e 初始值为 2，

两秒后减小 1，所以 variable_e 变为 1。

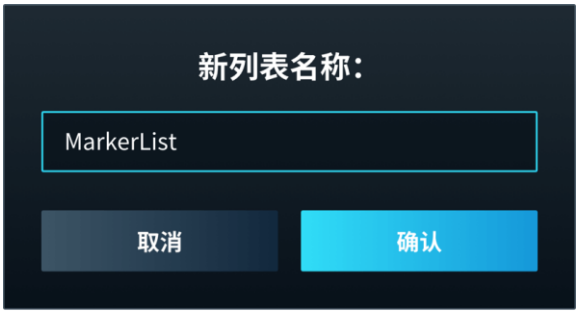
②“8”字形运动

控制机器人绕“8”字形路径运动，这是另一种实现方法。

5、创建一个列表

创建一个列表

（1）含义：创建一个列表，对列表进行命名。
（2）类型：设置类
（3）范例：新列表命名



列表名需要以下划线或字母开头，只能包含数字、大小写字母和下划线：

注意：列表创建完成后会出现多个相关模块，分别负责列表的调用、赋值、添加、删除等。

6、#列表#

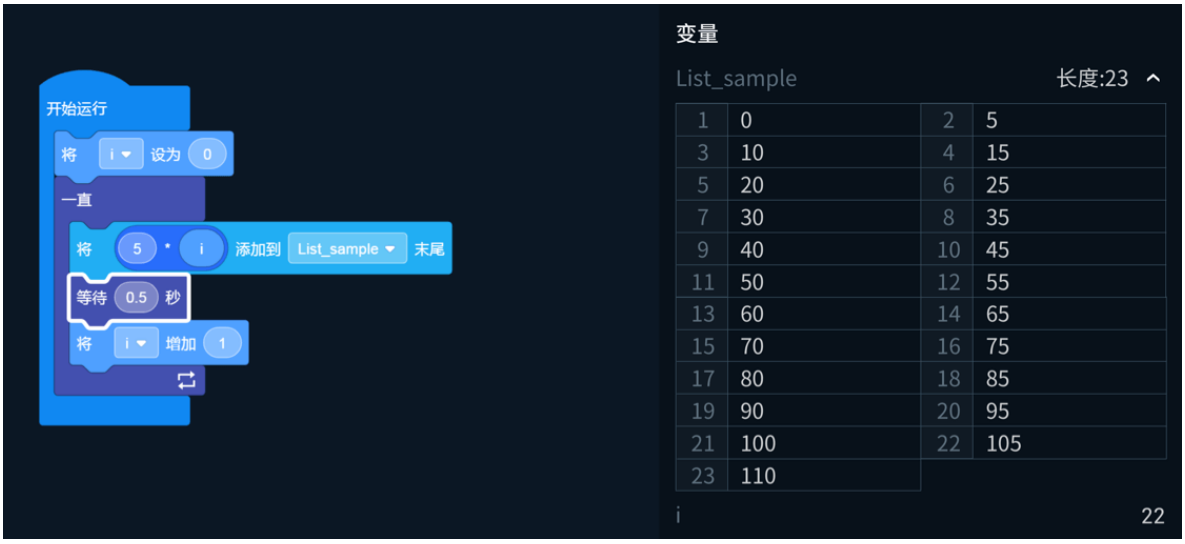
（1）含义：获取列表数据
（2）类型：信息类（列表）
（3）范例：列表复制
使新列表“CopyList”内的数据和列表“List”内的保持一致，都显示当前的整机运行时间。

可以在 FPV 窗口观察：

7、将（1）添加到（#列表#）末尾

（1）含义：将输入值添加到列表末尾
（2）类型：执行类
（3）范例：5 的倍数集
显示 5 的所有大于等于 0 的整数倍。

在 FPV 窗口观察，每隔 0.5 秒列表中就会新增出现 5 的倍数。

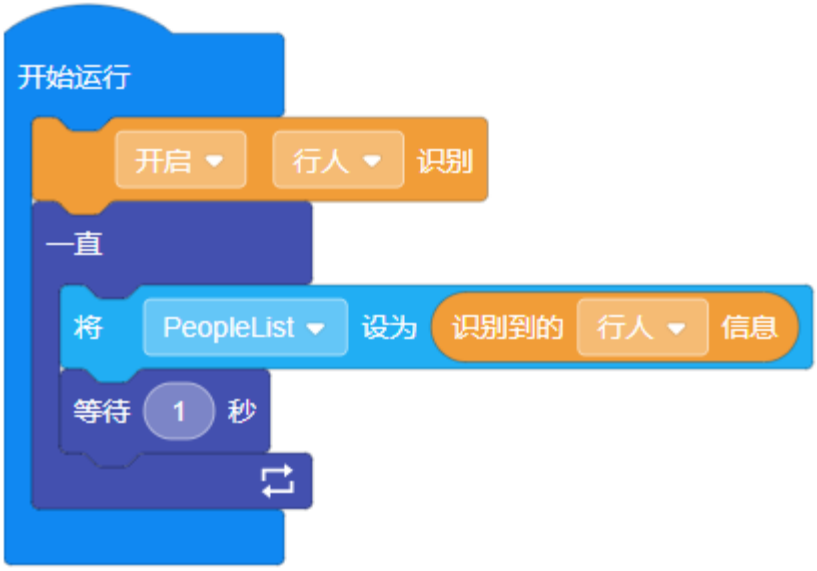


注意：

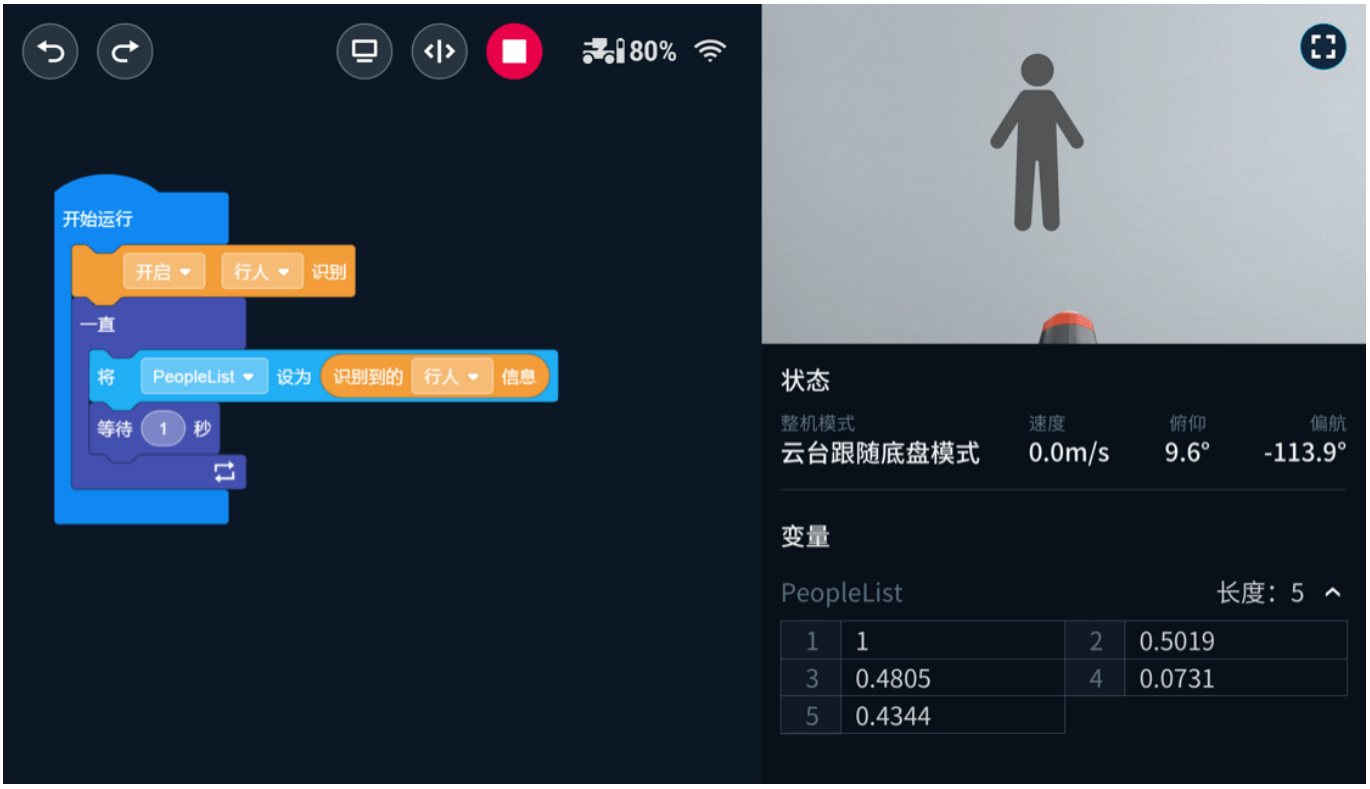
输入值可以是数字、变量、变量型数据，但不可以是列表、列表型数据。

8、将（#列表#）设为（空列表）

（1）含义：对列表进行赋值
（2）类型：执行类
（3）范例：行人识别



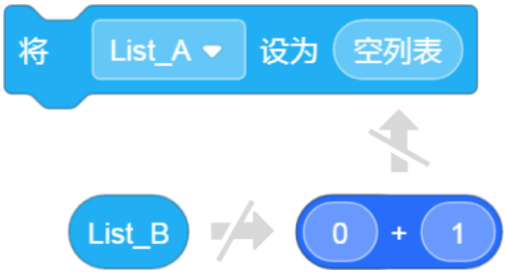
在 FPV 窗口观察识别到的行人信息：



注意：

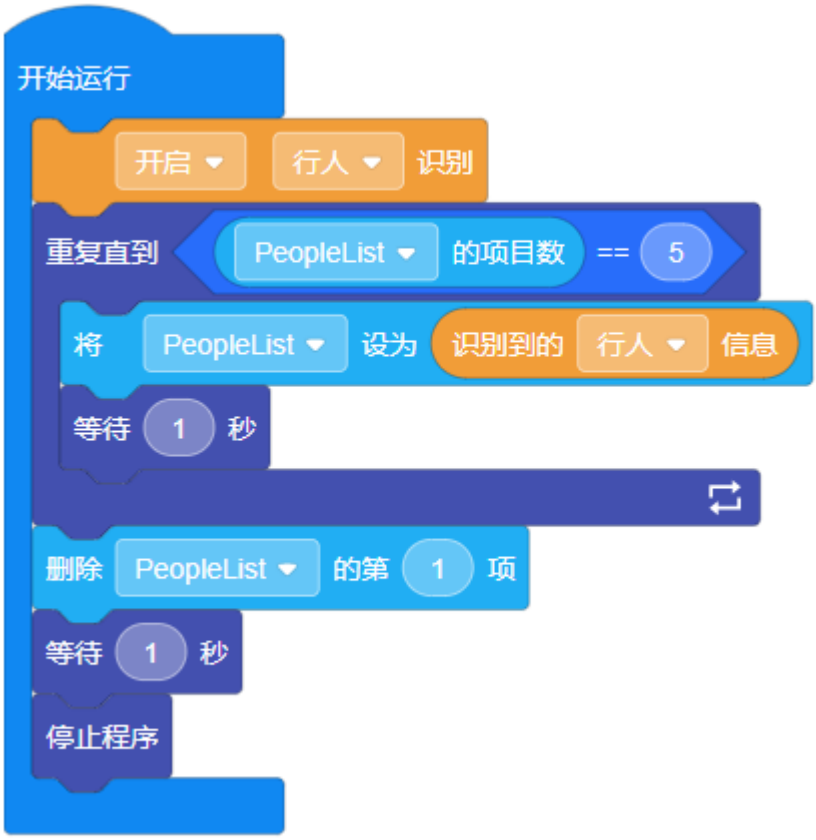
1）能给列表赋值的，可以是数字、列表或列表型数据，不能是变量或变量型数据。

2）不能对列表直接进行四则运算。



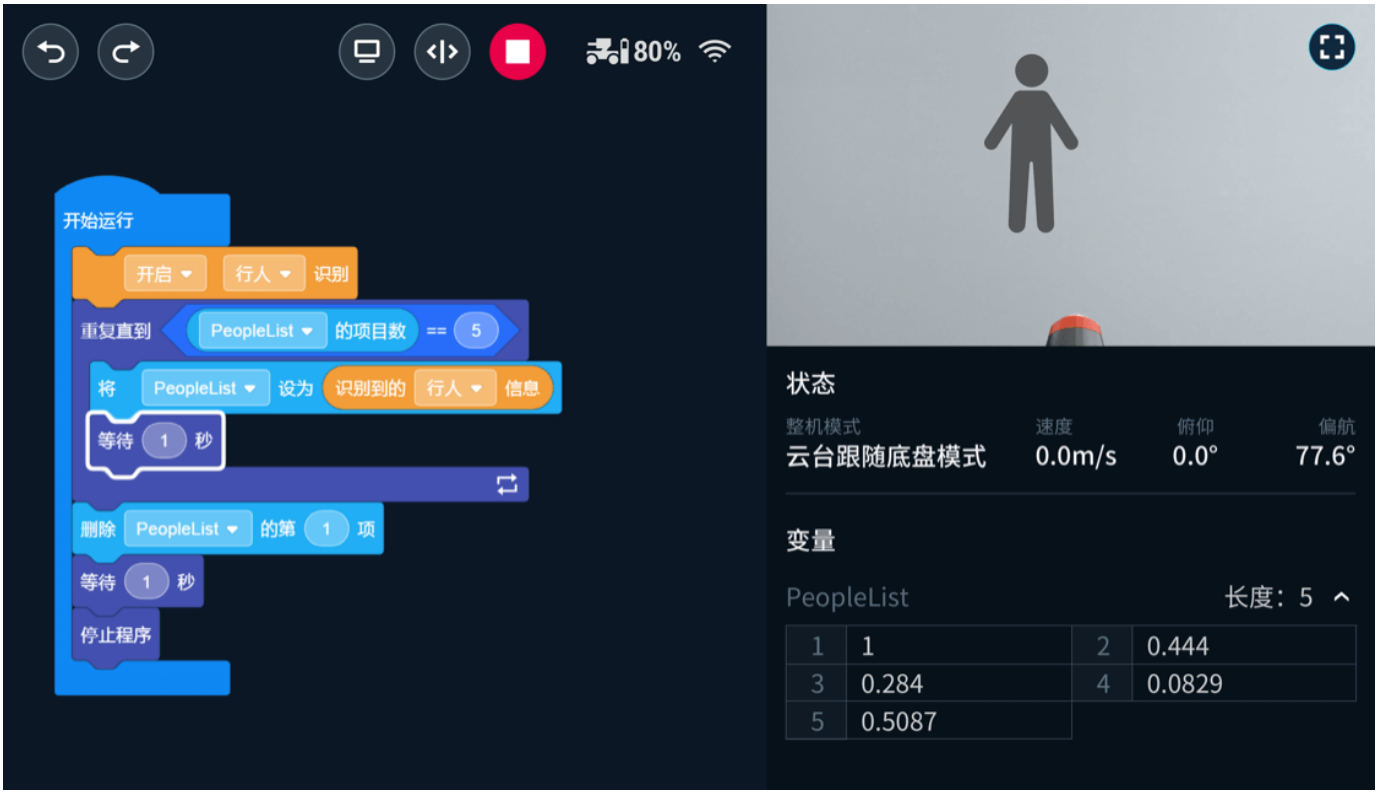
9、删除（#列表#）的第（1）项

（1）含义：删除列表中的某一项
（2）类型：执行类
（3）范例：删除行人数量



在 FPV 窗口可以清晰地看到，在删除列表中的第一项数据后，列表长度由 5 变为 4 。

删除之前：



删除之后：

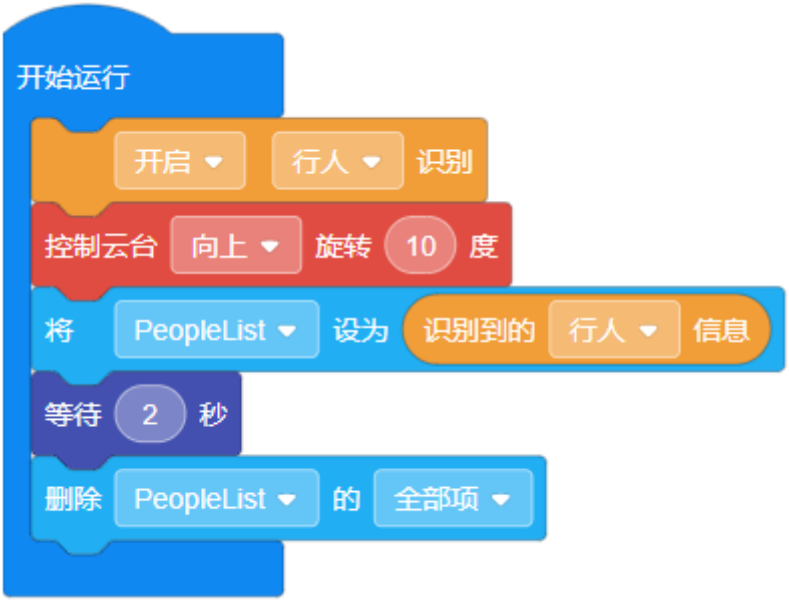
注意：

1）使用此模块的前提是明确知晓需删除项的序位数。

2）删除了某一项后，列表总体的项目数会减少 1；原后续项会全部前进一位，序位数在原有的基础上少 1。

10、删除（#列表#）的（全部）项

（1）含义：删除列表中的全部项或末尾项
（2）类型：执行类
（3）范例：列表清空



可以清楚地在 FPV 窗口观察到，列表长度由 5 变为 0。

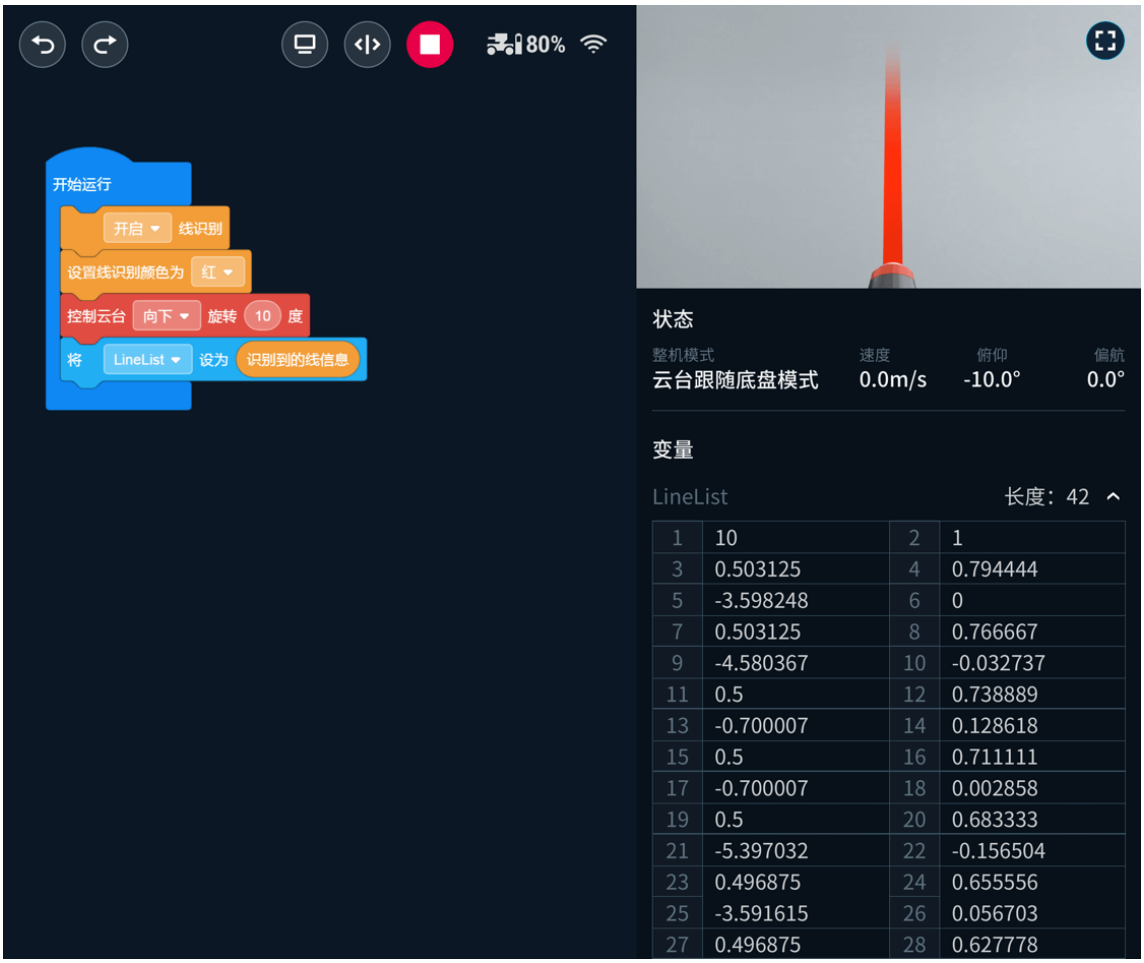
删除之前：

删除之后：

11、在（#列表#）的第（1）项前插入（1）

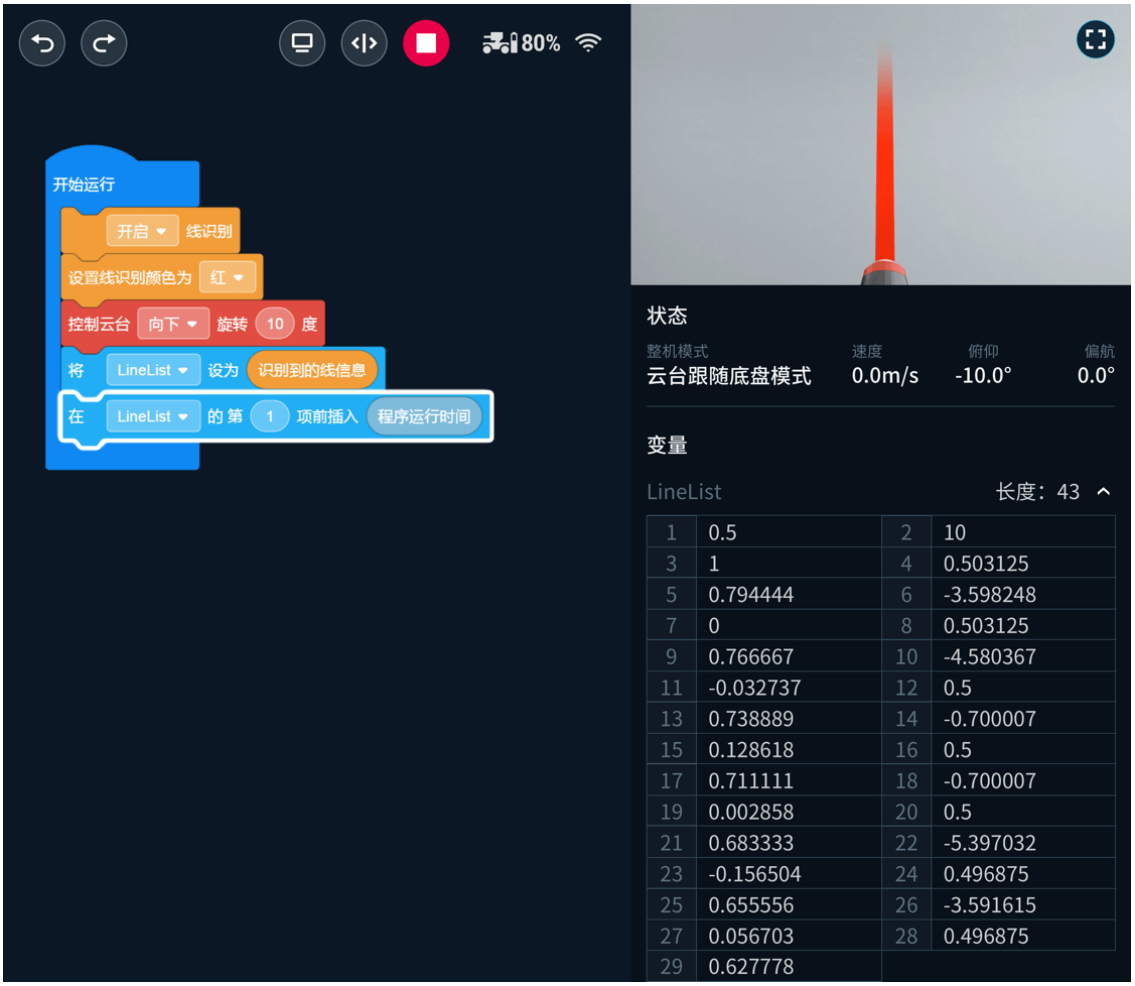
（1）含义：在列表的指定序位处添加某一项，后续项顺延
（2）类型：执行类
（3）范例：插入程序运行计时
粘好红色胶带，在 FPV 窗口观察。可以发现“程序运行时间”插入到了列表第 1 项，原后续项顺延，总项目数也增加了 1。

插入之前：



1	10	2	1
3	0.503125	4	0.794444
5	-3.598248	6	0
7	0.503125	8	0.766667
9	-4.580367	10	-0.032737
11	0.5	12	0.738889
13	-0.700007	14	0.128618
15	0.5	16	0.711111
17	-0.700007	18	0.002858
19	0.5	20	0.683333
21	-5.397032	22	-0.156504
23	0.496875	24	0.655556
25	-3.591615	26	0.056703
27	0.496875	28	0.627778

插入之后：



1	0.5	2	10
3	1	4	0.503125
5	0.794444	6	-3.598248
7	0	8	0.503125
9	0.766667	10	-4.580367
11	-0.032737	12	0.5
13	0.738889	14	-0.700007
15	0.128618	16	0.5
17	0.711111	18	-0.700007
19	0.002858	20	0.5
21	0.683333	22	-5.397032
23	-0.156504	24	0.496875
25	0.655556	26	-3.591615
27	0.056703	28	0.496875
29	0.627778		

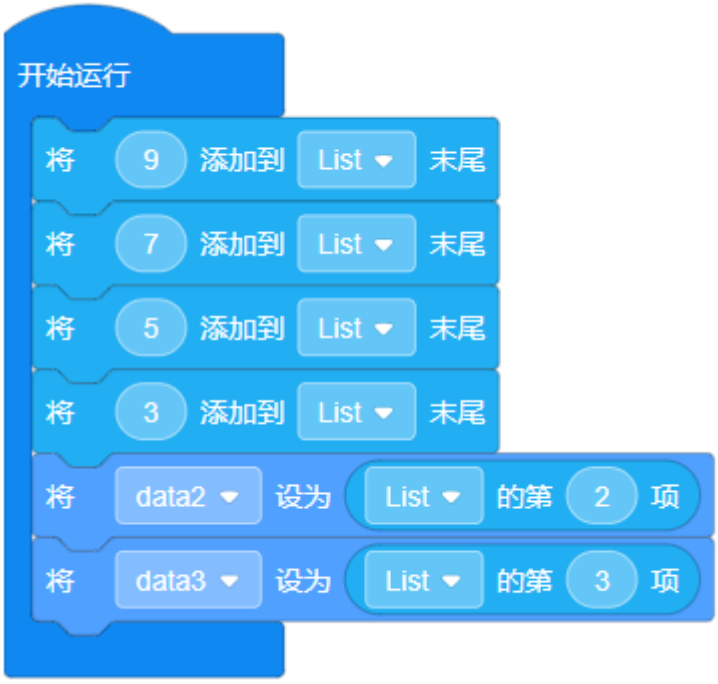
12、将（#列表#）的第（1）项替换为（1）

（1）含义：替换列表中的某一项数据
（2）类型：执行类
（3）范例：数据替换
让 List_B 列表中的数据都比 List_A 列表中的数据大 1。

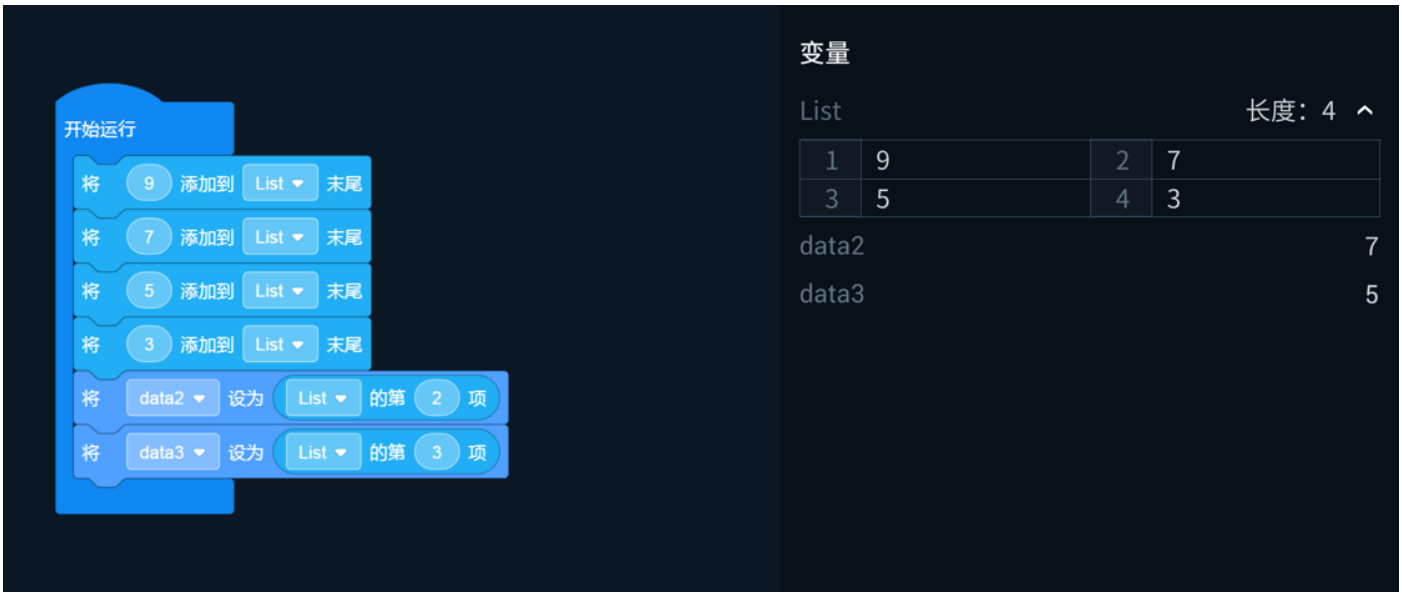
注意：
能够替换的前提是此项上原本就有数据。如果说列表 List_B 原本是空的，那么直接替换 List_B 的某一项是无效的。

13、（#列表#）的第（1）项

（1）含义：获取列表中指定项的数据
（2）类型：信息类（变量型数据）
（3）范例：列表指定项数据

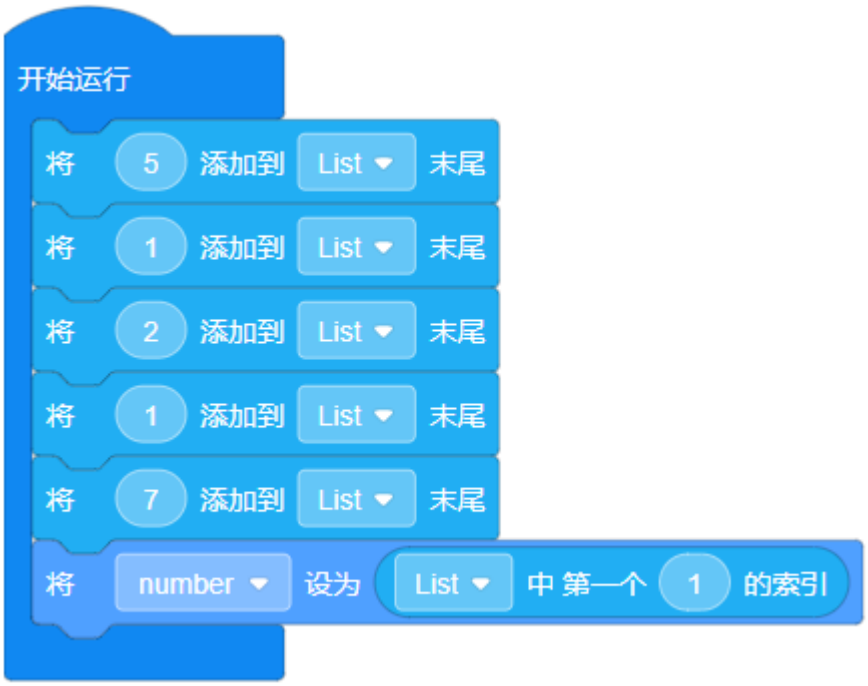


可以在 FPV 窗口观察数据值：data2=7，data3=5。



14、（#列表#）中第一个（1）的索引

（1）含义：获取列表中某数据第一次出现的位置（所在项数）
（2）类型：信息类（变量型数据）
（3）范例：读取索引值
创建列表{5,1,2,1,7}，则第一个 1 的索引 number 值为 2 。



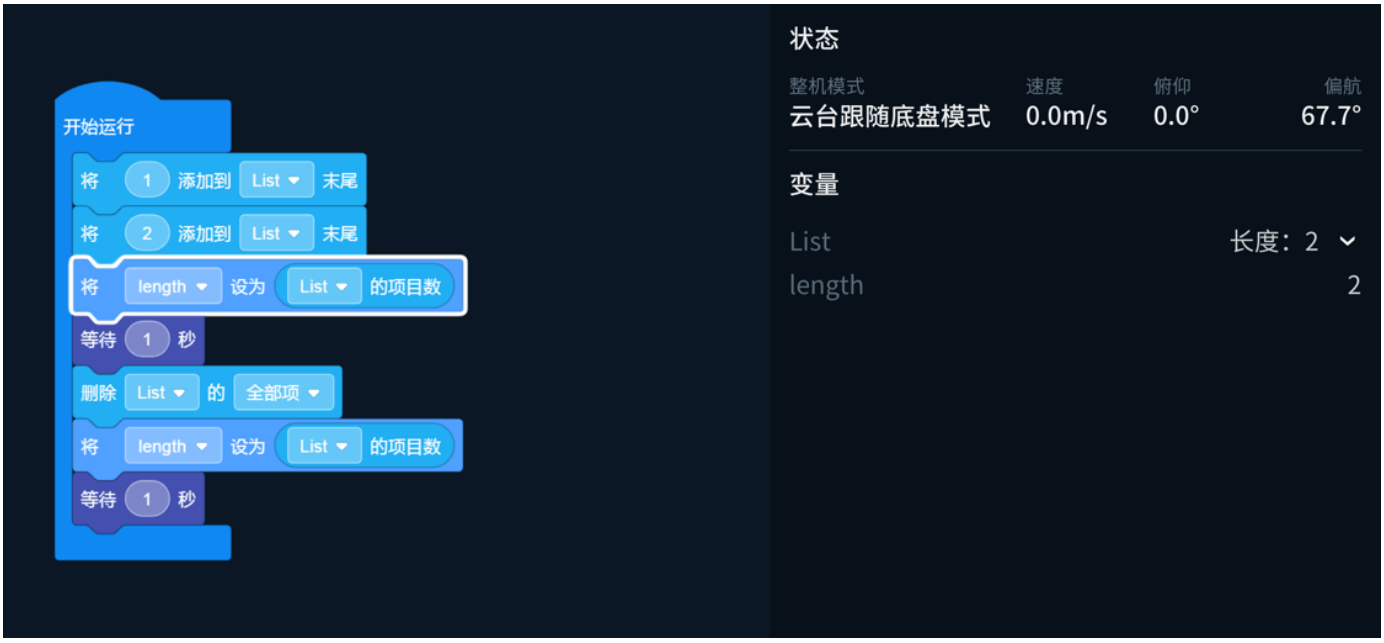
在 FPV 窗口观察：

15、（ #列表# ）的项目数

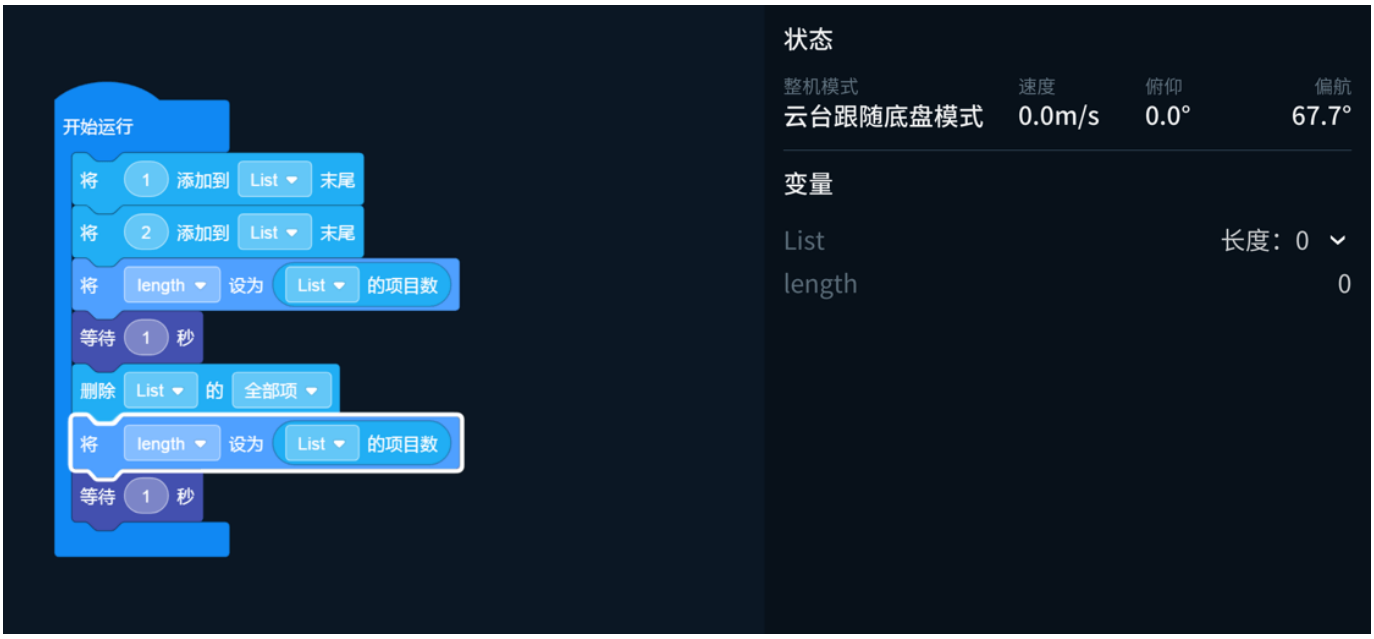
（1）含义：获知列表中有多少项
（2）类型：信息类（变量型数据）
（3）范例：计算项目数

可以在 FPV 窗口观察数值变化，length 值开始为 2，删除全部项后变为 0。

删除之前：

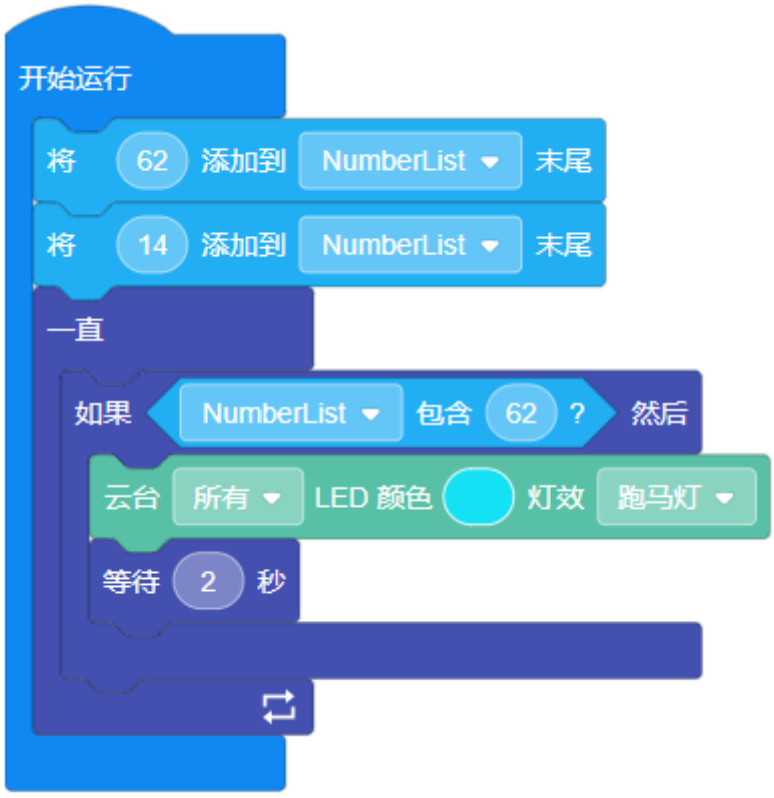


删除之后：



16、（#列表#）包含（1）？

（1）含义：列表包含输入值则返回“真”，否则返回“假”
（2）返回值：布尔型
（3）范例：列表数据判断
如果列表中包含某数值，云台所有 LED 跑马灯效示意。



注意：
判断项必须精准匹配才会判定条件成立。

17、创建 PID 控制器

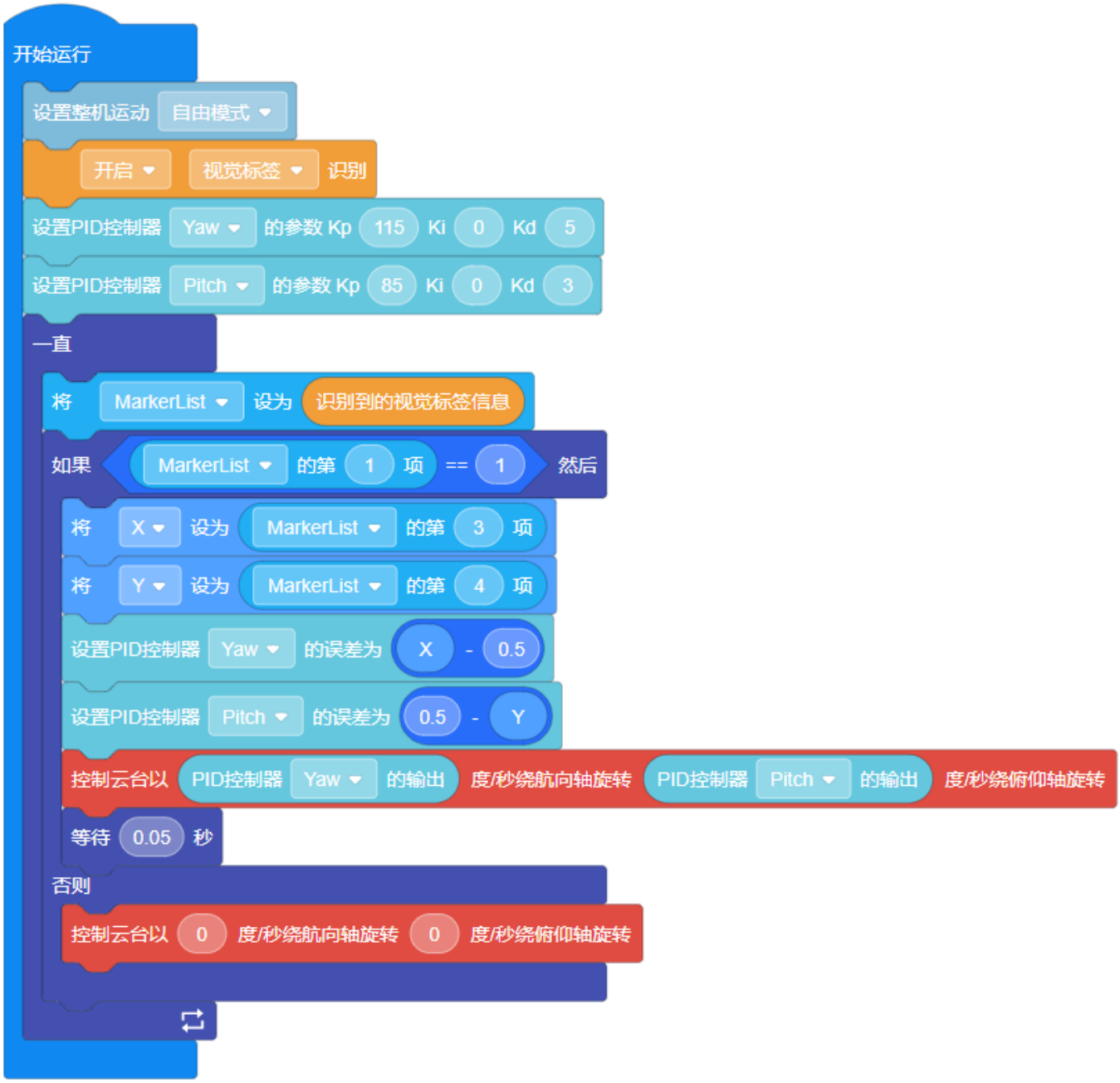
（1）含义：创建 PID 控制器，对控制器进行命名。
（2）类型：设置类
（3）范例：控制器命名

注意：控制器创建完成后会出现 3 个相关模块，分别负责误差设置、参数调节及输出信息获取。

Python API：
Class: rm_ctrl.PIDCtrl()

18、设置 PID 控制器（#PID#）的误差为（0）

（1）含义：设置 PID 控制器误差，误差指目标值与反馈值间的差值
（2）类型：设置类
（3）范例：视觉标签跟随
手持视觉标签移动，控制云台旋转跟随。
将 PID 控制器的误差设为视觉标签中心点坐标位置和机器人视野中心的差值。



注意：

运行程序前，请保证视觉标签正确地出现在机器人视野中。

	旋转	颠倒	透视	移动过快	光线不足	遮挡
情况						
影响	无影响	无影响	较大影响	较大影响	较大影响	无法识别

Python API：

Class: rm_ctrl.PIDCtrl()

Function：

- set_error(error)

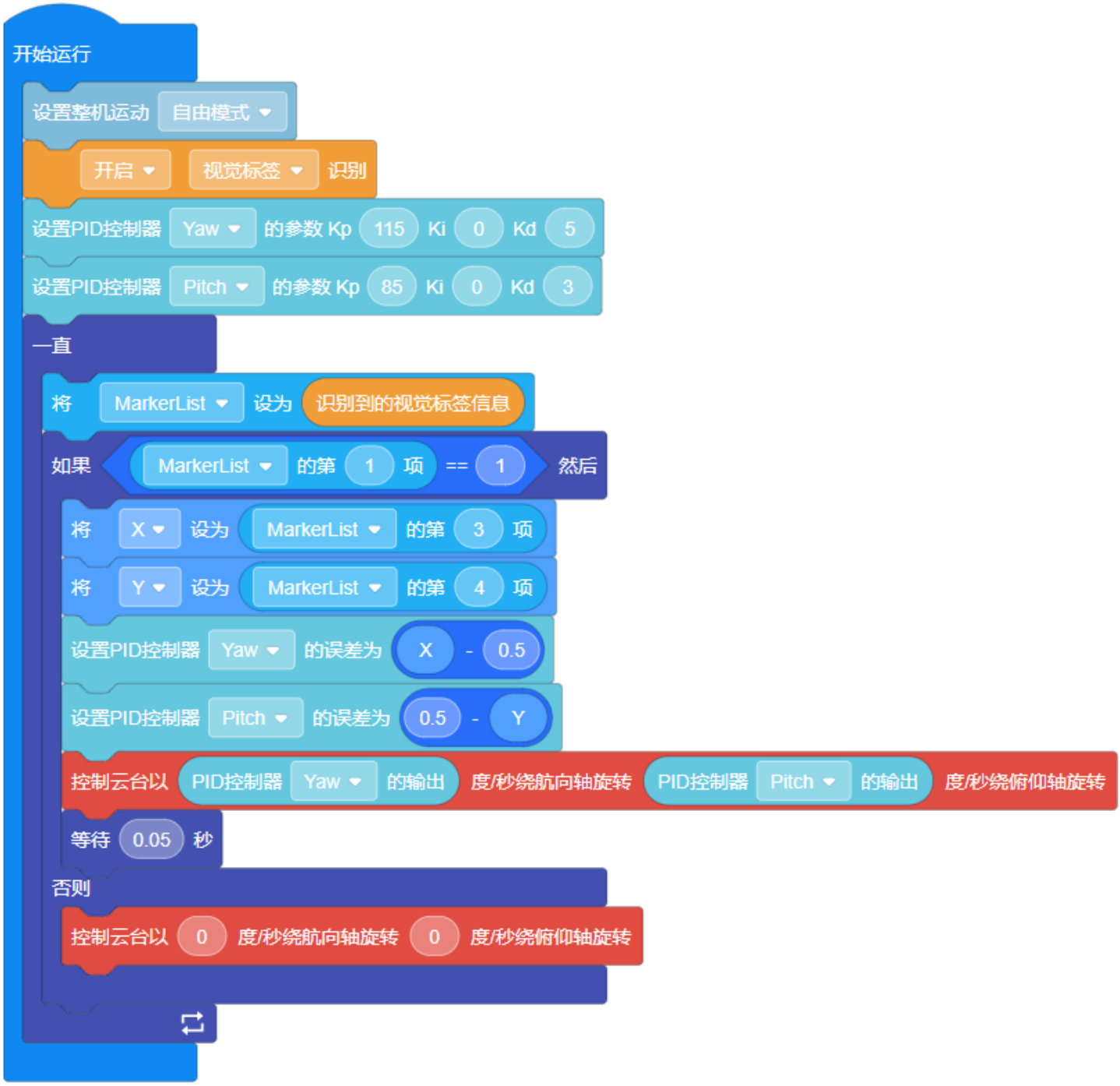
19、设置 PID 控制器（#PID#）的参数 Kp（0）Ki（0）Kd（0）

（1）含义：调节 PID 控制器参数，Kp 为比例系数，Ki 为积分系数，Kd 为微分系数

（2）类型：设置类

（3）范例：视觉标签跟随

修改 Kp、Ki、Kd 不同的参数值，使闭环控制系统达到最佳状态。



注意：

	作用	特点	缺点
比例控制（P 控制）	对误差信号进行放大或缩减。比例系数的大小决定了控制作用的强弱	比例系数越大，系统响应的快速性越好。但系数过大可能会引起系统振荡，稳定性变差	降低了系统的相对稳定性，不能消除稳态误差
积分控制（I 控制）	通过累积误差影响控制器的输出，并通过系统的负反馈作用减小误差	与误差信号存在时段有关。只要有足够的时间，积分控制便能消除稳态误差	不能及时克服干扰的影响
微分控制（D 控制）	能够反应误差信号的变化速度，在误差刚出现时就产生很大的控制量，具有超前的控制作用	有助于减小调整时间，改善系统的动态品质	不能消除系统的稳态误差

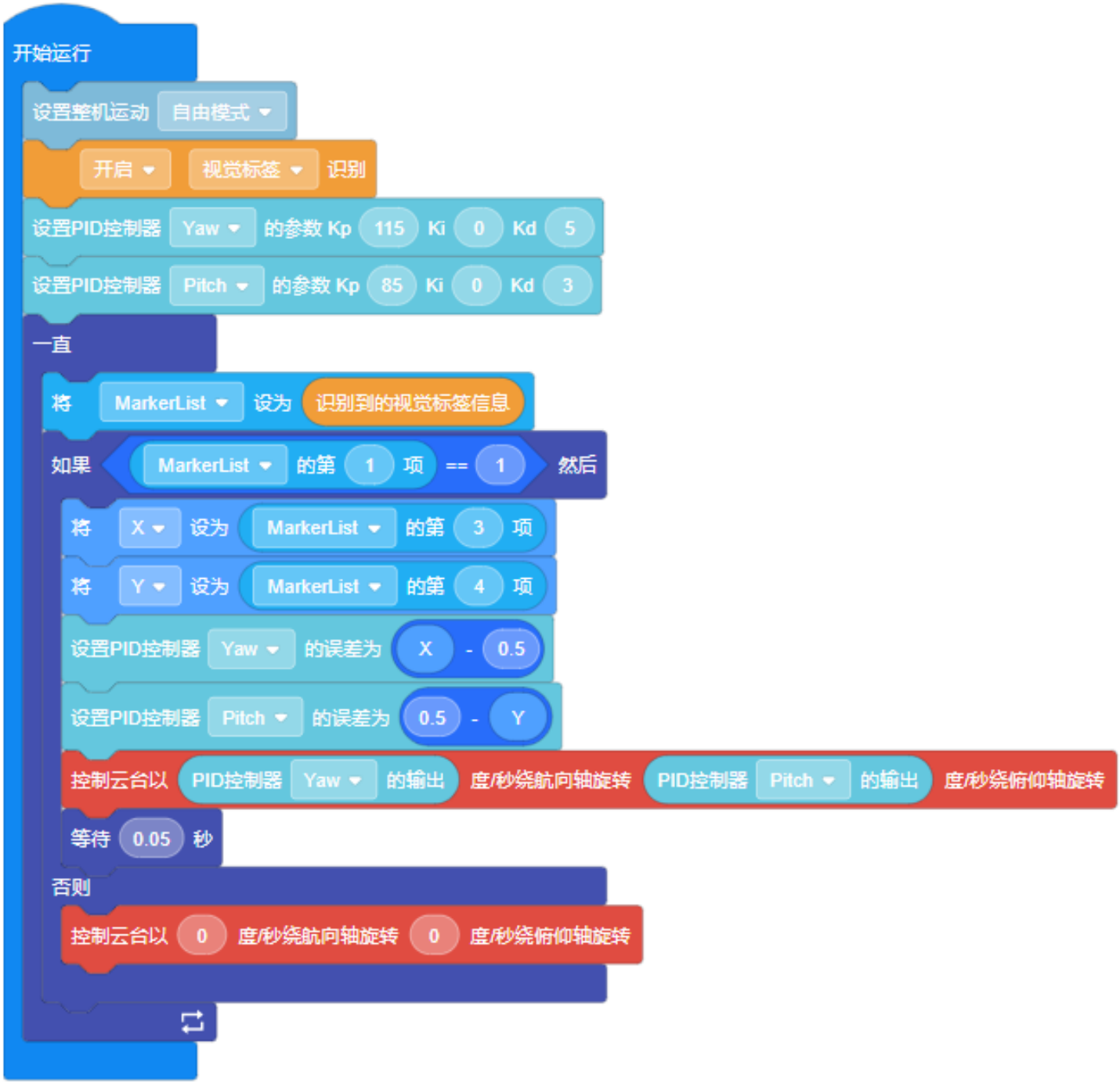
若想对 PID 内容有更加深入的了解，可以学习 RoboMaster App-大师之路中的《视觉标签跟随》项目。

Python API:
Class: rm_ctrl.PIDCtrl()

- Function:
  - set_ctrl_params(kp, ki, kd)
- Parameters:
  - kp(float)
  - ki(float)
  - kd(float)

20、PID 控制器（# PID#）的输出

（1）含义：获取指定 PID 控制器的输出值
（2）类型：信息类（变量型数据）
（3）范例：视觉标签跟随



Python API:

Class: rm_ctrl.PIDCtrl()

- Function:
  - get_output()
- Return value
  - output(float)

函数体

1、#函数#

（1）含义：把需要多次出现的程序封装成函数，方便调用。
（2）类型：函数类
（3）范例：挨打反击
如果底盘两侧装甲板受到攻击，云台会转向受攻击方向发弹反击。

注意：
1）函数名需要以下划线或字母开头，只能包含数字、大小写字母和下划线。

2）函数创建完成后会出现封装好的模块，供我们调用。

3）使用函数让整个程序逻辑更加简洁清晰。