DJI EDUCATION

ROBOMASTER EP

PROGRAMMING GUIDE

Catalog

(Click on the chapter title and jump to the corresponding page)

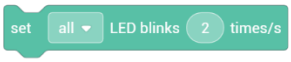
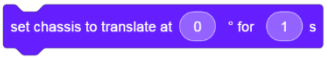
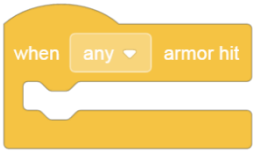
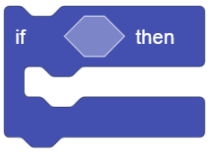
Brief Introduction

The RoboMaster EP programming guide is designed to help new users quickly learn programming techniques for controlling the EP.
The RoboMaster lab offers hundreds of graphical programming blocks that allow you to access features like PID control, computer vision, and more. While at first this may be challenging for beginners unfamiliar with robots or basic programming, this guide includes instructions and example programs for each block to help new users develop their skills. We suggest first reading through the guide to gain a basic understanding of programming. Afterward, you can refer to it for help with any questions or challenges you encounter while programming.

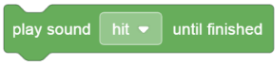
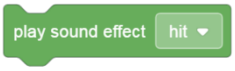
We hope you find this resource helpful to improving your programming skills, making the most of each block, and learning new ways to win.

1. block types

The RoboMaster EP programming lab offers five block types:

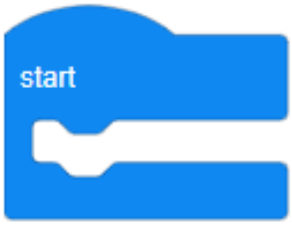
Block Type	Description	Block Example
Settings	Set parameters, such as speed, frequency, quantity and more.	
Execution	Control RoboMaster EP to execute specified commands.	
Event	Main function will pop out and begin to run programs included in event blocks when certain conditional statements are met.	
Information	Returning different types of obtained data such as variables, lists and more.	
Conditional Statement	Execute specified commands when conditional statements are met.	

2. blocking & non-blocking blocks

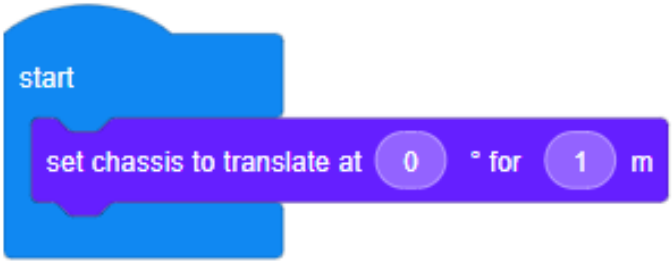
	Description	Block Example
Blcoking	Follow-up commands will not be executed before blocking blocks stop running. A "wait for xx" block is a typical blocking block.	 
Non-blocking	Follow-up commands can be executed when non-blocking blocks are running.	

System

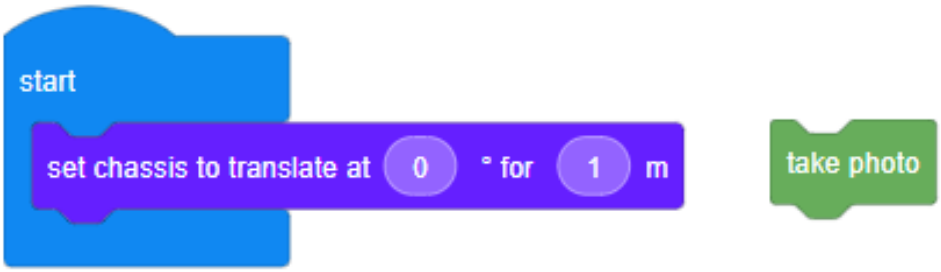
1. start



(1) Objective: Sets the first program executed when the robot powers on
(2) Type: Embedded block
(3) Example: Move forward 1 meter
This will control the EP to move forward by 1 meter.



Note:
If blocks (other than event triggering blocks and function blocks) are not placed within the “Start” area, they will not be executed. For example, in the program shown below, the EP will not be able to take a photo.

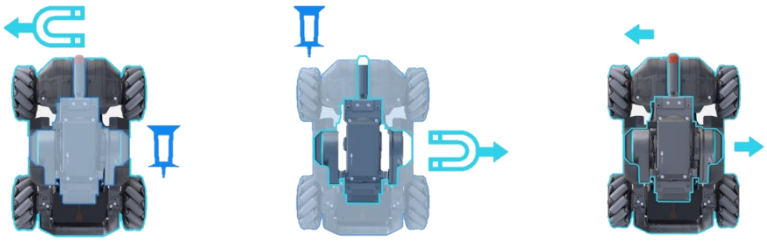


Python API:
Function: start()
Type: Main function

2. set travel mode to (chassis lead) mode

(1) Objective: Sets the travel mode

- Chassis Lead Mode: The gimbal follows the chassis to rotate along the yaw axis.
- Gimbal Lead Mode: The chassis follows the gimbal to rotate along the yaw axis.
- Free Mode: The gimbal and the chassis move without affecting each other.

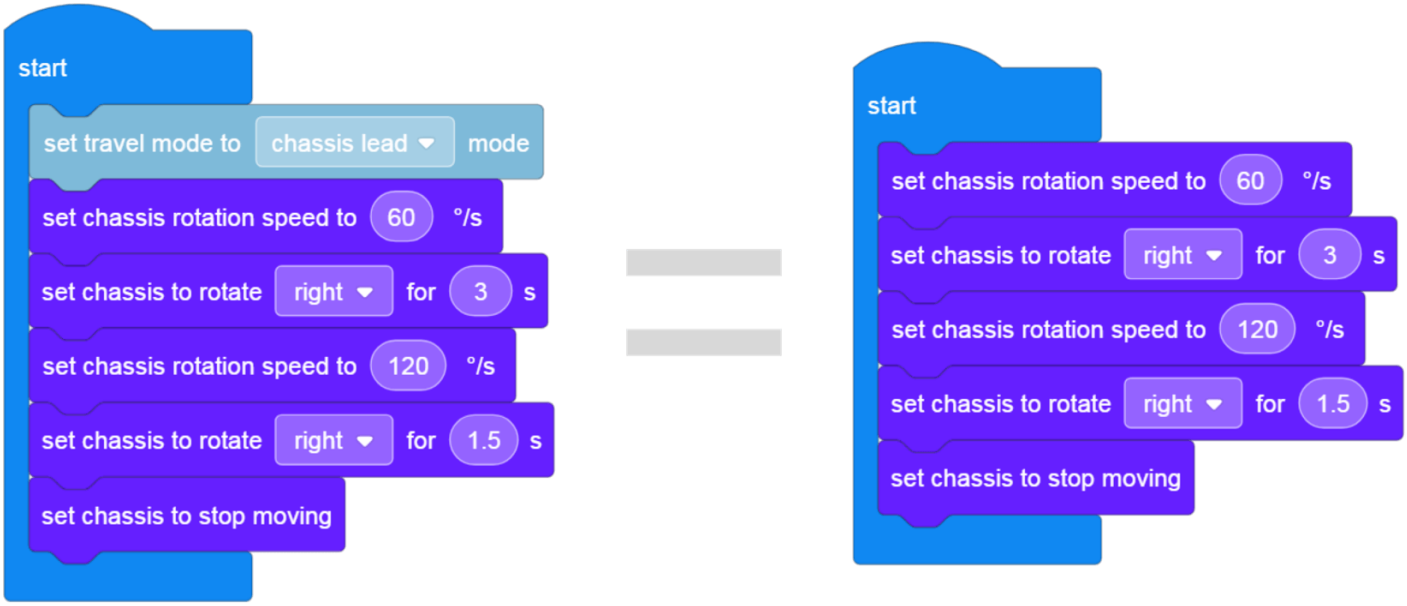


(2) Type: Settings block

(3) Examples: Rotate with variable speeds, Rotate together, Rotate contra directionally

① Rotate with variable speeds

In Chassis Lead Mode, you will only set parameters for the chassis. When the chassis is rotating to the left or right with variable rotation speeds, check that the gimbal continuously follows the chassis and that the angle between the gimbal and the chassis front is zero when chassis motion stops.



Note:

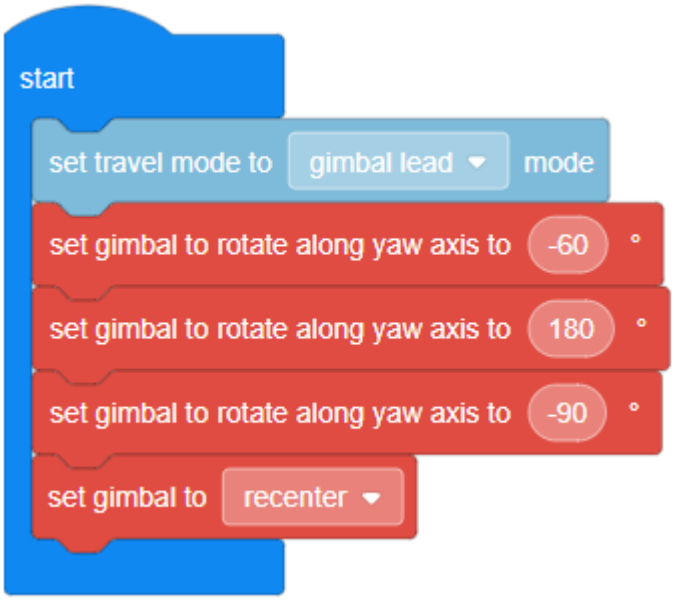
1) Chassis Lead Mode is the robot's default travel mode.

2) In Chassis Lead Mode, if the chassis is not rotating, the gimbal will not be able to rotate left or right on its own.

② Rotate together

In Gimbal Lead Mode, you will only set parameters for the gimbal.

When the gimbal is rotating along the yaw axis, check that the chassis also moves left and right and that it follows the gimbal to return to the original position.

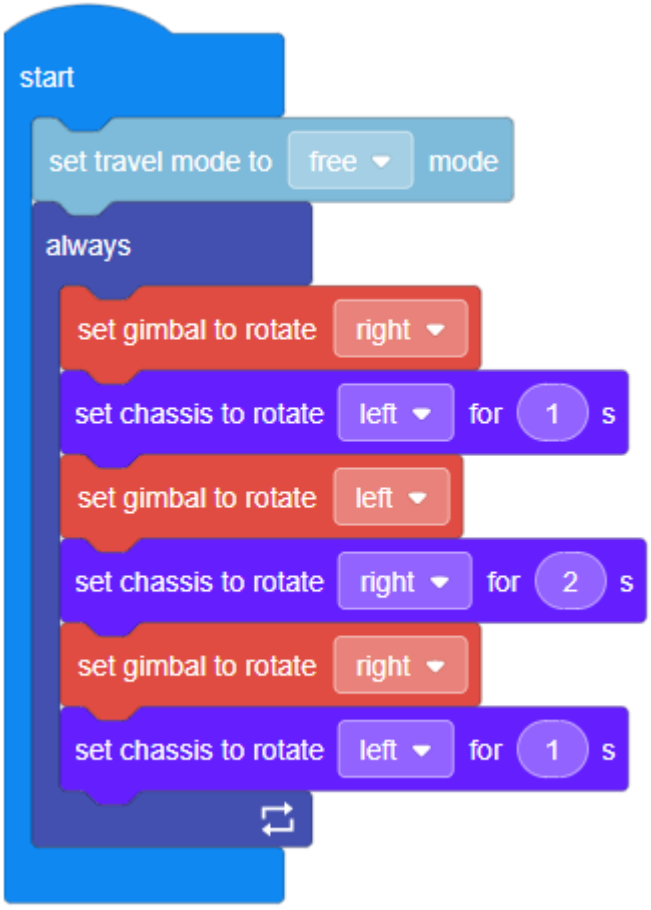


Note:

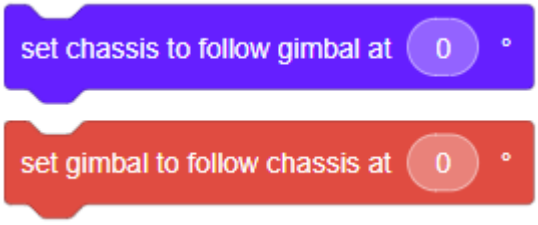
In Gimbal Lead Mode, if the gimbal is not rotating, the chassis will not be able to rotate left or right on its own.

③ Rotate contra directionally

In Free Mode, the gimbal and the chassis rotate in opposite directions. In this mode, you will need to check whether the gimbal and the chassis are interfering with each other's motion.



Note:



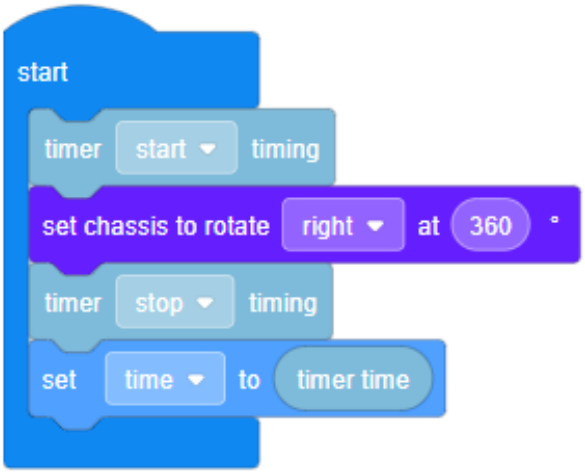
In free mode, the "set chassis to follow gimbal at (0) ° "and “set gimbal to follow chassis at (0)° ”blocks will not be able to take effect.

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

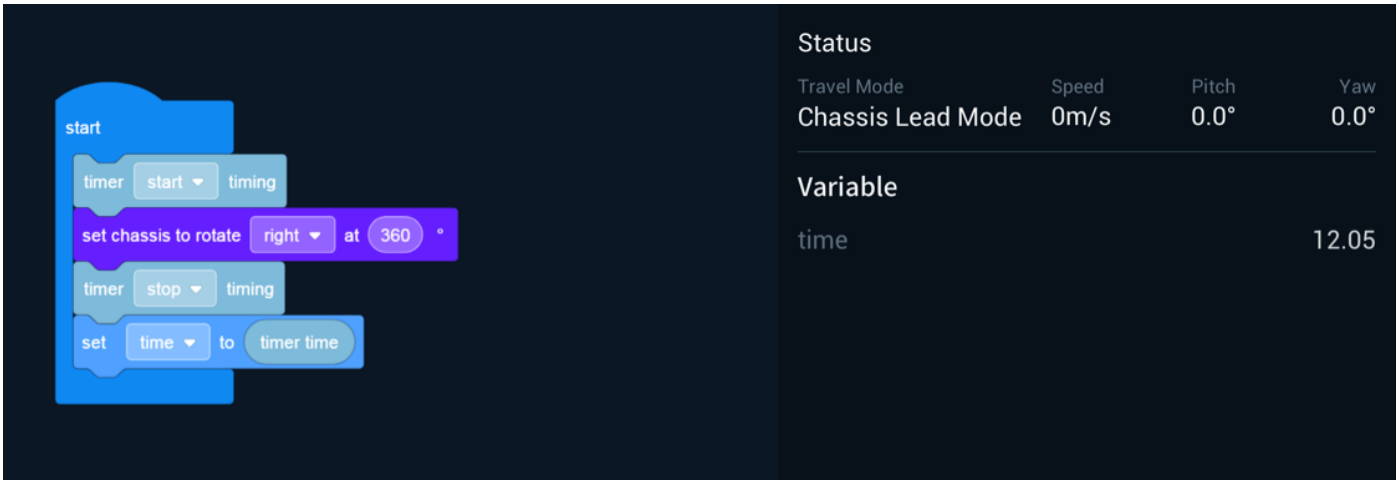
- mode_enum(enum)
  - rm_define.robot_mode_gimbal_follow
  - rm_define.robot_mode_chassis_follow
  - rm_define.robot_mode_free

3. timer (start) timing

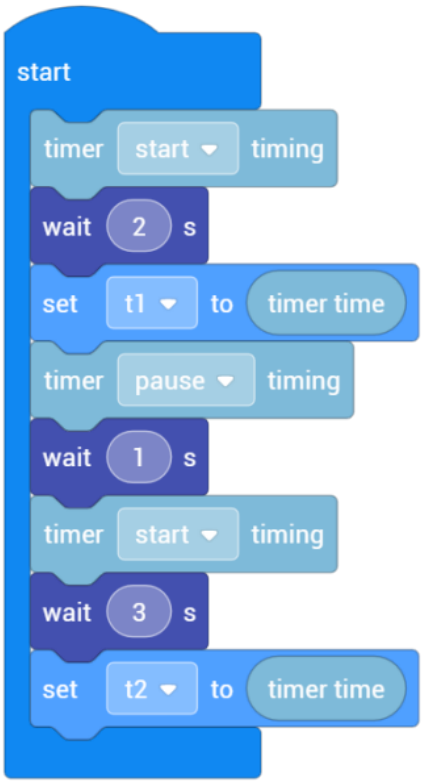
(1) Objective: Starts, pauses, or stops the timer
(2) Type: Execution block
(3) Example: Time a rotation
This measures the time it takes the chassis to complete one full rotation



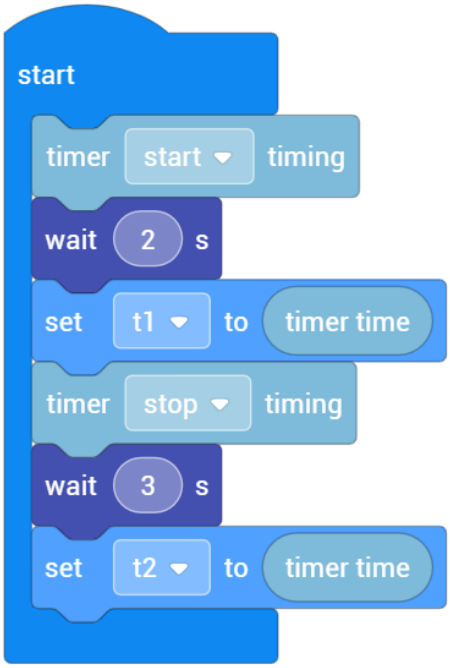
You can check details using the FPV window:



Note:
1) Selecting “Pause” will hold the time currently displayed on the timer. The timer will resume measurement from this time when it starts again.
Refer to the example below in which t1=2 and t2=5.



2) Select “Stop” to stop the timing process. The previously recorded time will be deleted, and the timer will begin measurement from zero when it starts again. You can refer to the example below in which t1=2 and t2=0.

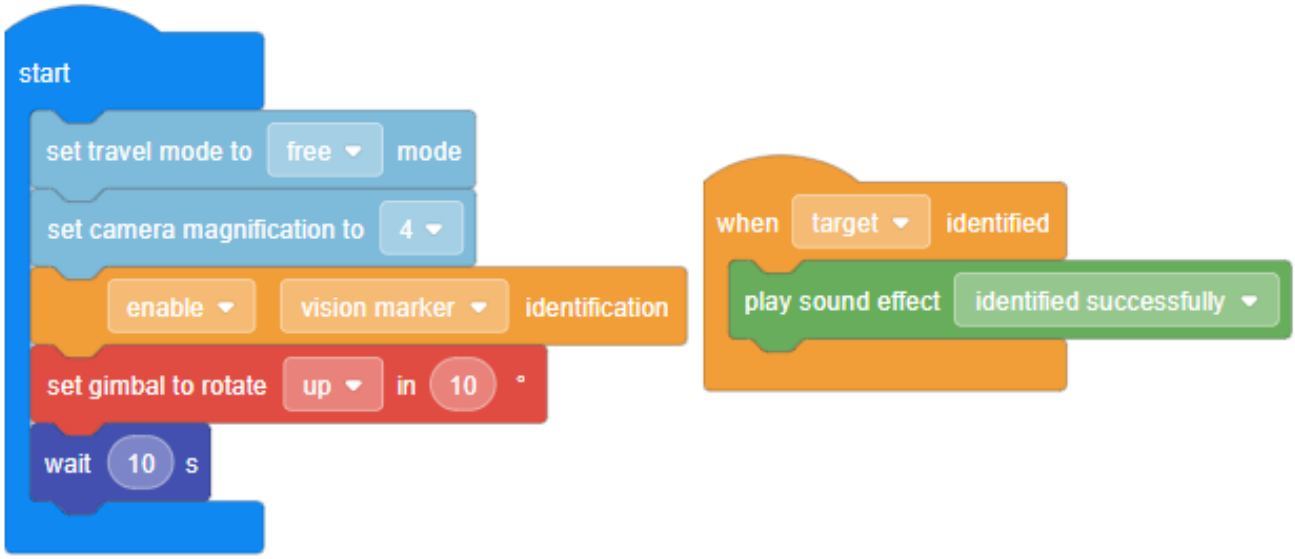


Python API:
Function: tools.timer_ctrl(behavior_enum)
Parameters:

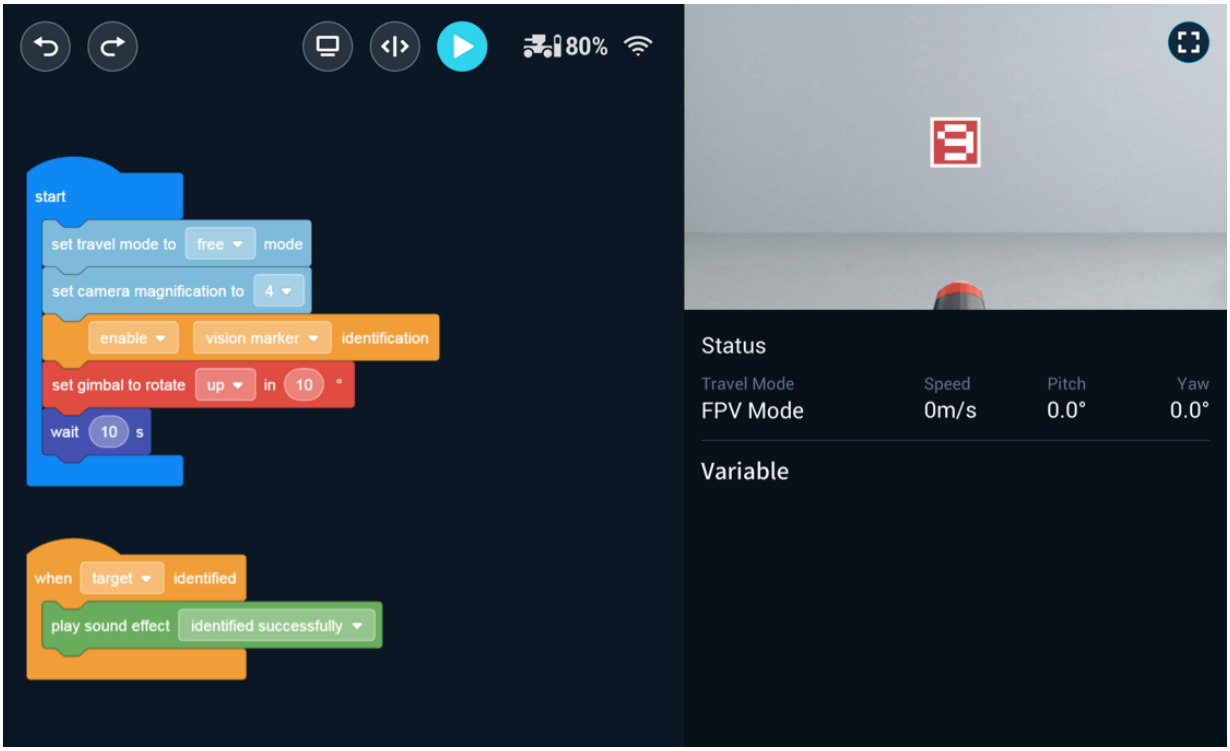
- behavior_enum(enum):
  - rm_define.timer_start
  - rm_define.timer_stop
  - rm_define.timer_reset

4. set camera magnification to (1)

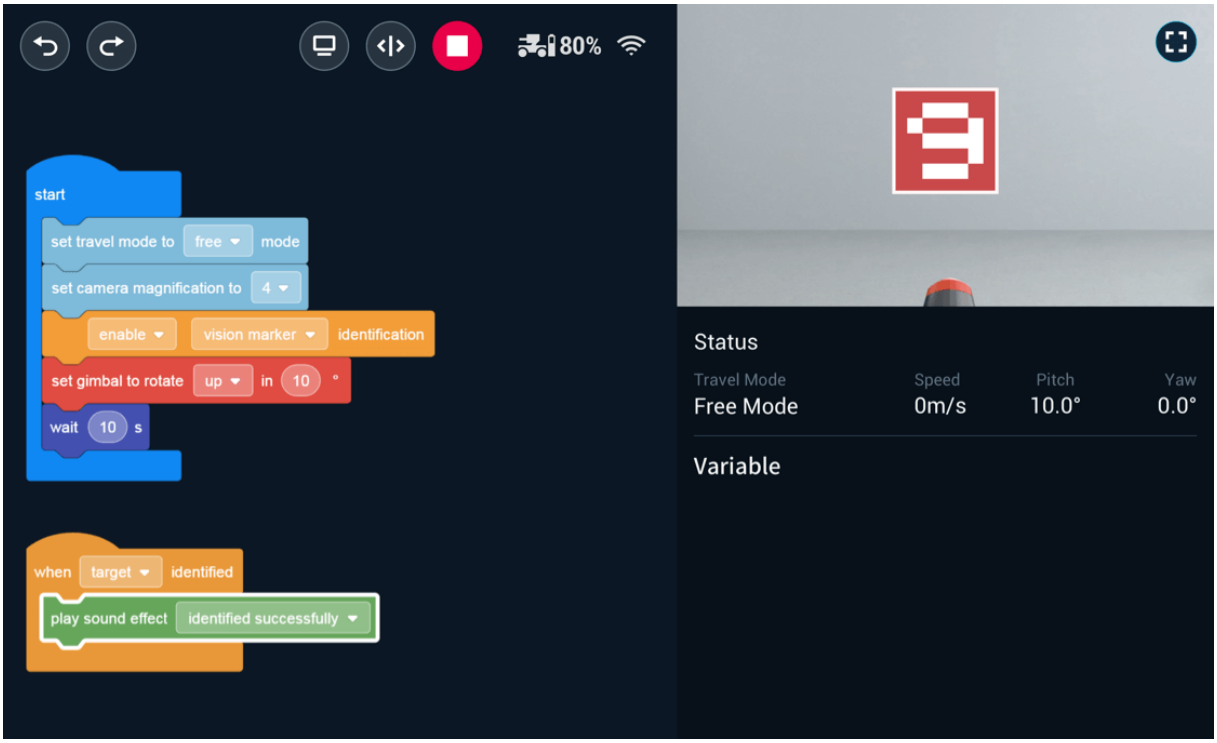
(1) Objective: Applies higher magnifications to support visual recognition over longer distances, enabling the robot to focus on unclear objects more accurately
(2) Type: Execution block
(3) Example: Enlarge camera frame
Place a Vision Marker 10 meters in front of the robot's gimbal and set the camera magnification to 4; the robot will now be able to accurately recognize the Vision Marker at this distance.



Before running the enlarge camera frame program:



After enlarging the camera frame:



Python API:

Function: media_ctrl.zoom_value_update(value)

Parameters:

- value (int): [1, 4]

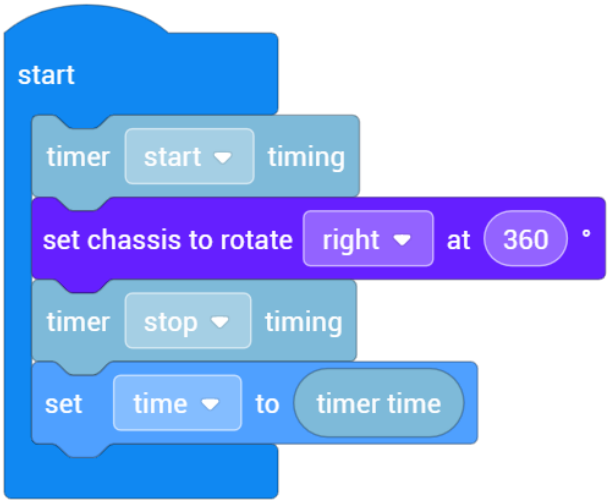
5. timer time

(1) Objective: Obtains the total time elapsed from when the timer first started to the current time (in seconds)

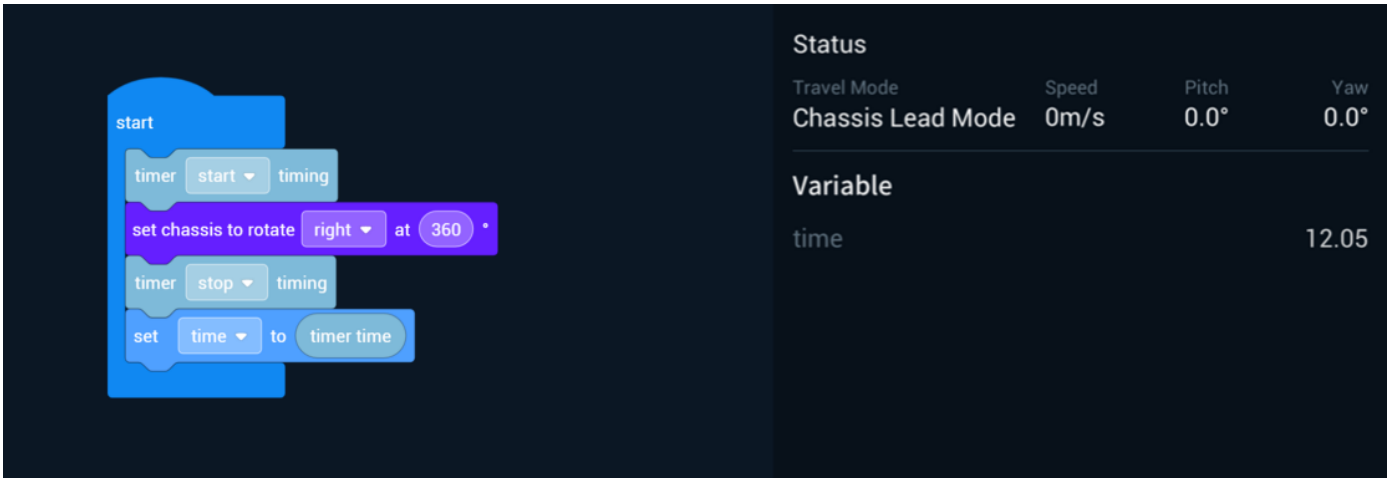
(2) Type: Information block (variable-type)

(3) Example: Time a rotation

This uses variables to measure the time it takes the chassis to complete one full rotation.



You can check details using the FPV window:



Python API:

Function: tools.timer_current()

Return value:

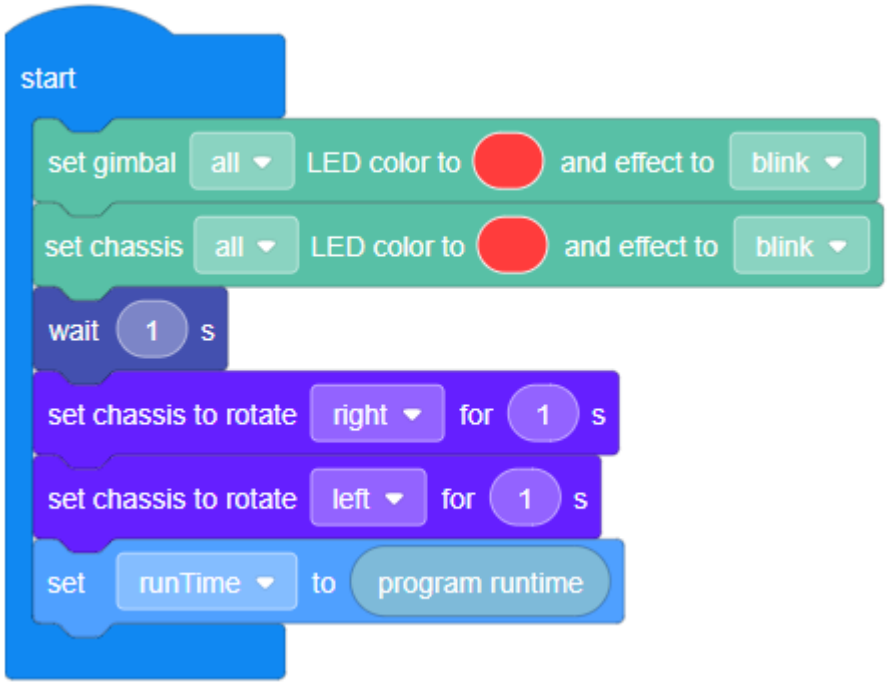
- time_stamp(float)

6. program runtime

(1) Objective: Obtains the program running time (in seconds)

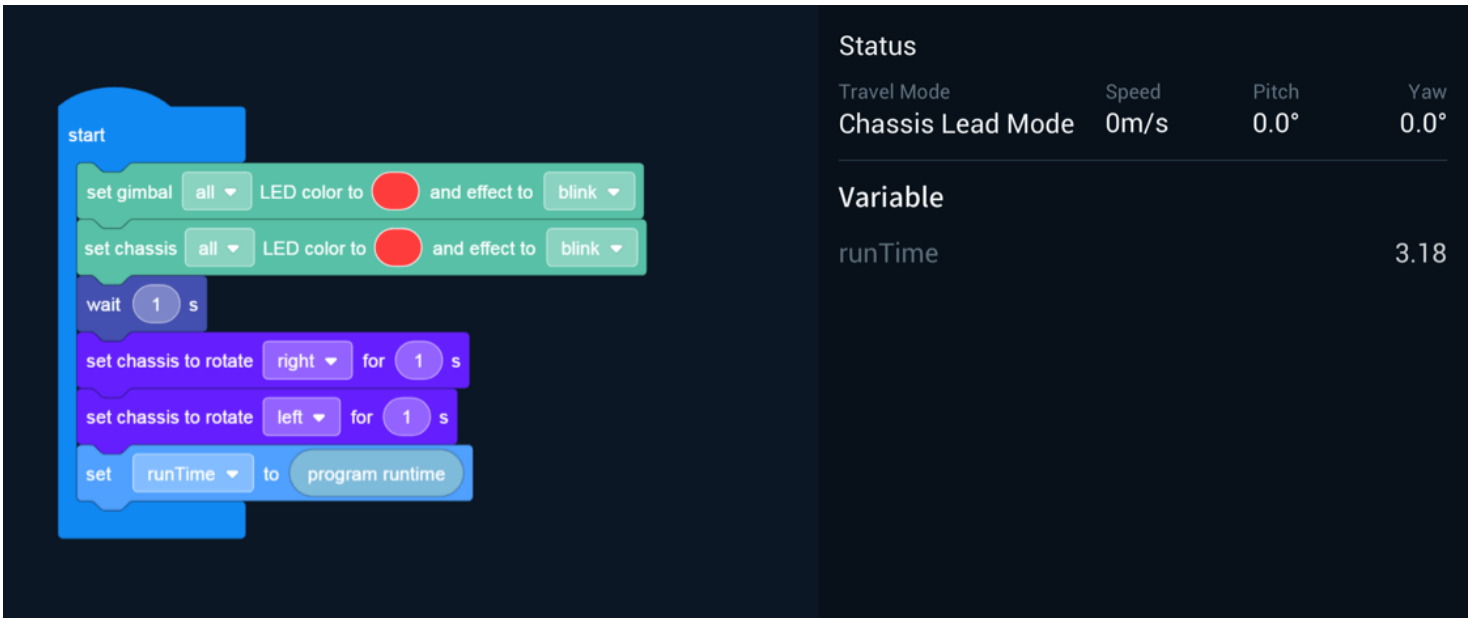
(2) Type: Information block (variable-type)

(3) Example: Calculate program runtime



This obtains the program running time using variables.

You can check specific details using the FPV window:



Python API:

Function: tools.run_time_of_program()

Return value:

- time (float)

7. current (year)

(1) Objective: Acquires current time information including the year, month, day, hour, minute, and second

(2) Type: Information block (variable-type)

(3) Example: Compare time values

If the number of the current month is larger than the number of the current day, the robot will nod its head; if the number of the current month is less than or equal to the number of the current day, the robot will shake its head.

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)

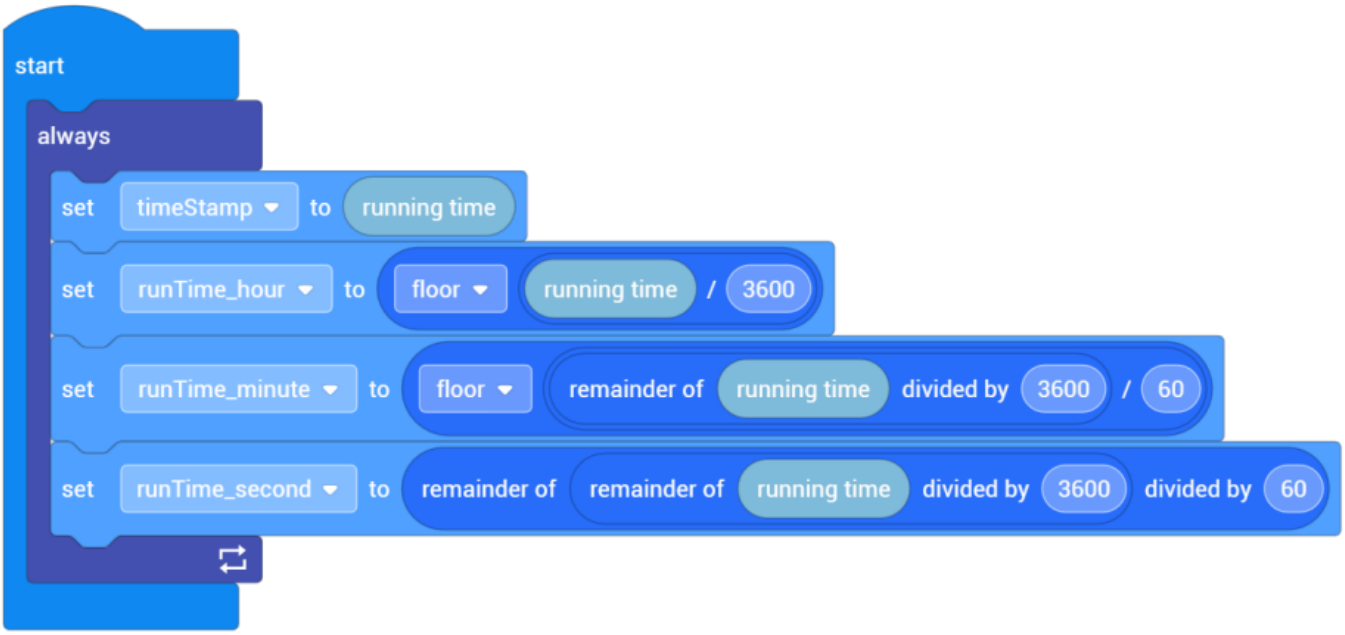
8. running time

(1) Objective: Indicates the total time elapsed from when the robot started running up to the current time (in seconds)

(2) Type: Information block (variable-type)

(3) Example: Calculate running time

This measures the total time elapsed from when the robot most recently started running up to the current hour, minute, and second.



You can check data changes using the FPV window.
The robot will need to take a break when program runtime reaches one hour.
(i.e., when runTime_hour > 1).

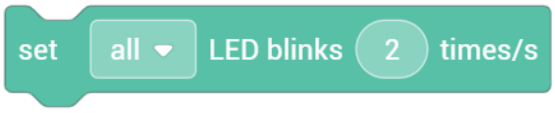
Note:
1) The start time refers to the time when the robot is powered on.
2) If the robot restarts after a power failure, it will recount the running time.

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

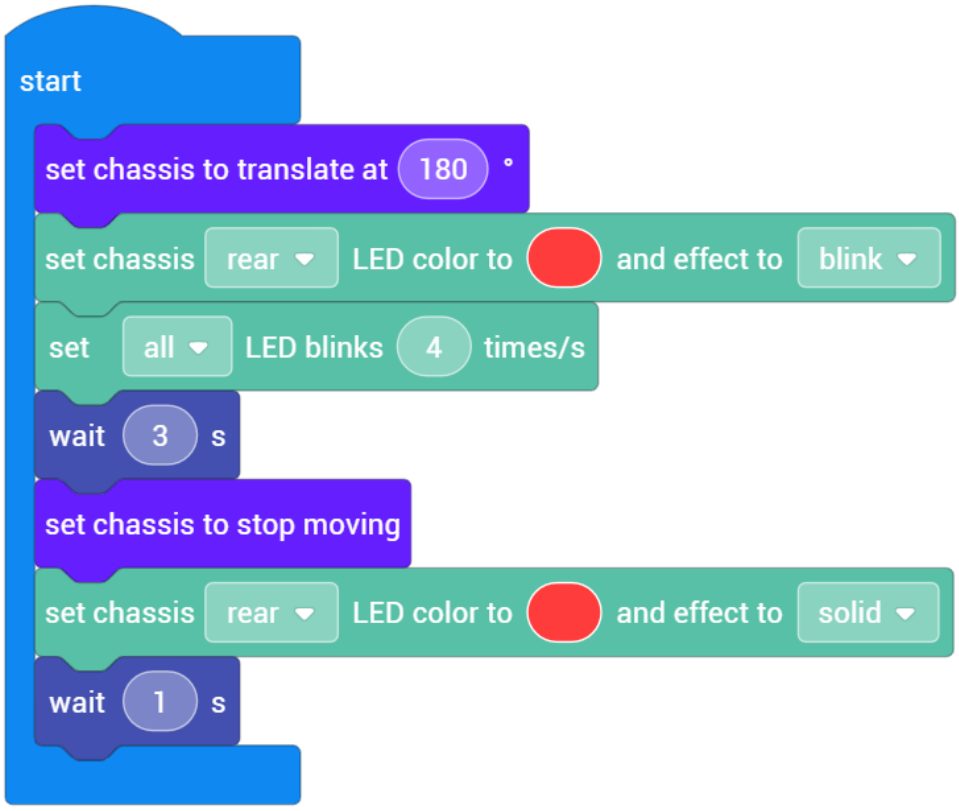
- time (float)

LED Effects

1. set (all) LED blinks (2) times/s



(1) Objective: Sets the flash rate for LEDs
(2) Type: Settings block
(3) Example: Configure reverse light
This will set the rear LED of the chassis to flash red four times per second when reversing.



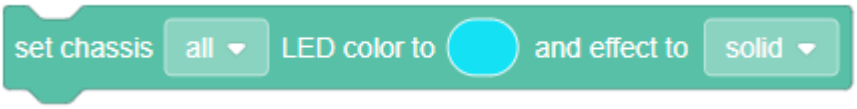
Python API:

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. set chassis (all) LED color to (green) and effect to (solid)



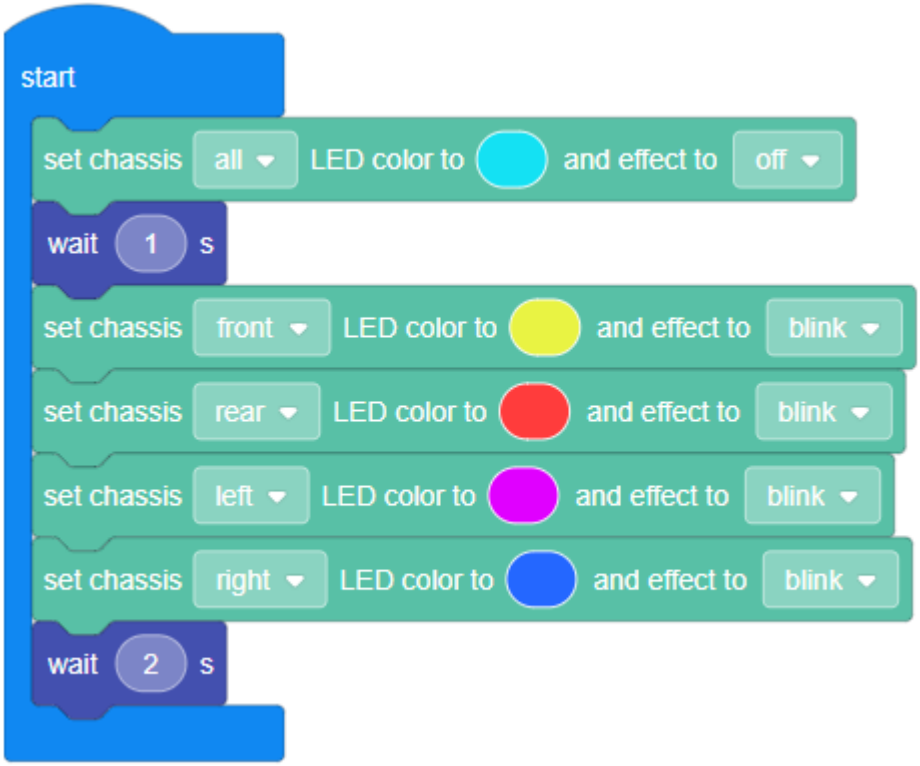
(1) Objective: Sets chassis LED colors and effects

- Solid: LED will remain steady
- Off: LED will switch off
- Pulse: LED will flicker
- Blink: LED will blink at a specified frequency

(2) Type: Execution block

(3) Example: Display streaming color effects

This will set the robot to switch off all LEDs on the chassis for one second, and then flash all LED colors in sequence.



Python API:

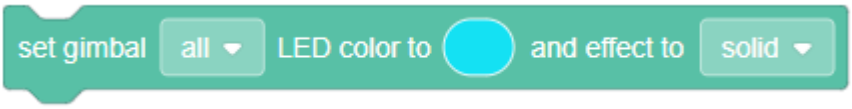
Function: led_ctrl.set_bottom_led(armor_enum, r, g, b, led_effect_enum)

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. set gimbal (all) LED color to (green) and effect to (solid)



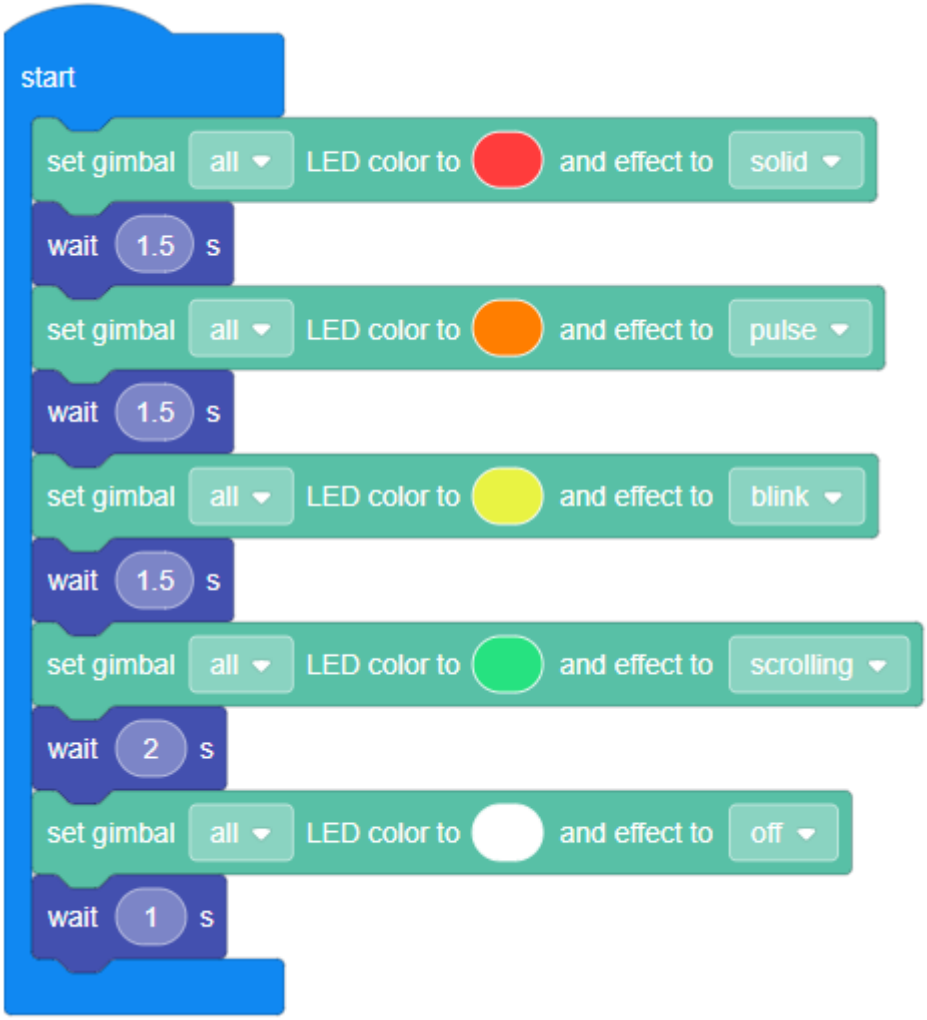
(1) Objective: Sets gimbal LED colors and effects

- Solid: LED will remain steady
- Off: LED will switch off
- Pulse: LED will flicker
- Blink: LED will blink at a specified frequency
- Scrolling: Eight LEDs arranged in a circle will light up in clockwise direction

(2) Type: Execution block

(3) Example: Show off gimbal LED effects

This will set the robot to display all five gimbal LED effects in sequence.



Python API:

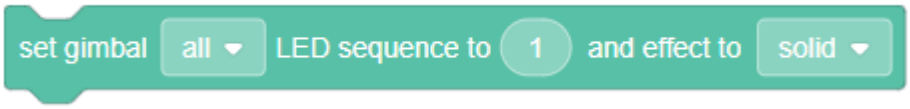
Function: led_ctrl.set_top_led(armor_enum, r, g, b, led_effect_enum)

Parameters:

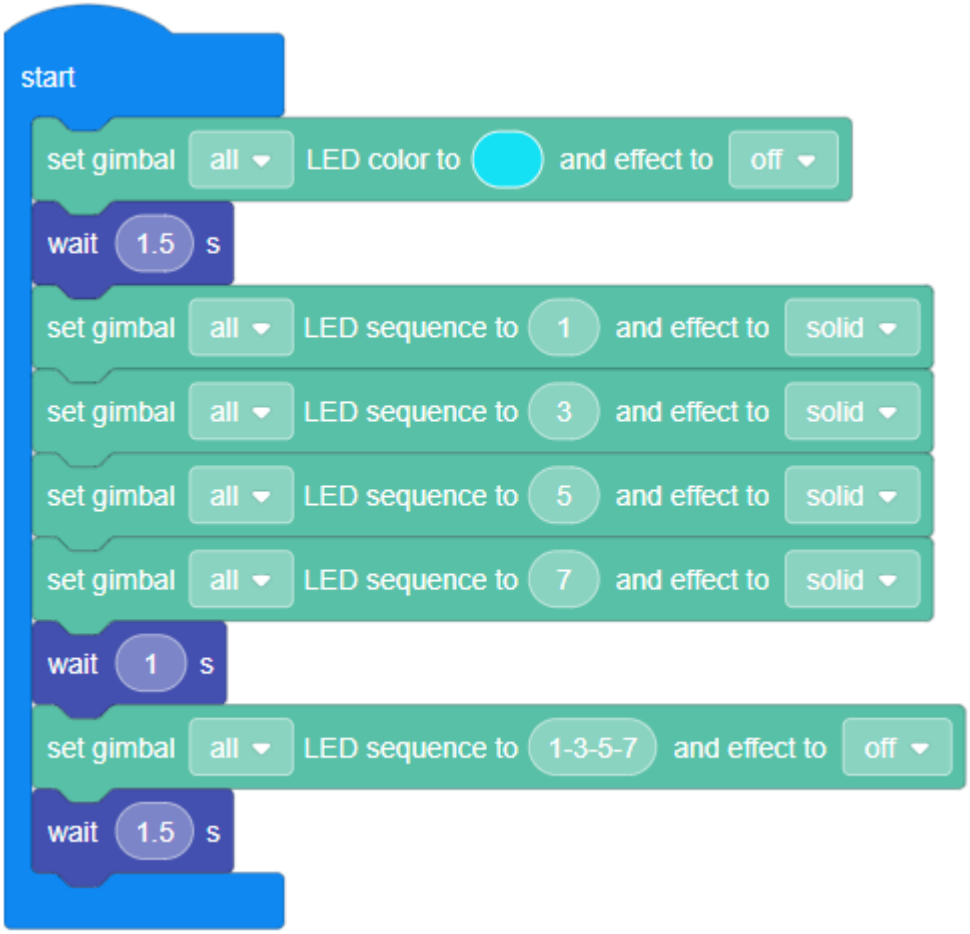
- armor_enum(enum):

    - 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

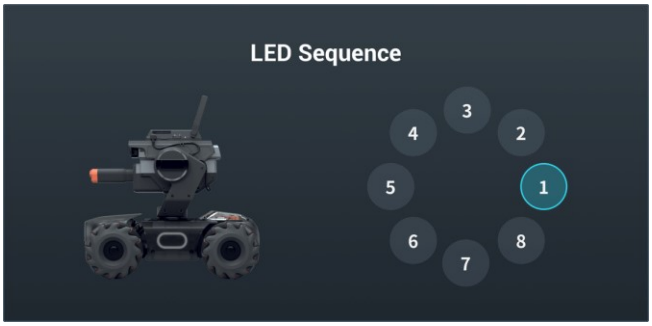
4. set gimbal (all) LED sequence to (1) and effect to (solid)



(1) Objective: Sets the flash sequence for the gimbal LEDs; eight LEDs are located on each side of the gimbal and can be controlled independently
(2) Type: Execution block
(3) Example: Switch on a single light
This will set the robot to switch off all the gimbal LEDs, switch on the odd-numbered LEDs in ascending order, and then switch off all LEDs.



The figure below shows a counterclockwise LED arrangement.

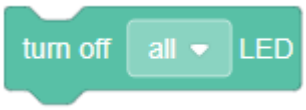


Note:
You can choose multiple LEDs to activate simultaneously.

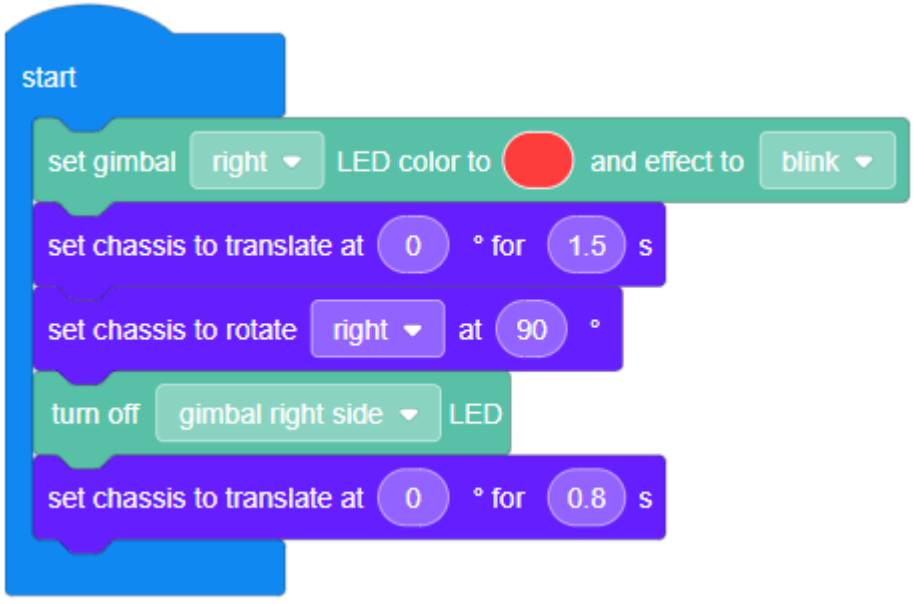
Python API:
Function: led_ctrl.set_signle_led(armor_enum, led_index, led_effect_enum)
Parameters:

- 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. turn off (all) LEDs



(1) Objective: Switches off designated LEDs
(2) Type: Execution block
(3) Example: Configure signal light
This will set the robot to switch on LEDs on the right side of the gimbal before turning right; after the turn is complete, it will switch the LED off.



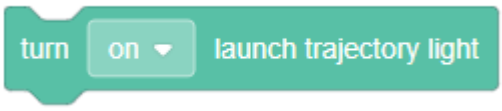
Python API:

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. turn (on) launch trajectory light

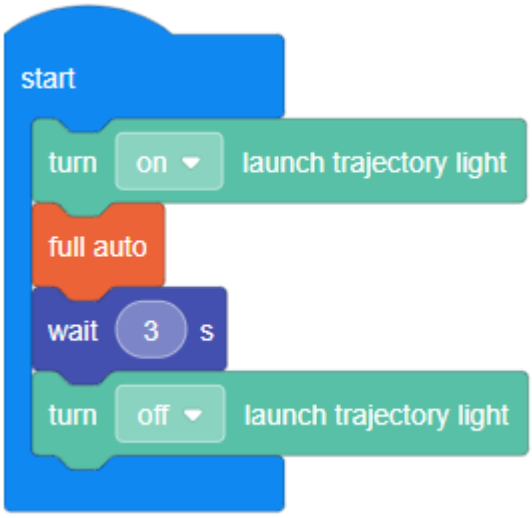


(1) Objective: Controls the trajectory light

(2) Type: Execution block

(3) Example: Launch the trajectory light

This will set the robot to switch on the trajectory light when firing shots.



Python API:

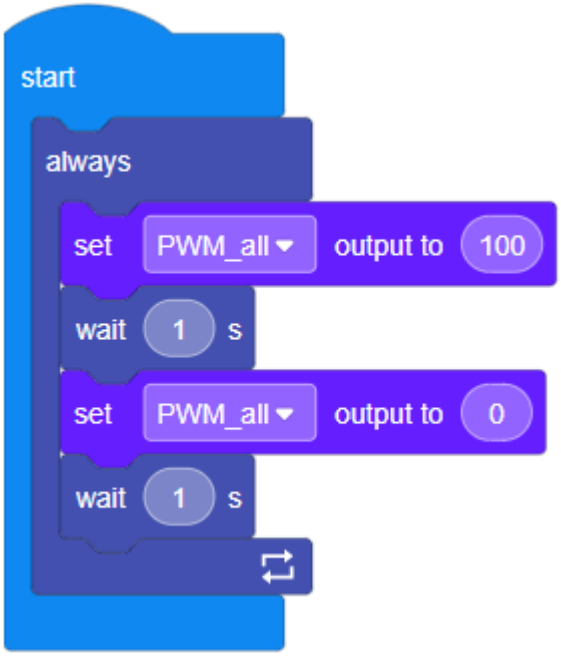
Function: led_ctrl.gun_led_on()

led_ctrl.gun_led_off()

Chassis

1. set (PWM_all) output to (7.5)

(1) Objective: Sets the output percentage for the PWM port; the larger the value used, the longer the port will maintain a high level of output over the specified time period. (The basic frequency for the PWM port is 50Hz.)
(2) Type: Settings block
(3) Examples: Turn LED on/off, Rotate navigation gear
① Turn LED on/off
This enables you to connect an LED to any PWM port and then turn the LED on or off.

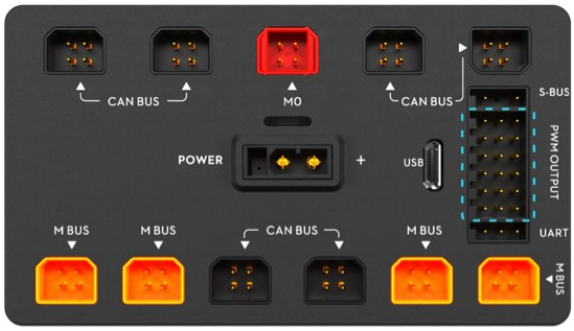


② Rotate navigation gear
This enables you to connect an external navigation gear to any PWM port and then control its rotation.

Notes:

1) PWM ports are located on the chassis control module. To access them, remove the transparent cover.

There are six PWM ports in total.

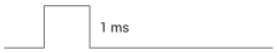


2) PWM (pulse width modulation) controls the duration of a high level of output during a certain period, and is broadly used to control LEDs, navigation gears, and more.

3) For LEDs, the PWM output rate ranges from 0% to 100%, with 0% corresponding to an LED's lowest brightness and 100% to its highest brightness.

4) For navigation gears, the PWM output rate ranges from 2.5% to 12.5%. Because most navigation gears have a control impulse frequency of 50Hz and a control period of 20 ms, and because the high-level pulse width of outputs with an adjustable angle range of -90°to 90° ranges from 0.5ms to 2.5ms, control of the navigation gear's duty ratio ranges from 0.5/20 to 2.5/20, which is to say from 2.5% to 12.5%.

You can set the navigation gear PWM output percentage based on the rotation angles you wish to control.

Pulse Width	Servo Angle
	
	
	
	
	

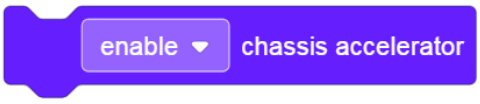
Python API：

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]

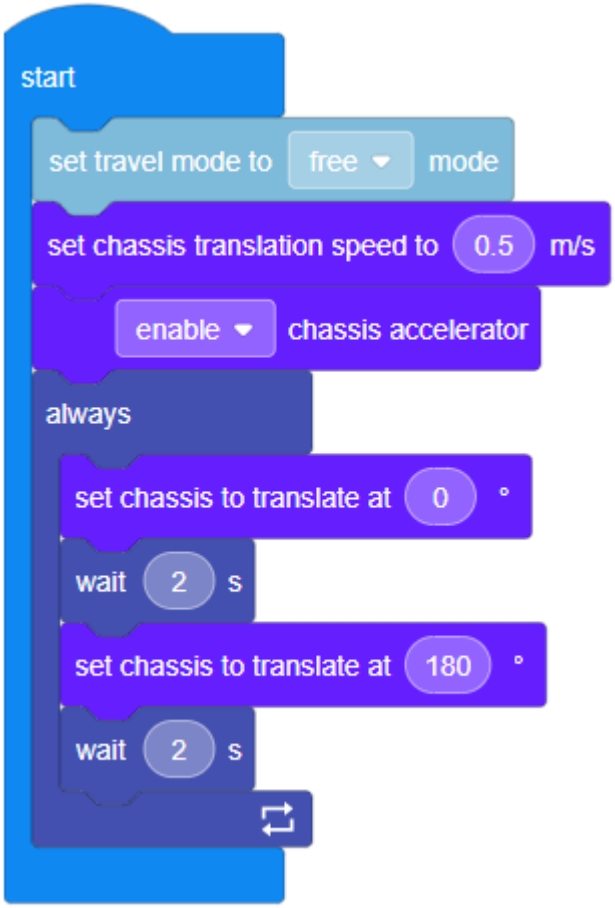
2. (enable) chassis accelerator



(1) Objective: Enables/disables the chassis accelerator

(2) Type: Settings block

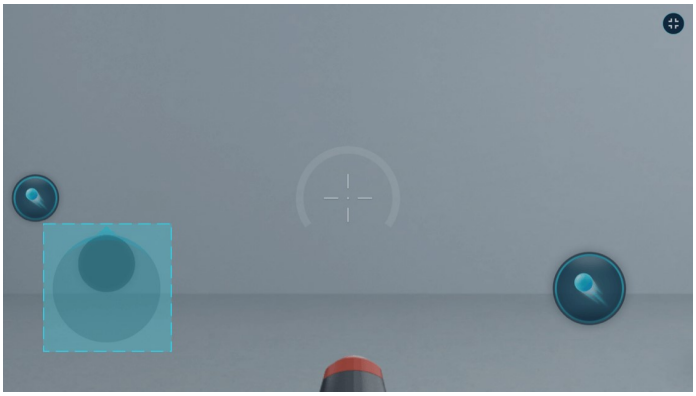
(3) Example: Operate chassis accelerator When the chassis is moving automatically, this will enable you to use the joystick to manually rotate or translate the chassis and increase its translation speed.



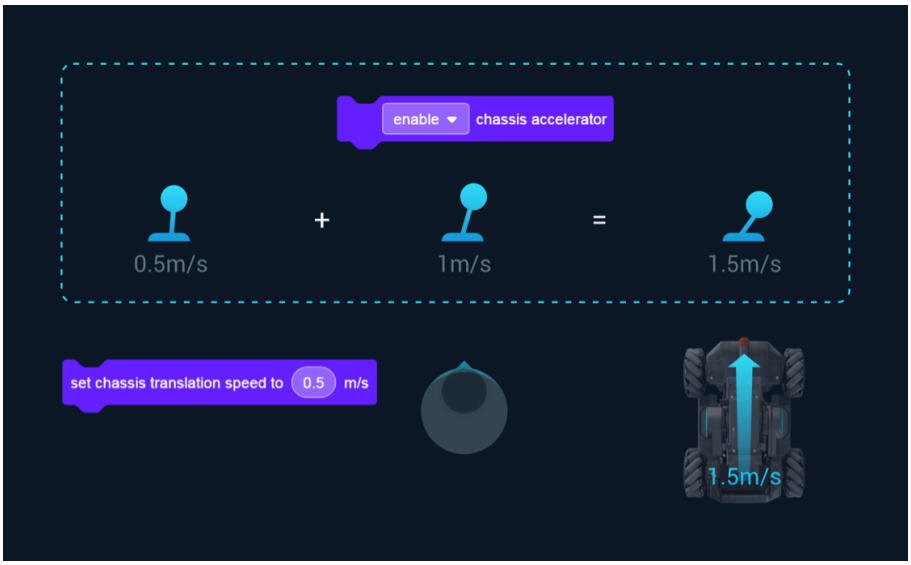
Notes:

1) If the “Enable chassis accelerator” block has not been added, you will not be able to manually control the chassis while running the program. After adding this block, you will be able to manually control and accelerate the robot’s movements.

2) Joystick sensitivity refers to the push range of the joystick; the joystick’s sensitivity ranges from -1 to 1. In the image below, the virtual joystick shown on the FPV interface has reached its upper limit, meaning its sensitivity value is 1.



3) When the chassis accelerator is enabled, the programmed speed will be added to the current speed. As the image below shows, the chassis translates at a programmed speed of 0.5 m/s. When the joystick is pushed to its limit and the chassis accelerator is enabled, the robot will add the two speeds together and translate in a forward direction at a total speed of 1.5 m/s (0.5 m/s + 1 * the joystick’s maximum speed).

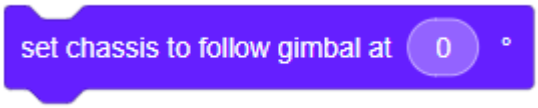


Python API:

Function: chassis_ctrl.enable_stick_overlay()

chassis_ctrl.disable_stick_overlay()

3. set chassis to follow gimbal at (0)°



(1) Objective: In Gimbal Lead Mode, maintains a specific angle for the chassis relative to the movement of the gimbal

(2) Type: Settings block

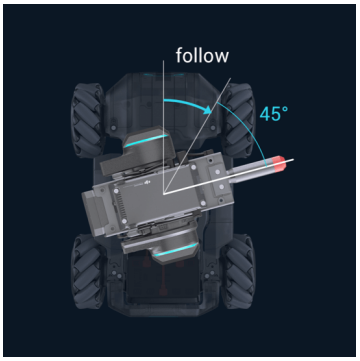
(3) Example: Gimbal Lead Mode

The angle between the chassis and the movement of the gimbal will increase at first, then eventually fall to zero.

Notes:

1)This block will not work in Chassis Lead Mode or in Free Mode.

2)A zero-degree angle between the chassis and the movement of the gimbal means that the chassis and the gimbal are rotating in the same direction along the yaw axis.



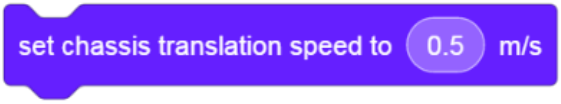
Python API:

Function: chassis_ctrl. set_follow_gimbal_offset(degree)

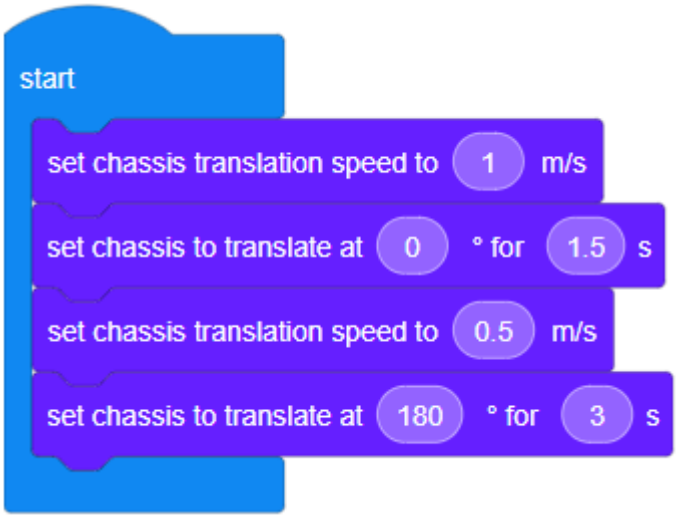
Parameters:

- degree(int): [-180, 180]°

4. set chassis translation speed to (0.5) m/s



(1) Objective: Sets the default translation speed of the chassis to 0.5 m/s; the chassis will move faster when set to a higher speed value.
(2) Type: Settings block
(3) Example: Reduce reversal speed
This will control the chassis to translate forward at 1 m/s for 1.5 seconds, then translate backwards at 0.5 m/s for three seconds to return to the starting point.

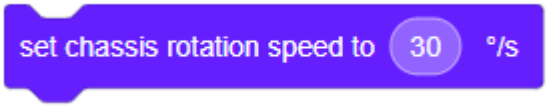


Note:
Please ensure there are no obstacles in the robot's intended path before setting the chassis to a high translation speed.

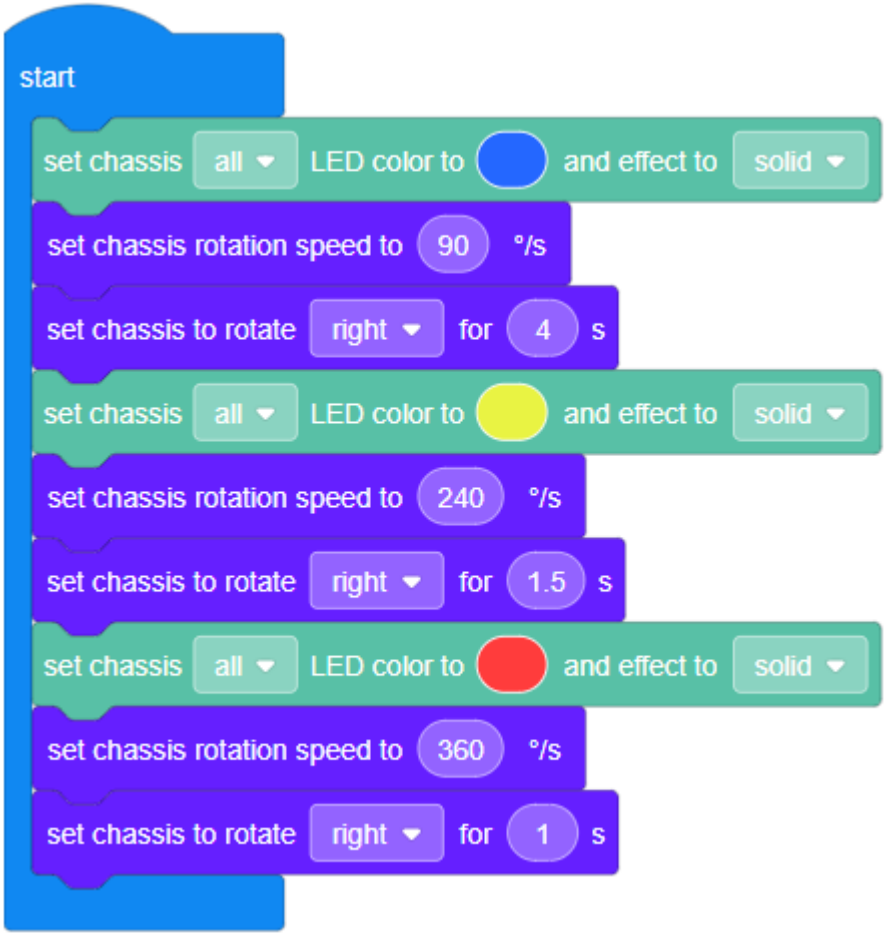
Python API:
Function: chassis_ctrl.set_trans_speed(speed)
Parameters:

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

5. set chassis rotation speed to (30)°/s



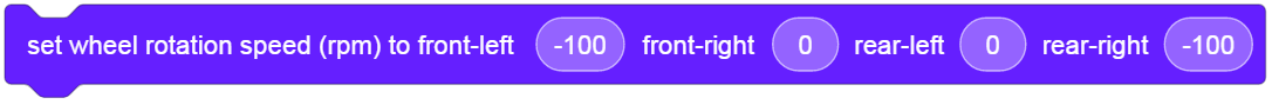
(1) Objective: Sets the default rotation speed of the chassis to 30°/s; the chassis will rotate faster when set to a higher rotation speed value.
(2) Type: Settings block
(3) Example: Set acceleration warning
As the warning LED color changes, the rotation of the chassis will accelerate.



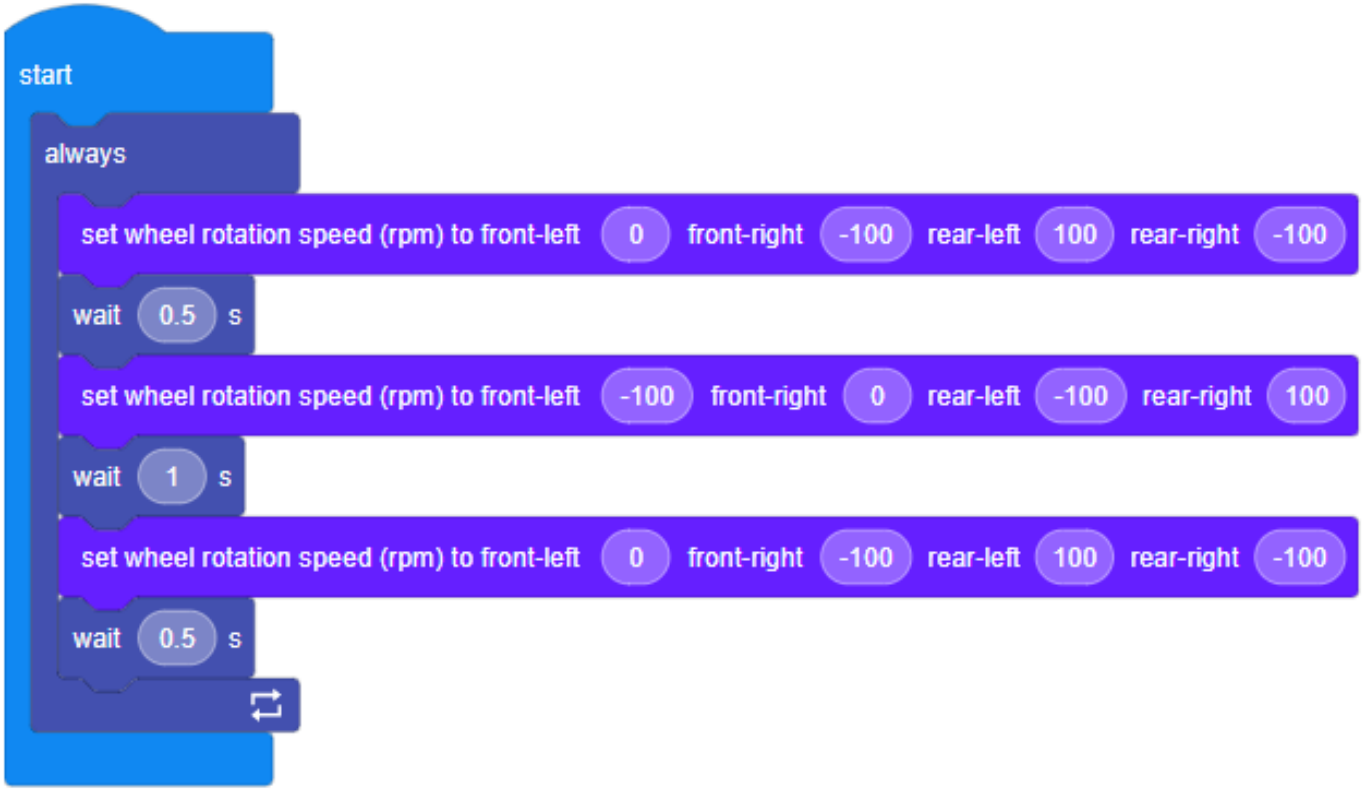
Python API：
Function: chassis_ctrl.set_rotate_speed(speed)
Parameters:

- speed(int): [0, 600] °/s

6. set wheel rotation speed (rpm) to front-left (100) front-right (100) rear-left (100) rear-right (100)



(1) Objective: Allows each wheel's rotation speed to be set independently; only valid combinations of rotation direction and speed will take effect.
(2) Type: Execution block
(3) Examples: Reversing in an "S" pattern, Translating in a circle
① Reversing in an "S" pattern This will control the chassis to move backwards along an S-shaped path.



② Translating in a circle

This will control the robot to translate along a circular path.

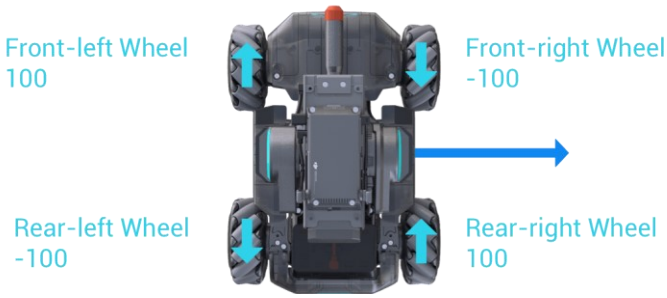
Notes:

1) The robot will translate forward at the default rotation speed of 100 rpm for the front-left wheel, 100 rpm for the front-right wheel, 100 rpm for the rear-left wheel, and 100 rpm for the rear-right wheel.

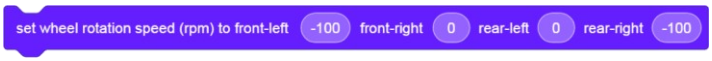
2) To determine a valid combination of wheel rotation directions and speeds, push the robot manually to move it in the desired pattern and observe the rotational direction of each wheel; the rotation speed's value should be positive for wheels rotating forward and negative for wheels rotating backward.

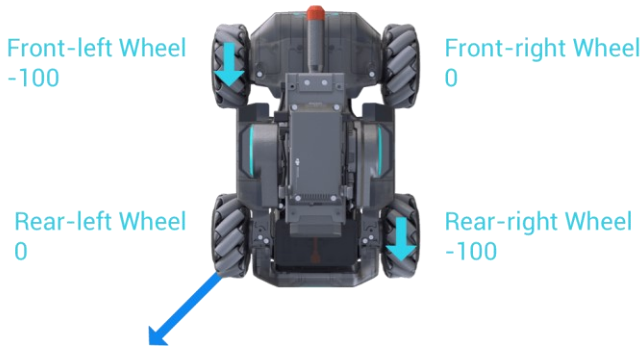
For example:

When the robot translates to the right, the front-left wheel and rear-right wheel will rotate forward, so their rotation speed values should be positive, while the front-right wheel and the rear-left wheel will rotate backward, so their rotation speed values should be negative.

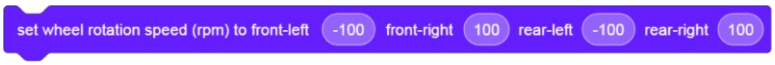


When the robot translates backward and to the left, the front-left wheel and the rear-right wheel will rotate backward, so their rotation speed values should be negative, while the front-right wheel and the rear-left wheel will stay still, so their rotation speed values should be zero.





When the robot rotates to the left, the front-right wheel and the rear-right wheel will rotate forward, so their rotation speed values should be positive, while the front-left wheel and the rear-left wheel will rotate backwards, so their rotation speed values should be negative.



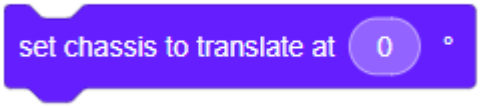
Python API:

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

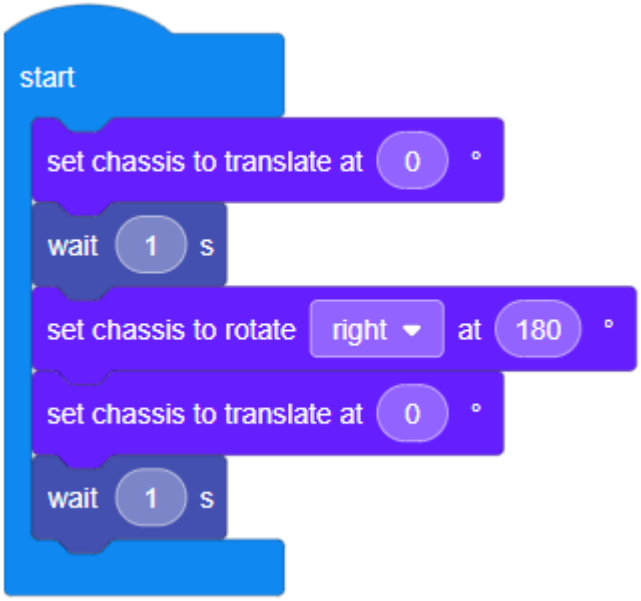
Parameters:

- 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. set chassis to translate at (0)°



(1) Objective: Sets the chassis to translate in a specified direction
(2) Type: Execution block
(3) Example: Make a round trip
This will control the EP to translate forward for one second, then turn around and return to the starting point.

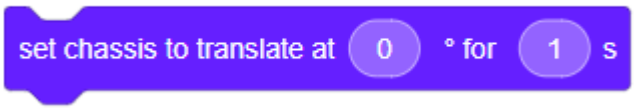


Note:
This block will control the chassis to continuously translate in a specified direction until the robot receives a “set chassis to stop moving,” ”wait (1) s,“ or other command that controls it to stop.

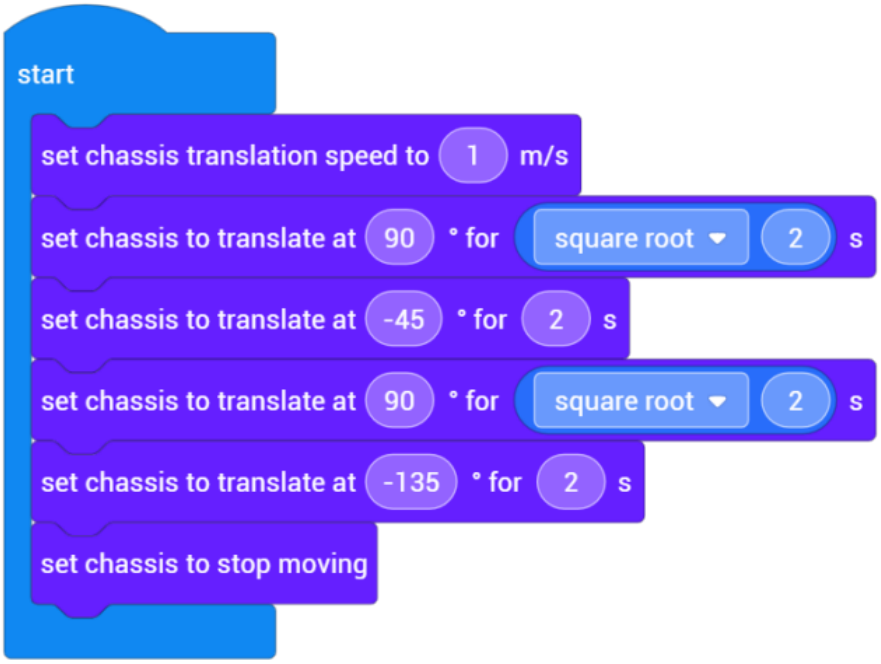
Python API：
Function: chassis_ctrl.move(degree)
Parameters:

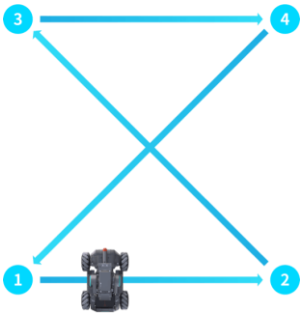
- degree (int): [-180, 180] °

8. set chassis to translate at (0)° for (1) s



(1) Objective: Sets the duration of time the chassis will translate in a specified direction
(2) Type: Execution block
(3) Example: Translate in an X-pattern
This will control the robot to translate to the right, forward-left, right, and backward-left in an X-shaped sequence.





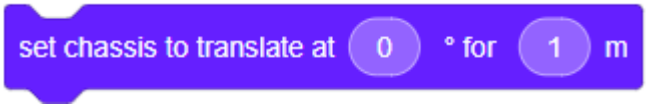
Python API：

Function: chassis_ctrl.move_with_time(degree, time)

Parameters:

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

9. set chassis to translate at (0)° for (1) m

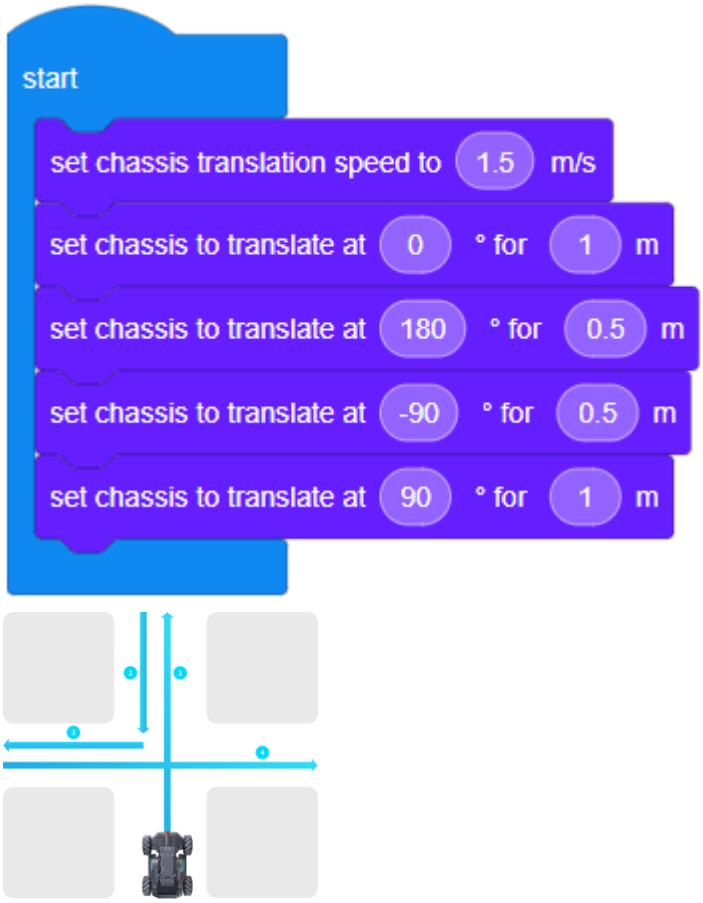


(1) Objective: Sets the distance the chassis will translate in a specified direction

(2) Type: Execution block

(3) Example: Translate in a cross-pattern

This will control the robot to translate forward, backward, left and right in a cross-shaped sequence.



Python API:

Function: chassis_ctrl.move_with_distance(degree, distance)

Parameters:

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

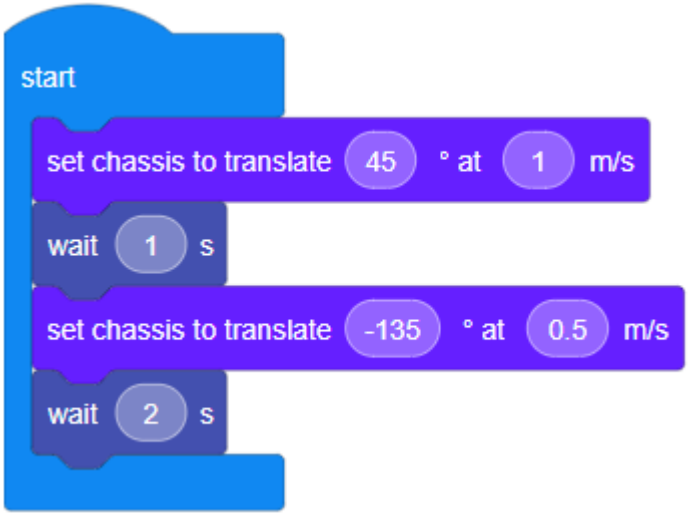
10. set chassis to translate (0)° at (0.5)m/s

(1) Objective: Sets the chassis to translate in a specified direction and at a specified speed

(2) Type: Execution block

(3) Example: Return to starting position

This sets the chassis to translate forward and to the right at 1 m/s, then translate backward and to the left at 0.5 m/s to return to the original position.



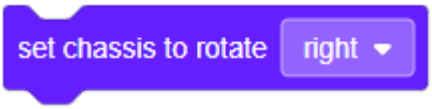
Python API:

Function: chassis_ctrl.move_degree_with_speed(speed, degree)

Parameters:

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

11. set chassis to rotate (right)



(1) Objective: Sets the chassis to rotate in a specified direction

(2) Type: Execution block

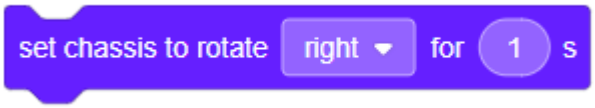
(3) Example: Rotate at variable speed This will control the chassis to rotate increasingly quickly to the right.

Note:
1) This block is not available in Gimbal Lead Mode.
2) Be sure there are no obstacles around the robot before setting a high chassis rotation speed.
3) This block will set the chassis to rotate constantly in a specified direction until it receives a “set chassis to stop moving”,“wait (1) s” or other command that causes it to stop.

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

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

12. set chassis to rotate (right) for (1) s



(1) Objective: Sets the duration the chassis will rotate in a specified direction
(2) Type: Execution block
(3) Example: Cross-rotate the gimbal and chassis

Note:

This block is not available in Gimbal Lead Mode.

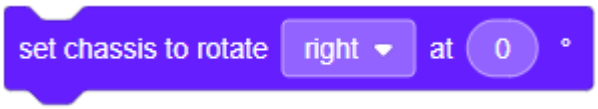
Python API:

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. set chassis to rotate (right) at (0)°



(1) Objective: Sets the angle and direction of chassis rotation

(2) Type: Execution block

(3) Example: Translate continuously back and forth This sets the EP to translate back and forth continuously with the gimbal directed outward.

Note:

This block is not available in Gimbal Lead Mode.

Python API:

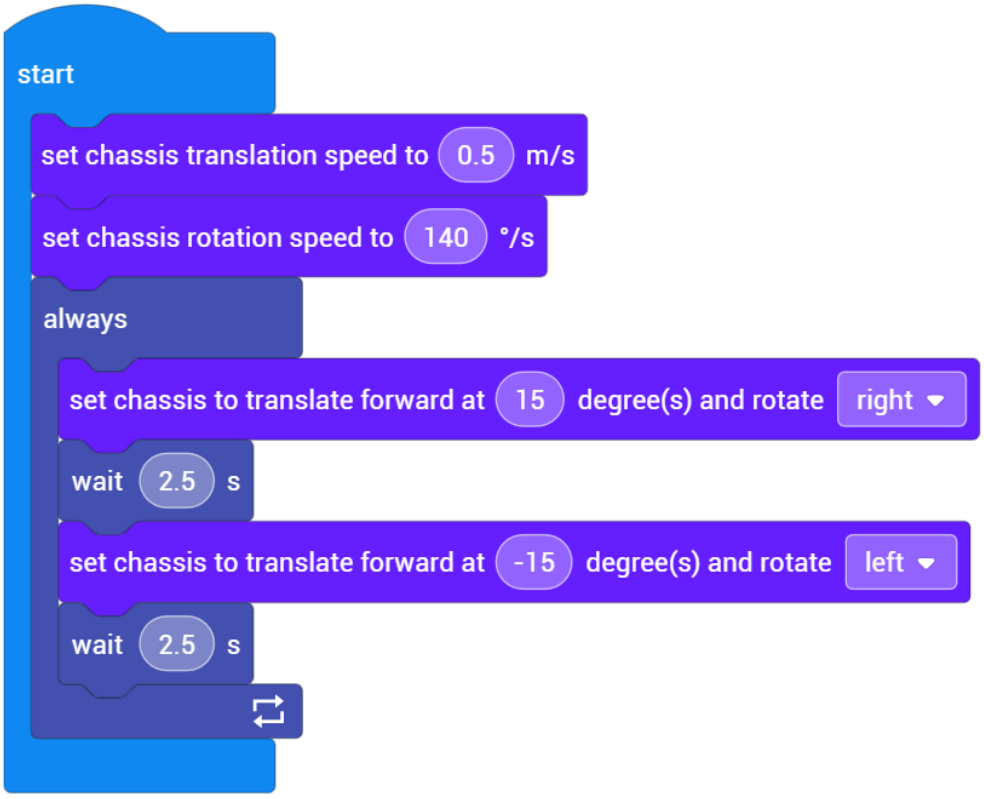
Function: chassis_ctrl.rotate_with_degree(direction_enum, degree)

Parameters:

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

14. set chassis to translate forward at (0)° and rotate (right)

(1) Objective: Sets the chassis to move in a specified direction while rotating simultaneously
(2) Type: Execution block
(3) Example: Translate in a figure-8 pattern
This sets the robot to translate in a figure-8.



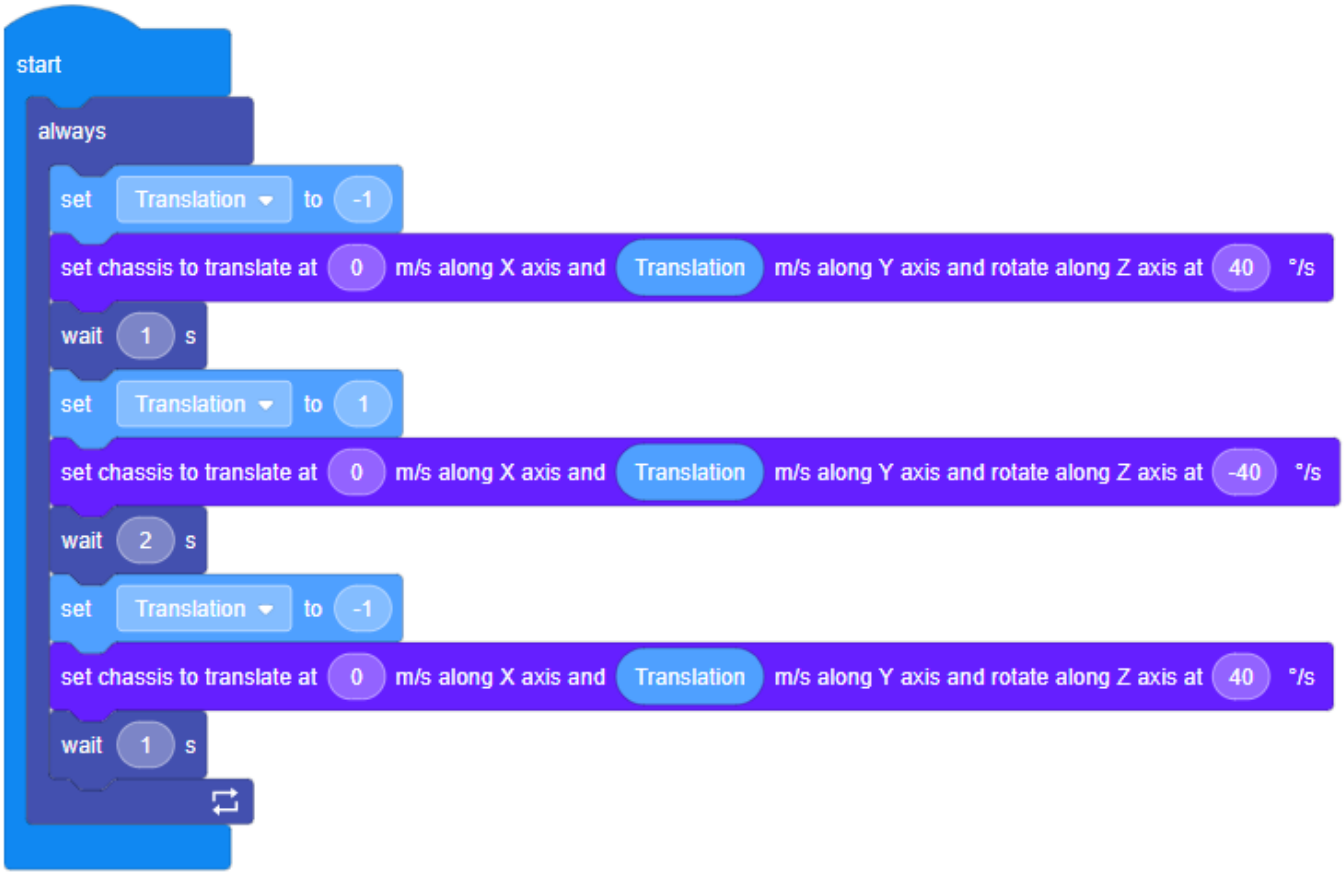
Note:
The “set the chassis to rotate” block is not available in Gimbal Lead Mode; however, the “set the chassis to translate” block is supported in Gimbal Lead Mode.

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. set chassis to translate at (0.5) m/s along X axis and (0.5) m/s along Y axis and rotate along Z axis at (30)°/s

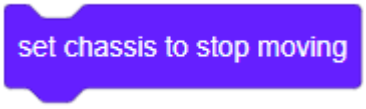
(1) Objective: Sets the chassis to translate in a specified direction and at a specified speed
(2) Type: Execution block
(3) Example: Translate in a circle with the gimbal aimed toward the central point



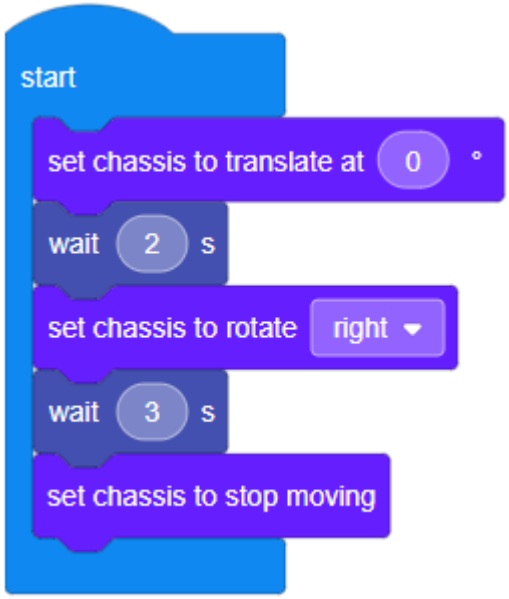
Python API:
Function: chassis_ctrl.move_with_speed(speed_x, speed_y, speed_rotation)
Parameters:

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

16. set chassis to stop moving



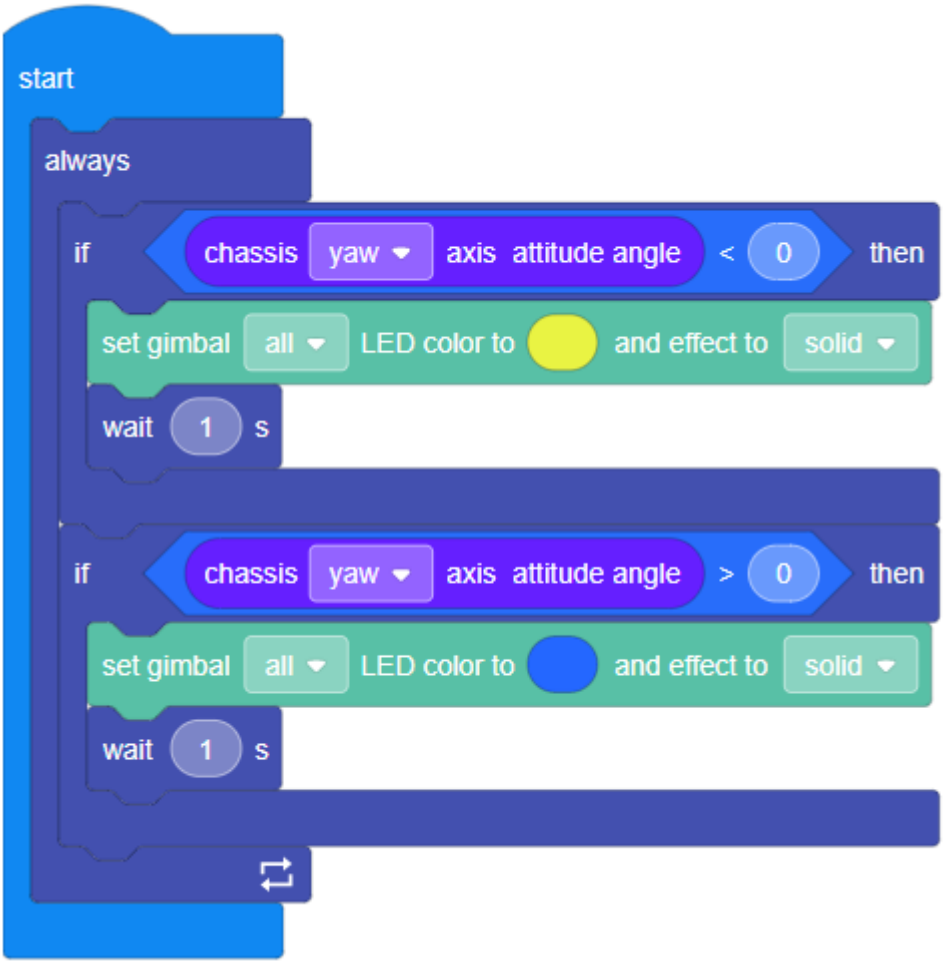
(1) Objective: Stops all chassis movements
(2) Type: Execution block
(3) Example: Rotate chassis right
This sets the chassis to translate forward for 2 seconds at default speed, then turn right and stop moving



Python API：
Function: chassis_ctrl.stop()

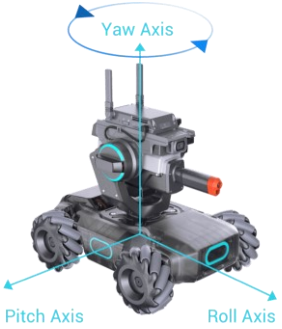
17. chassis (yaw) axis attitude angle

(1) Objective: Obtains the current pitch angle for the chassis along its current yaw/pitch/roll axes based on the chassis location when the robot begins running
(2) Type: Information block (variable-type)
(3) Example: Signal turn
This sets the yellow LED indicator to come on when you manually control the robot to turn left, and sets the blue LED indicator to come on when you manually control the robot to turn right.

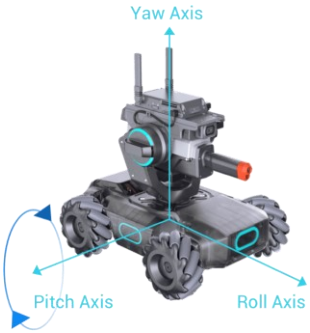


Note:

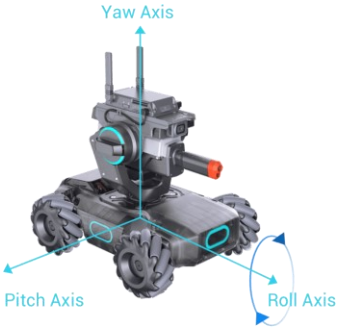
1) Rotating the chassis along the yaw axis results in left and right rotation.



2)Rotating the chassis along the pitch axis results in up and down rotation.



3) Rotating the chassis along the roll axis will cause the robot to turn over.



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:

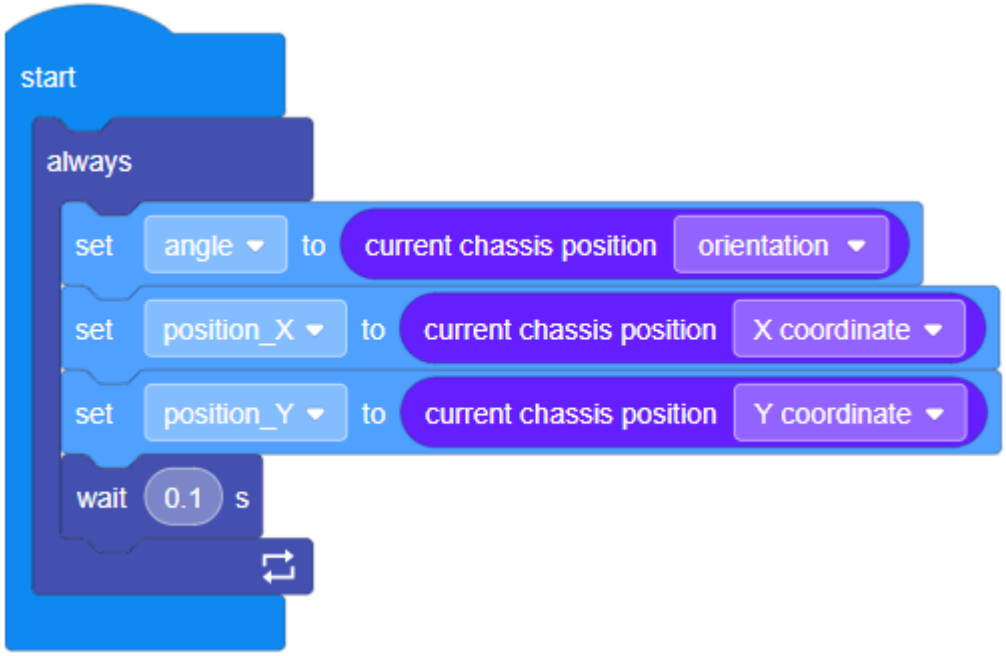
- degree(float)

18. current chassis position (X coordinate)

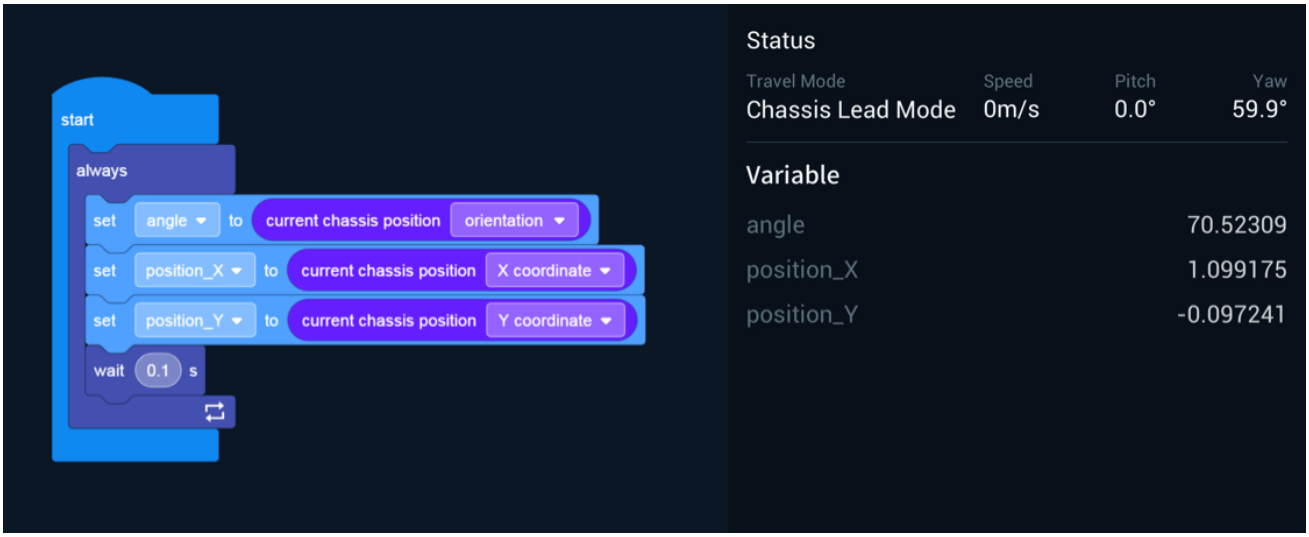
(1) Objective: Obtains the current location coordinates and orientation of the chassis

(2) Type: Information block (variable-type)

(3) Example: Obtain current position information This enables you to check numerical data for the current chassis position by manually pushing the robot back and forth or turning it left and right.



You can observe numerical changes using the FPV interface.



Note:
Because the chassis is under closed-loop control, it is normal to feel some resistance when moving the robot manually.

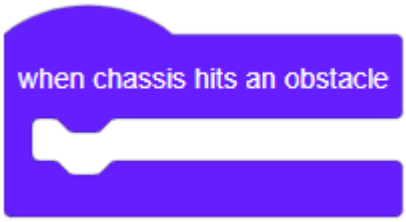
Python API：
Function: chassis_ctrl.get_position_based_power_on(action_enum)
Parameters:

- action_enum(enum):
  - rm_define.chassis_forward
  - rm_define.chassis_translation
  - rm_define.chassis_rotate

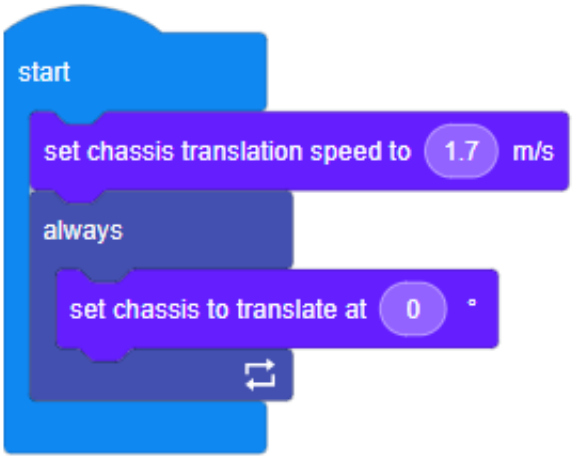
Return value:

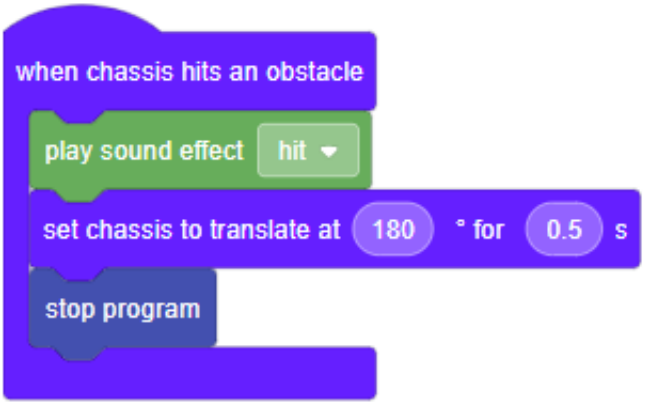
- position(float)

19. when chassis hits an obstacle



(1) Objective: Runs the program for the corresponding block when the chassis hits an obstacle while driving
(2) Type: Event block
(3) Example: Self-defend
This enables the self-defense mechanism to activate when the chassis impacts an obstacle, causing the robot to retreat and stop running the block.





Python API:

Function: def chassis_impact_detection(msg)

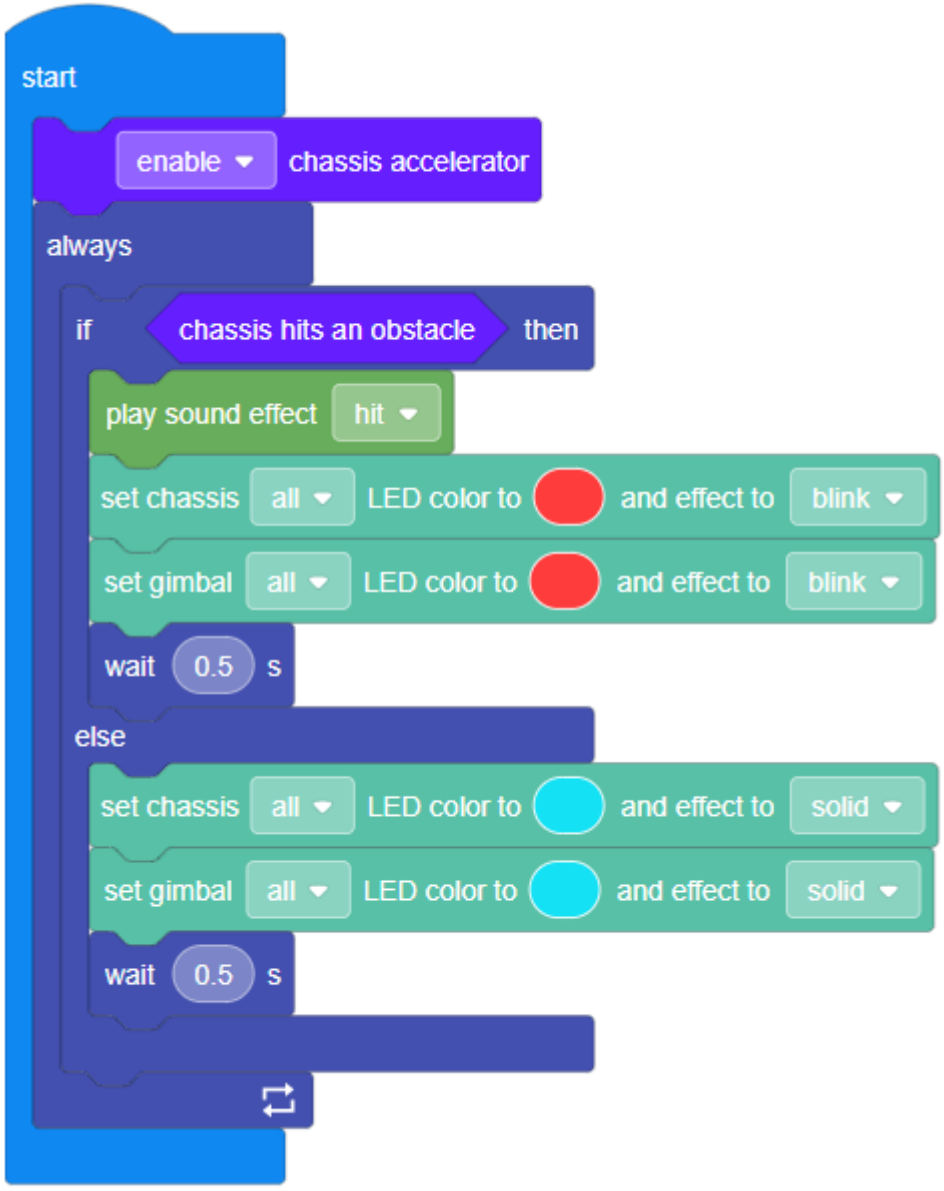
Type: Event callback

20. chassis hits an obstacle

(1) Objective: Returns “True” when the chassis hits an obstacle while driving; otherwise, returns “False”

(2) Type: Boolean block

(3) Example: Emit danger warning If the chassis impacts an obstacle while the robot is being controlled through the FPV interface, the “hit” sound will play and all red LED indicators for the chassis and gimbal will blink; in all other scenarios,these indicators will remain on in a steadystate.



Python API：

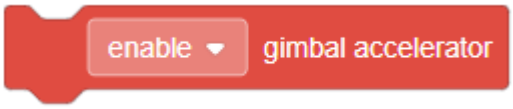
Function:chassis_ctrl.is_impact()

Return value:

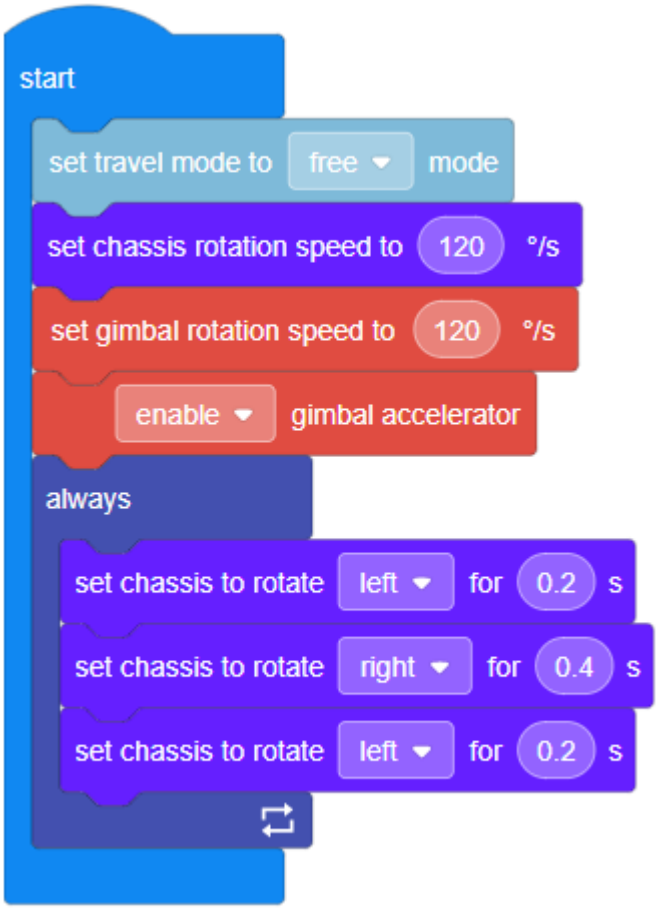
● impact_status(bool)

Gimbal

1. (enable) gimbal accelerator

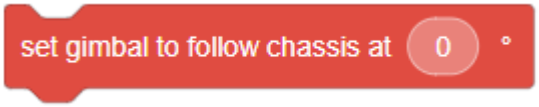


(1) Objective: Enables/disables the gimbal accelerator
(2) Type: Settings block
(3) Example: Activate semi-automatic defense
This allows you to use the FPV interface to manually control the gimbal's rotation for aiming and firing, while the robot's chassis continuously rotates left and right.



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

2. set gimbal to follow chassis at (0)°



(1) Objective: Allows the gimbal to maintain a specific angle according to the chassis' movement in Chassis Lead Mode

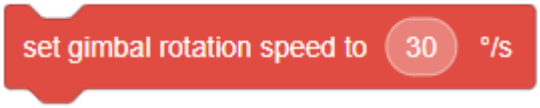
(2) Type: Settings block
(3) Example: Chassis Lead Mode
This will set the gimbal to turn 45 degrees clockwise, and then 90 degrees counterclockwise before returning to the center when the chassis is stationary.

Note:
1)This block is not available in Gimbal Lead Mode or Free Mode.
2) A zero-degree angle indicates that the gimbal and chassis are translating in the same direction along the yaw axis.
3) This is an alternative way to control the gimbal's to left and right rotation.

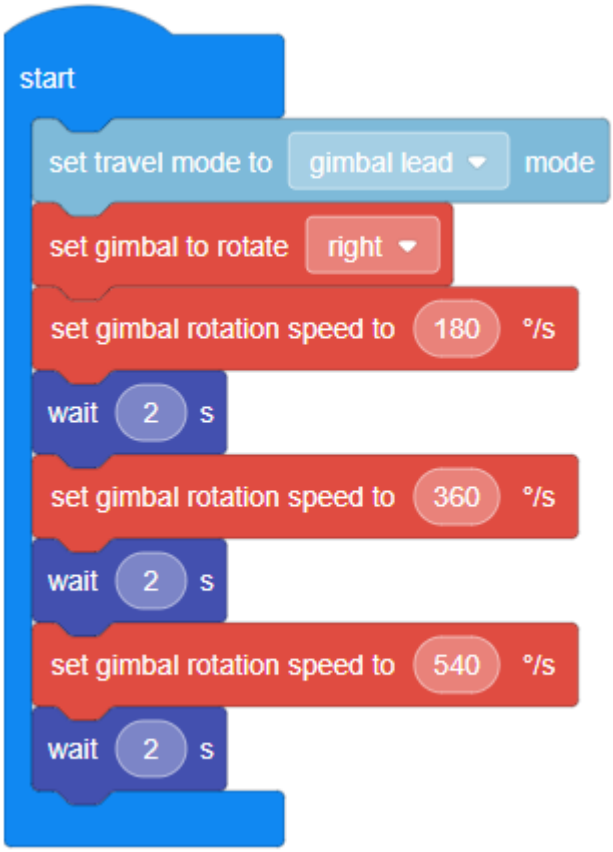
Python API:
Function: gimbal_ctrl.set_follow_chassis_offset(degree)
Parameters:

- degree(int): [-180, 180] °

3. set gimbal rotation speed to (30)°/s



(1) Objective: Sets default gimbal rotation speed to 30°/s; the higher the value, the quicker the rotation
(2) Type: Settings block
(3) Example: Rotate with variable speeds This will set the chassis to rotate clockwise with the gimbal as it gradually increases its speed.

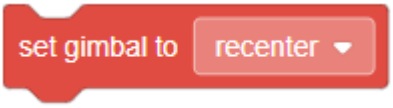


Note:
Before increasing the gimbal's rotation speed, ensure that there are no obstacles near the robot.

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

- speed(float): [0, 540] °/s

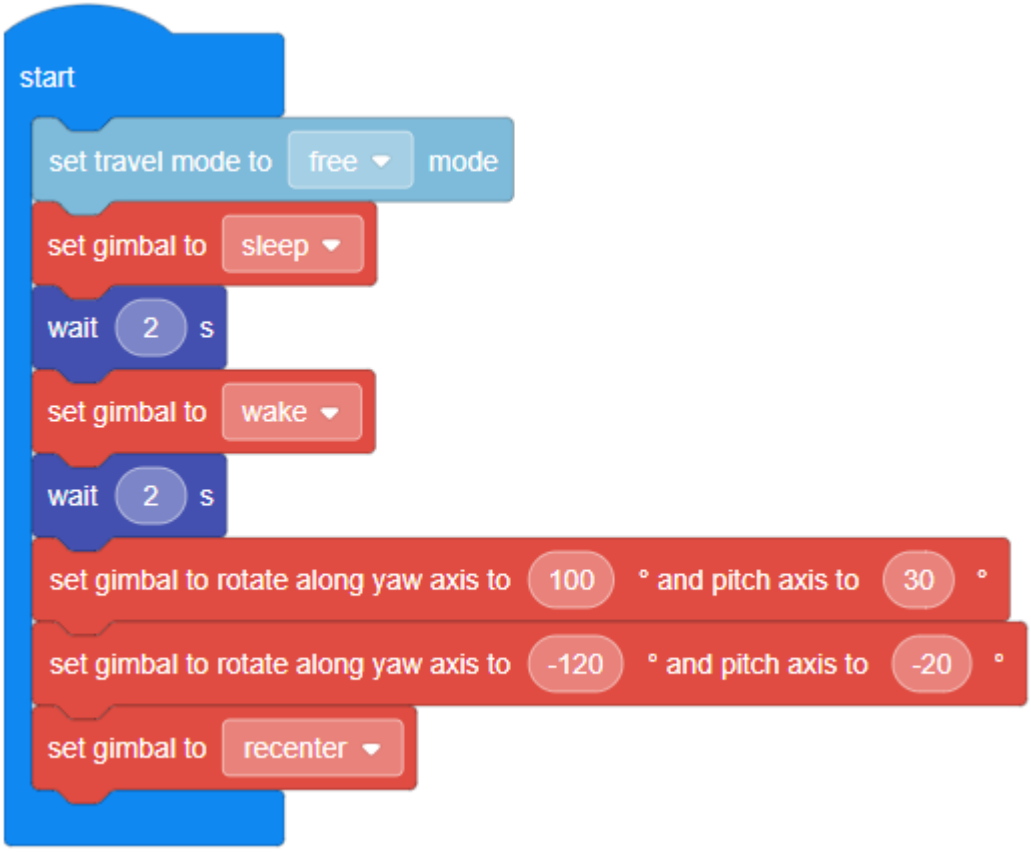
4. set gimbal to (recenter)



(1) Objective: Sets gimbal movements

- Recenter: The gimbal will return to its original position along the pitch axis and yaw axis.
- Stop: The gimbal will stop translating but can still be controlled.
- Sleep: The gimbal will go to sleep.
- Wake: The gimbal will power on again.

(2) Type: Execution block
(3) Example: Wake gimbal up, Hovering
① Wake gimbal up
In Sleep Mode, there is no power from the motor, so the gimbal can be translated freely by hand. When the gimbal wakes up, it will power on again and rotate to its designated position before returning to center.

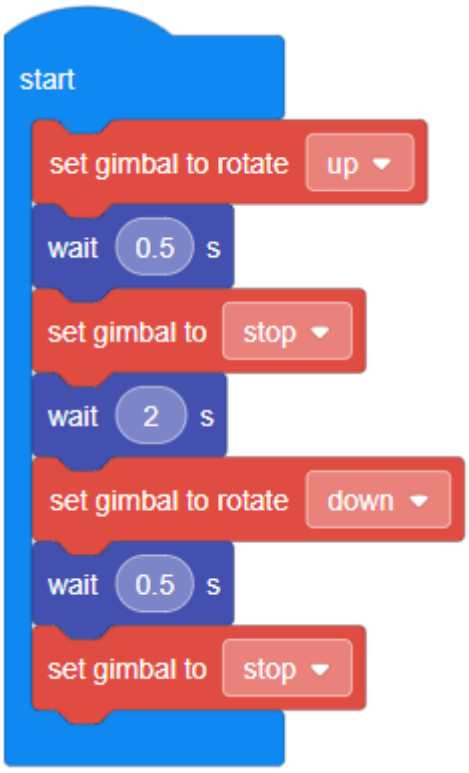


Note:

In Chassis Lead Mode, “set gimbal to recenter” is not available.

②Hovering

This will set the gimbal to rotate upward for 0.5 second, then pause for 2 seconds. It will then rotate downward for 0.5 second, then return to its original position and stop translating.

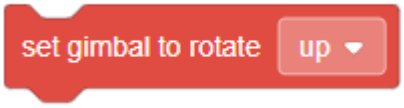


Python API：

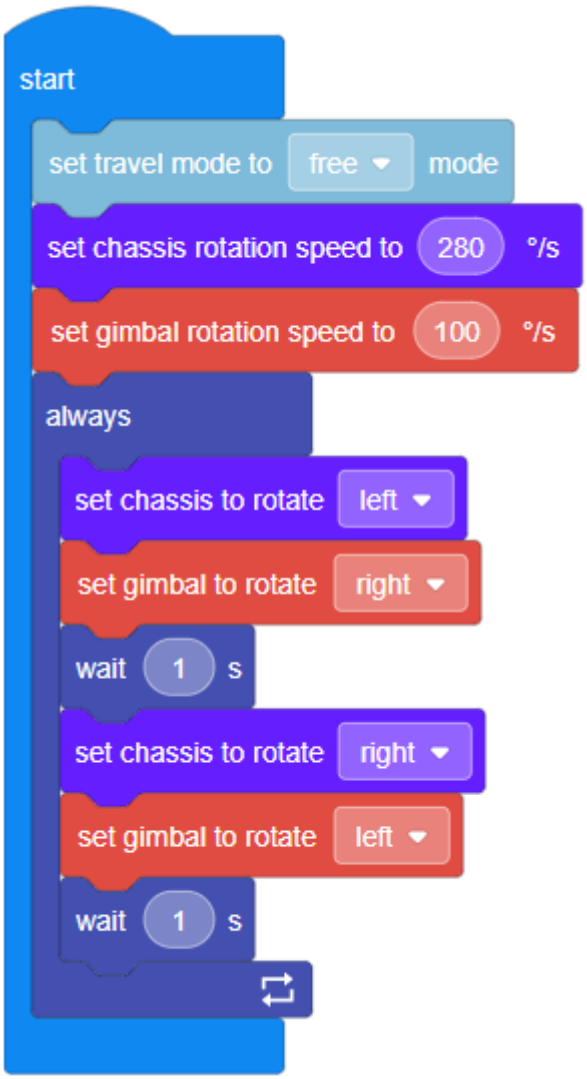
Function: gimbal_ctrl.recenter()

gimbal_ctrl.stop()
gimbal_ctrl.suspend()
gimbal_ctrl.resume()

5. set gimbal to rotate (up)



(1) Objective: Sets a specific rotation direction for the gimbal
(2) Type: Execution block
(3) Example: Rotate chassis and gimbal in opposite directions



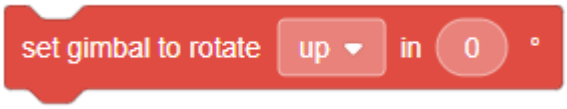
Note:
1) In Chassis Lead Mode, only the “set gimbal to rotate (up/down)” block is available; the “set gimbal to rotate (left/right)” block is not available.
2) This block sets the gimbal to constantly rotate to a specific direction until it receives a command such as “set gimbal to stop”, “wait (1)s”and so on.

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

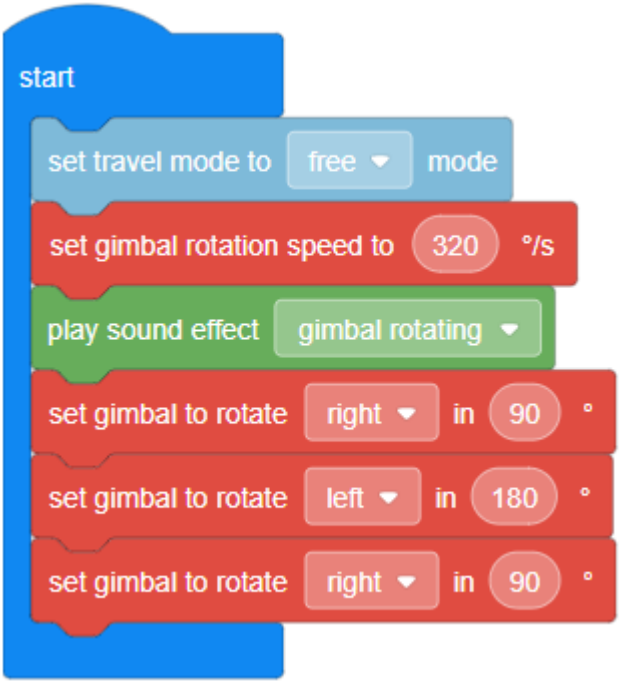
- direction_enum(enum):

- rm_define.gimbal_up
- rm_define.gimbal_down
- rm_define.gimbal_left
- rm_define.gimbal_right

6. set gimbal to rotate (up) in (0)°



(1) Objective: Sets the gimbal to rotate at a specific angle and direction
(2) Type: Execution block
(3) Example: Gimbal rotation
This will set the gimbal to rotate left or right with the selected sound effect.

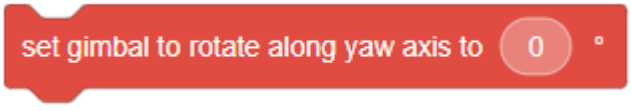


Note:
In Chassis Lead Mode, only the “set gimbal to rotate (up/down) in (0)°” block is available; the “set gimbal to rotate (left/right) in (0)°” block is not available.

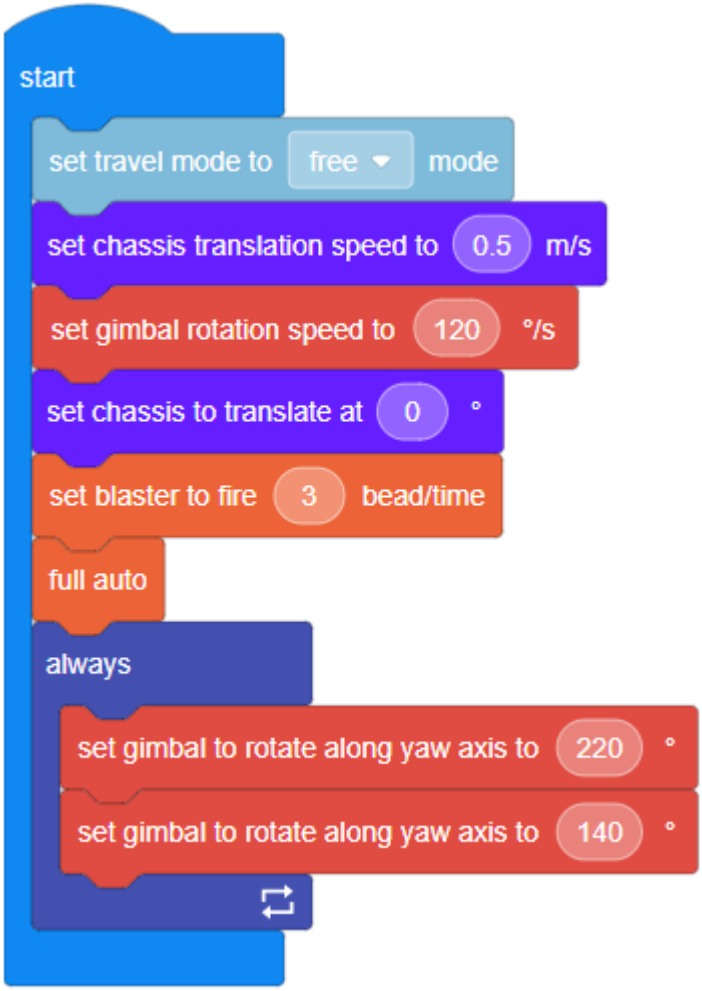
Python API：
Function: gimbal_ctrl.rotate_with_degree(direction_enum, degree)
Parameters:

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

7. set gimbal to rotate along yaw axis to (0)°



(1) Objective: Sets the gimbal to rotate along yaw axis to a designated position
(2) Type: Execution block
(3) Example: Configure rear gun
This will set the chassis to translate forward while the gimbal continuously fires shots behind it.

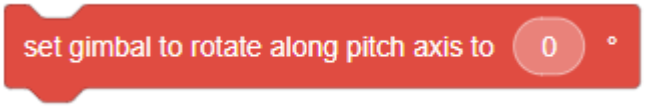


Note:
1) In Chassis Lead mode this block is not available.
2) The gimbal will rotate left and right along the yaw axis and up and down along the pitch axis.

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

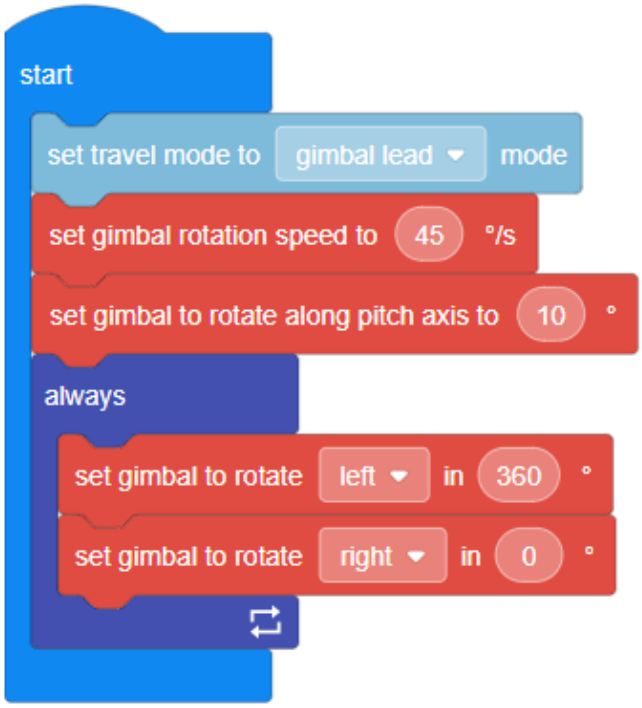
- degree(int): [-250, 250]°

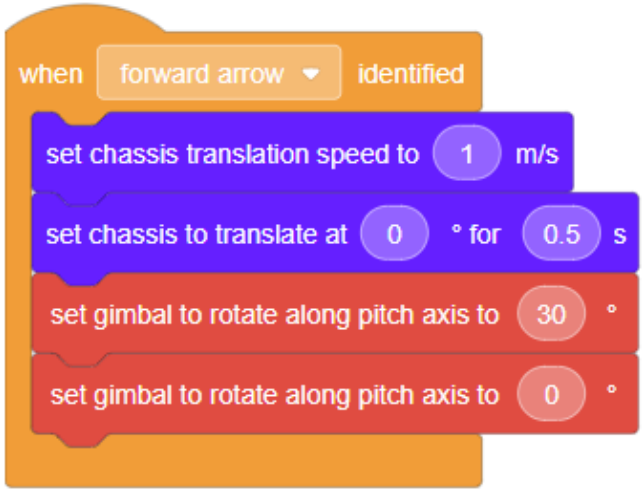
8. set gimbal to rotate along pitch axis to (0)°



(1) Objective: Sets the gimbal to rotate along pitch axis to a designated position
(2) Type: Execution block
(3) Examples: Simulate the power on process, Follow arrow markers
①Simulate power on process
This will simulate the power on process for the robot.

②Follow arrow markers



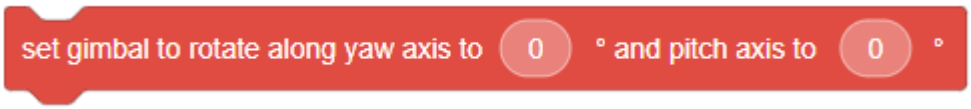


Note:
The gimbal will rotate up and down along the pitch axis and left and right along the yaw axis.

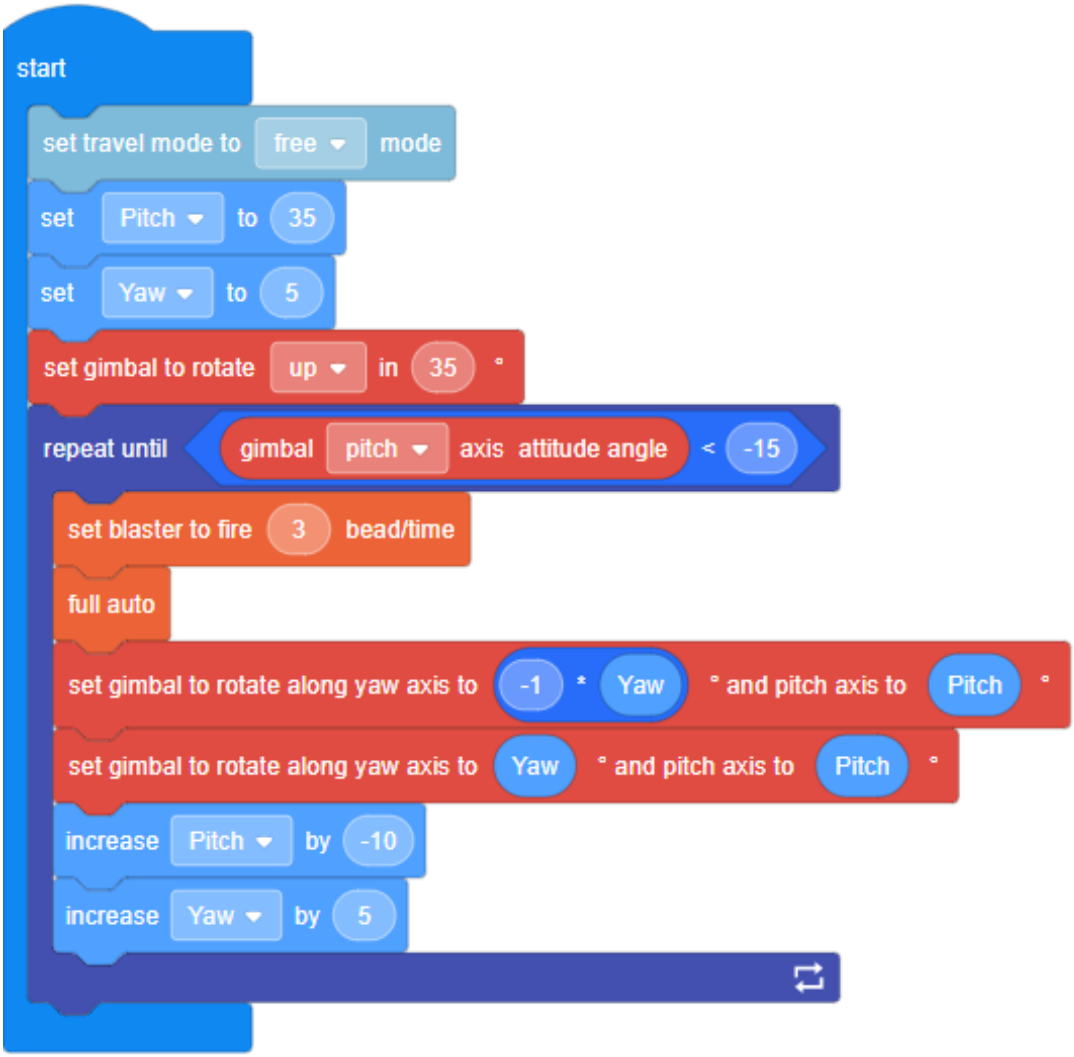
Python API:
Function: gimbal_ctrl.pitch_ctrl(degree)
Parameters:

- degree(int): [-20, 35]°

9. set gimbal to rotate along yaw axis to (0)° and pitch axis to (0)°



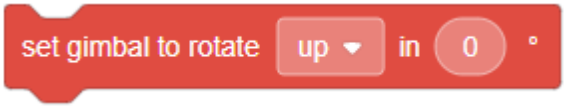
(1) Objective: Sets the gimbal to rotate at a specific angle at a designated position
(2) Type: Execution block
(3) Example: Tiered firing



Note:

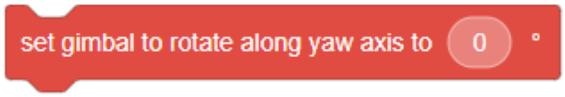
1)In Chassis Lead Mode, the “set gimbal to rotate along pitch axis to (0)°” block is available but the “set gimbal to rotate along yaw axis to (0)°” block is not.

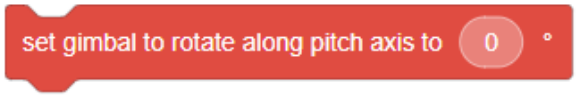
2)

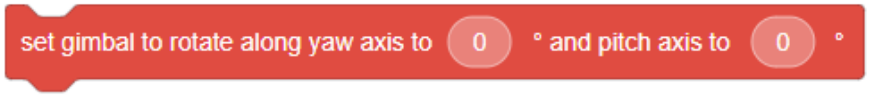


”set gimbal to rotate (up) in (0)°” refers to the position relative to the gimbal’s current position.

3)







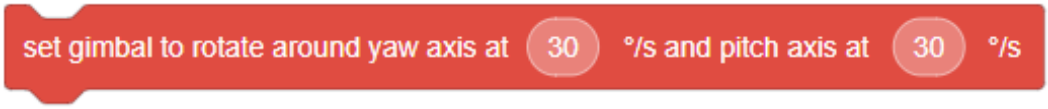
“set gimbal to rotate along yaw axis to (0)°”、“set gimbal to rotate along pitch axis to (0)°”、“set gimbal to rotate along yaw axis to (0)°, and pitch axis to (0)°” refers to the absolute position based on the chassis’ current location.

Python API：

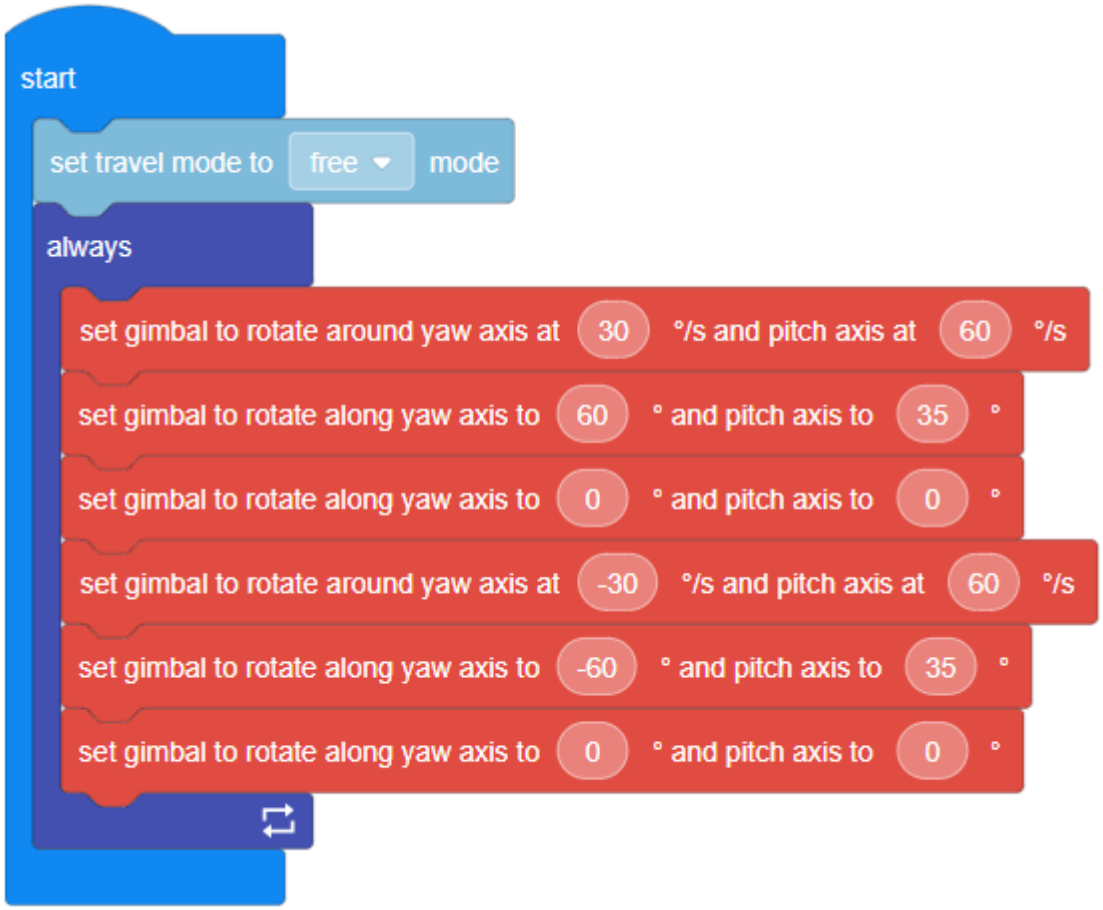
Function: gimbal_ctrl.angle_ctrl(yaw_degree, pitch_degree)
Parameters:

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

10. set gimbal to rotate around yaw axis at (30)°/s and pitch axis at (30)°/s



(1) Objective: Sets the gimbal to rotate on its yaw and pitch axis simultaneously at a specified rotation speed
(2) Type: Execution block
(3) Example: Flexible neck rotation

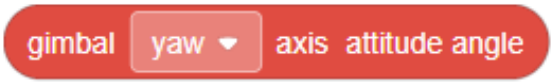


Note:
The value of the rotation speed represents the gimbal's rotation direction.
At the yaw axis, a positive value indicates right rotation, and a negative value indicates left rotation.
At the pitch axis, a positive value indicates upward rotation, and a negative value indicates downward rotation.

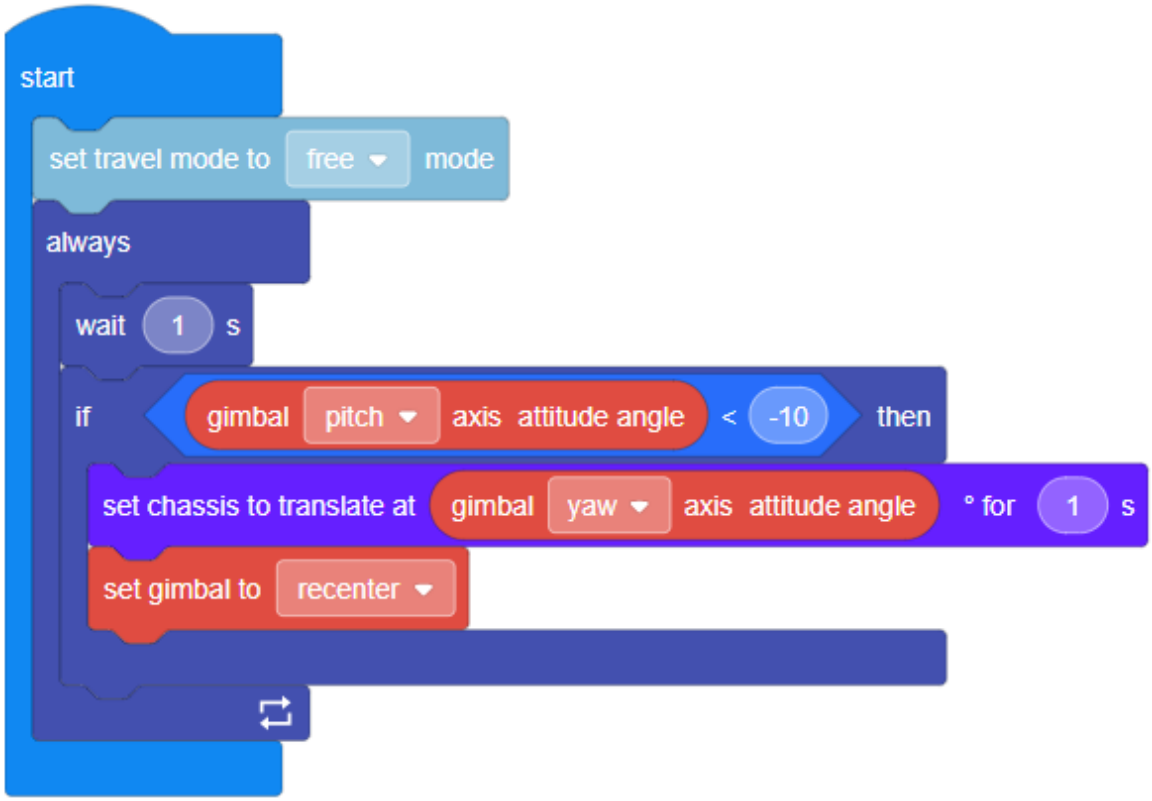
Python API:
Function: gimbal_ctrl.rotate_with_speed(yaw_speed, pitch_speed)
Parameters:

- yaw_speed(float): [-360, 360]°/s
- pitch_speed(float): [-360, 360]°/s

11. gimbal (yaw) axis attitude angle



(1) Objective: Obtains the current attitude angle for the gimbal along its current yaw or pitch axes
(2) Type: Information block (variable-type)
(3) Example: Translate in the gimbal's direction
This will set the gimbal to aim towards a specific direction. The robot will translate towards the specified direction for one second after the gimbal is pressed.



Python API:
Function: gimbal_ctrl.get_axis_angle(axis_enum)
Parameters:

- axis_enum (enum):
  - rm_define.gimbal_axis_yaw
  - rm_define.gimbal_axis_pitch

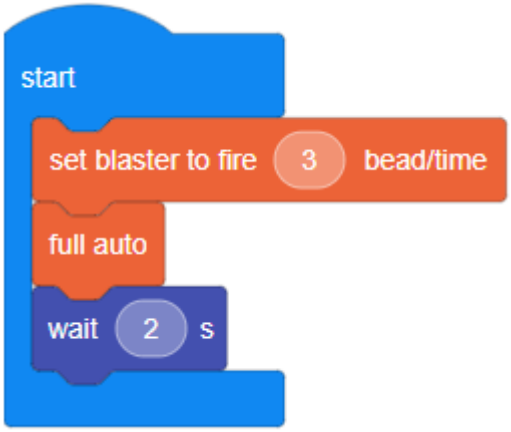
Return value:

- degree(int)

Blaster

1. set blaster to fire (1) bead/time

(1) Objective: Sets the number of beads fired per shot
(2) Type: Settings block
(3) Example: Fire 6 beads in sequence

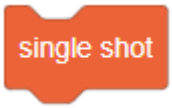


Note:
A maximum of 8 beads can be fired each time.

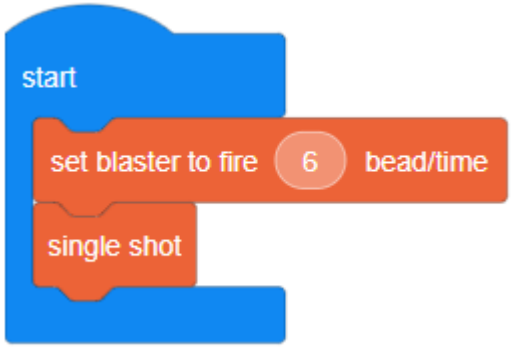
Python API:
Function: gun_ctrl.set_fire_count(count)
Parameters:

- count(int): [1, 8]

2. single shot



(1) Objective: Sets the Blaster to single shot mode
(2) Type: Execution block, Blocking block
(3) Example: Fire single shot



Note:
Single Shot mode means one bead is fired each time by default.
You can shoot several beads at once by setting the “set blaster to fire (1) bead/time” block.

Python API：
Function: gun_ctrl.fire_once()

3. full auto

(1) Objective: Sets the Blaster to full auto mode
(2) Type: Execution block, Non-blocking block
(3) Example: Fire strafe shots

Note:
1) The default firing frequency is one bead per second.
2) The Full Auto block is a non-blocking block, meaning the robot will continuously fire beads until it receives the “Stop the Blaster from Firing” command or the program ends.
3) The difference between Single Shot and Full Auto is as below:

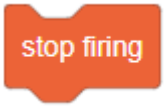
Fires beads continuously for five seconds

Fire one bead and wait for five seconds

Python API:

Function: gun_ctrl.fire_continuous()

4. stop firing



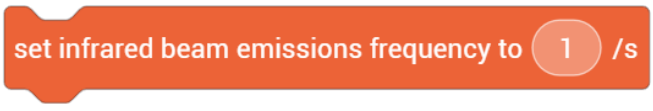
(1) Objective: Stops the Blaster from firing
(2) Type: Execution block
(3) Example: Stop firing

Note:
Because the “single shot”is a blocking block, the “stop firing” block does not apply to it.
”stop firing”only has limitations on “full auto”.

Python API:

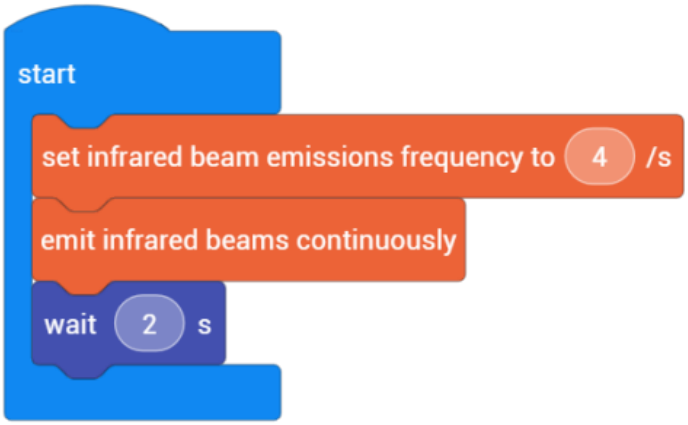
Function: gun_ctrl.stop()

5. set the infrared beam shooting frequency to (1) time per second

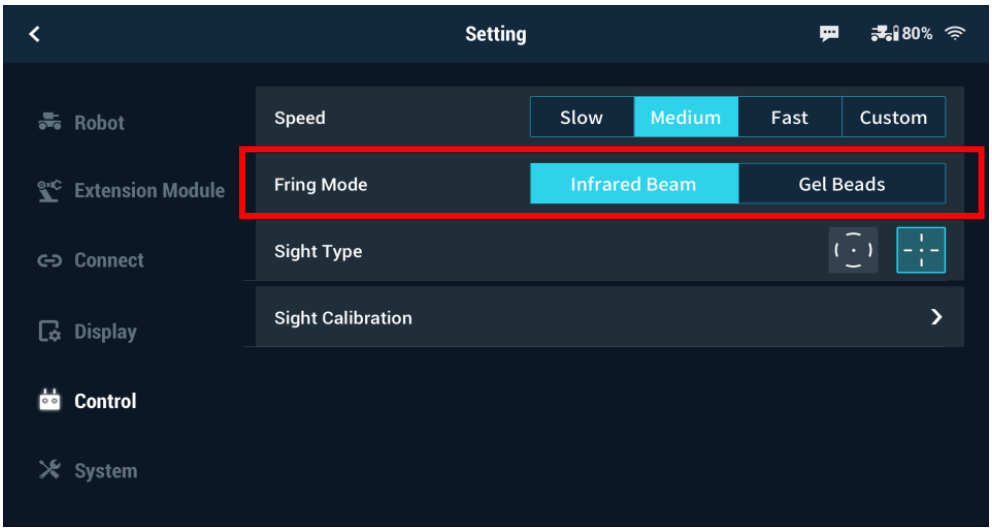


(1) Description: Set the infrared beam shooting frequency, i.e., number of infrared beam shot per second.
(2) Type: Setting
(3) Example: shoot infrared beam 8 consecutive times.

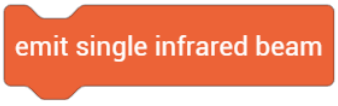
Control the blaster to shoot the infrared beam consecutively within two seconds, four shots per second.



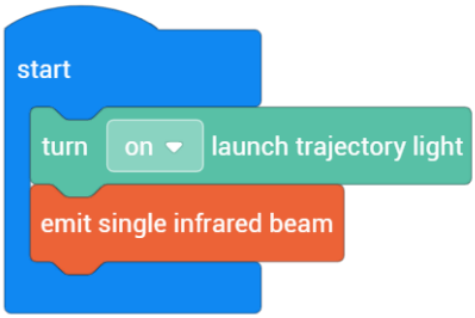
Note:
1) The robot can shoot infrared beam eight times per second at most.
2) This module has no effect on the "Shoot Infrared Beam Once" module.
3) During programming in the laboratory, blast-type operations are not affected by system settings. For example, in Current setting > Control, infrared beam is chosen for the "blast type". However, during programming, the gel bead module is used, so gel beads will still be launched when the program runs. Designs different from the system setting outside the laboratory will not take effect.



6. shoot infrared beams once

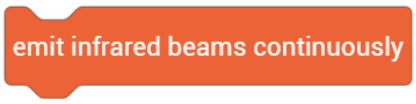


(1) Description: Control blaster to shoot infrared beam for only once.
(2) Type: Execution, Blocking
(3) Example:
The launch trajectory light lights up when shooting infrared beam once.



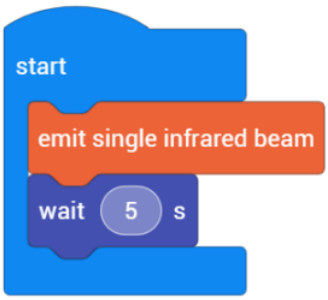
Note:
There is no visible effect when using the modules "Shoot Infrared Beam Once" and "Shoot Infrared Beam Consecutively" to shoot infrared beams.

7. shoot infrared beams consecutively

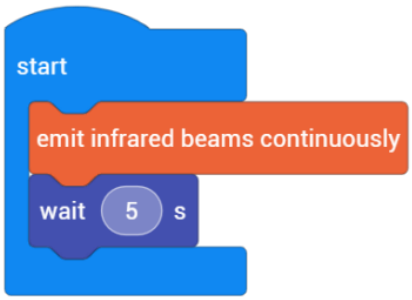


(1) Description: Control the blaster to shoot infrared beams consecutively.
(2) Type: Execution, Non-blocking
(3) Example: Infrared shooting.
The gimbal rotates around the yaw axis and the blaster consecutively shoots the infrared beam.

Note:
1) The default frequency is one shot per second.
2) The module "Shoot Infrared Beam Consecutively" is a non-blocking module, that is, it will continue to shoot until it encounters the "Stop Shooting Infrared Beam" command or until the program ends.
3) The difference between the modules "Shoot Infrared Beam Once" and "Shoot Infrared Beams Consecutively" are as follow:

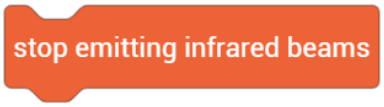


Wait for five seconds quietly after shooting infrared beam once.

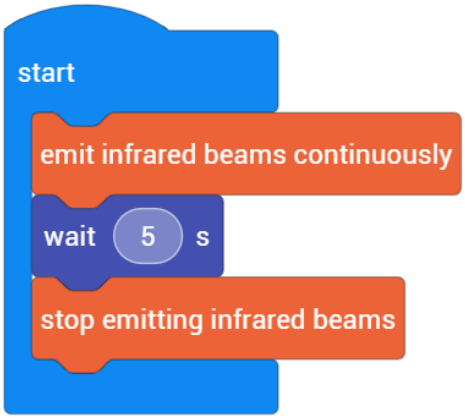


Shoot infrared beams five time consecutively without interruption within five seconds.

8. Stop Shooting Infrared Beam



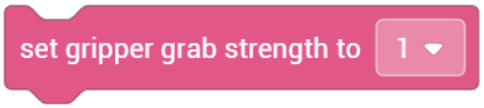
(1) Description: Stop shooting infrared beam.
(2) Type: Execution
(3) Example: Stop shooting
Stop after the blaster shoots infrared beam consecutively for five seconds.



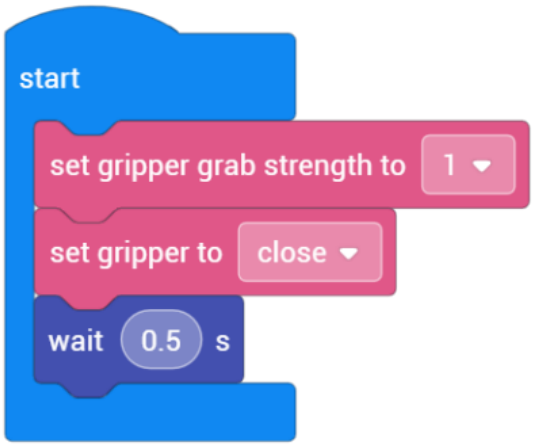
Note:
1) As "Shoot Infrared Beams for a Single Time" is a blocking module, the "Stop Shooting Infrared Beam" command has no effect on it.
2) The "Stop Shooting Infrared Beam" command only restricts the module "Shoot Infrared Beam Consecutively".

Extension Module

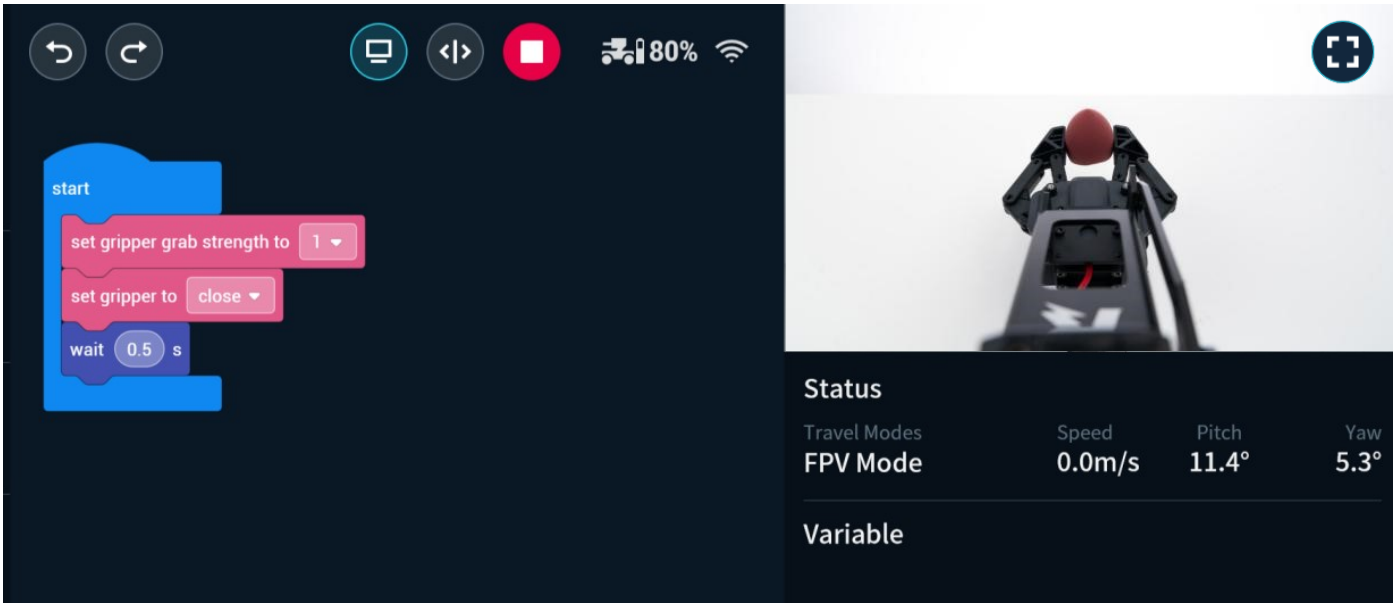
1. set grip force level to level (1)



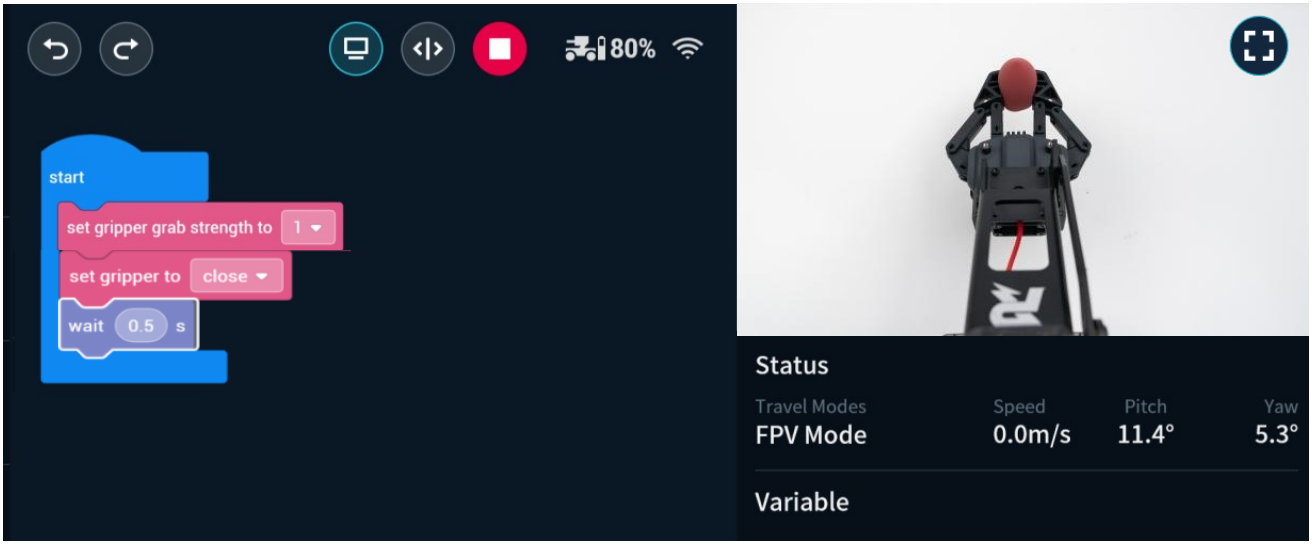
(1) Description: Set the grip force level of the gripper. The higher the level, the greater the grip force.
(2) Type: Setting
(3) Example: Object shape changes upon grip
Place a soft object onto the gripper and continuously increase the grip force. It can be observed through the FPV window that the shape change of the gripped object becomes more and more obvious.



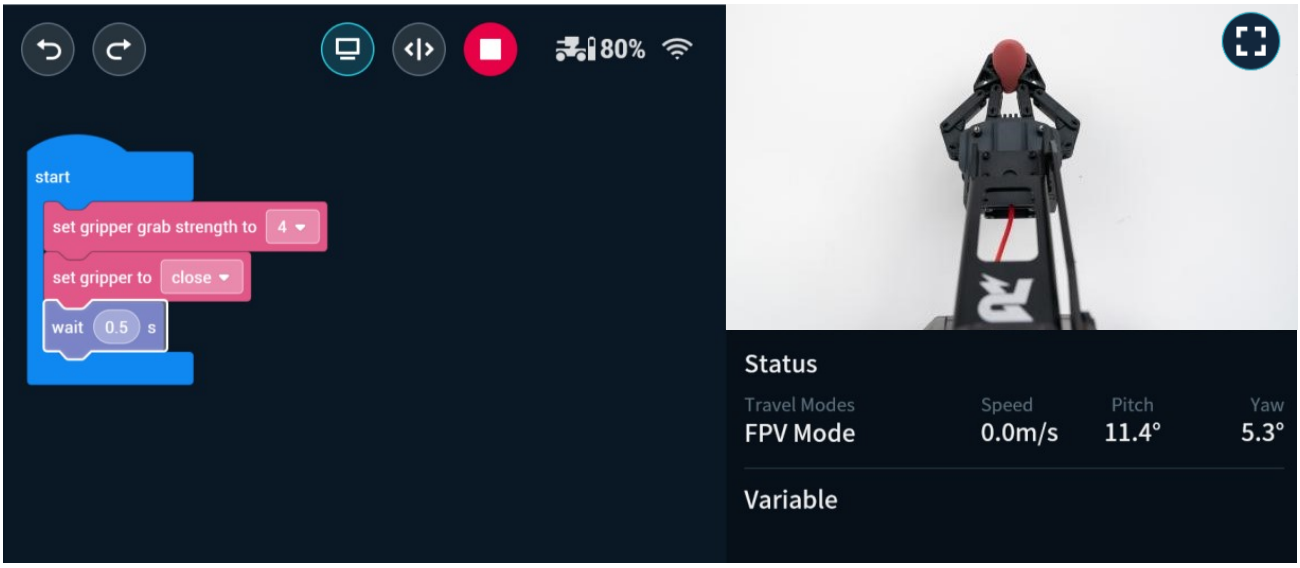
Before the gripper operates, the object maintains its shape:



When the grip force level is set at level 1, there was a slight change in the shape of the object:



When the grip force level is set at level 4, there was obvious change in the shape of the object:

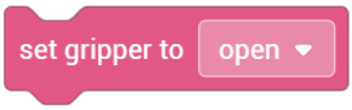


Note:

1) The higher the force level, the greater the grip force, and the faster the operation speed.

2) The force required for the gripper to grip different objects varies. For example, the force level is recommended to be set at a lower level for gripping egg shells and at a higher level for gripping a cup. Set the force level based on your actual requirement and desired effect.

2. control gripper (open)

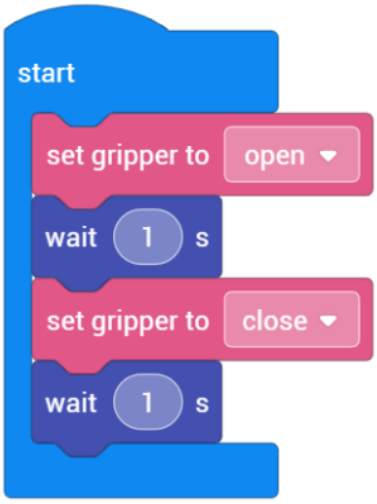


(1) Description: Control Gripper to Open, Close, or Not Move.

(2) Type: Execution

(3) Example: Flexible gripper movement

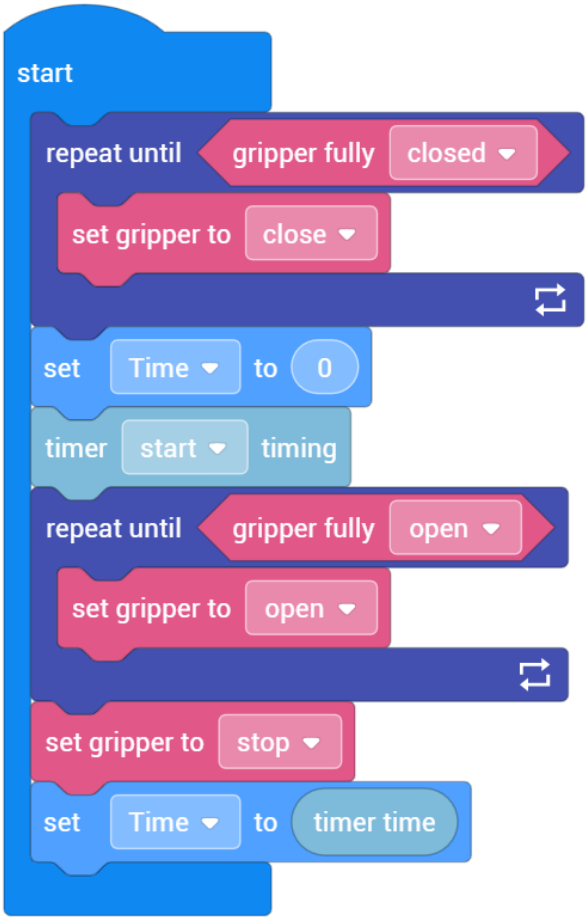
Control the gripper to first open then close.



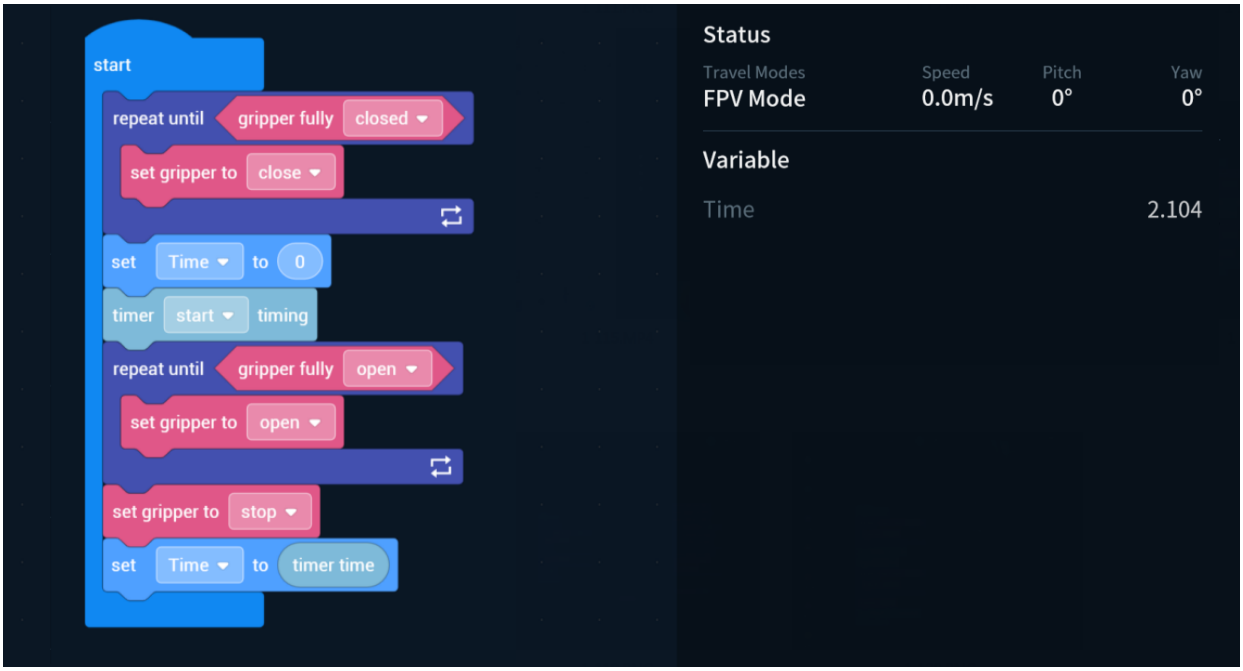
Note:
1) Gripper control (Open, Close) is a continuous movement; therefore it must be used with time control modules such as "Wait for (1)s".
2) The gripper remains still after it reaches its limit.

3.the gripper has fully (opened)

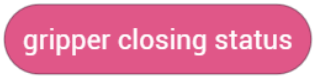
(1) Description: Returns "True" when the current gripper status meets the condition; otherwise, returns "False".
(2) Return value: Boolean
(3) Example: Gripper opening duration.
Learn the time taken for the gripper to fully open from being fully closed.



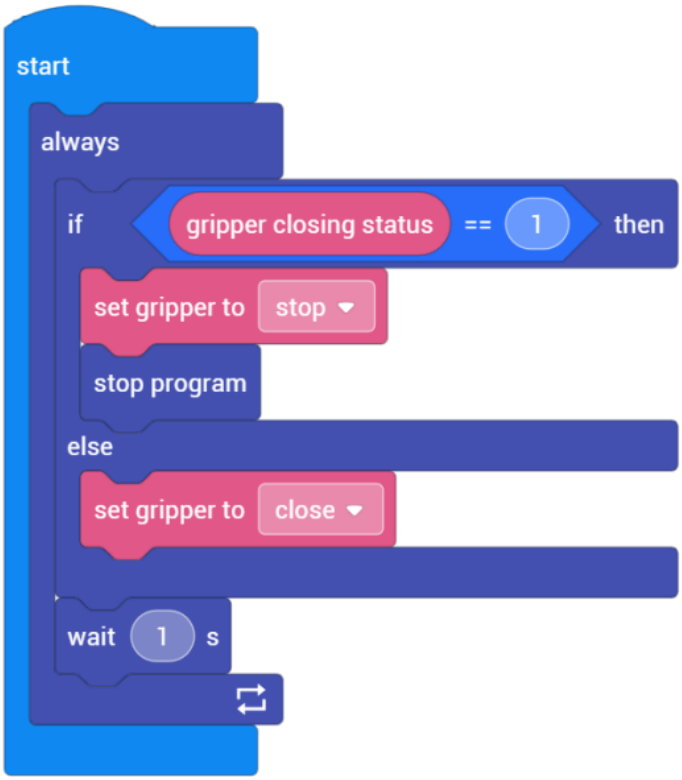
It can be observed from the FPV window that it takes about 2.1 seconds.



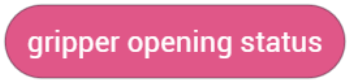
4. gripper's close state



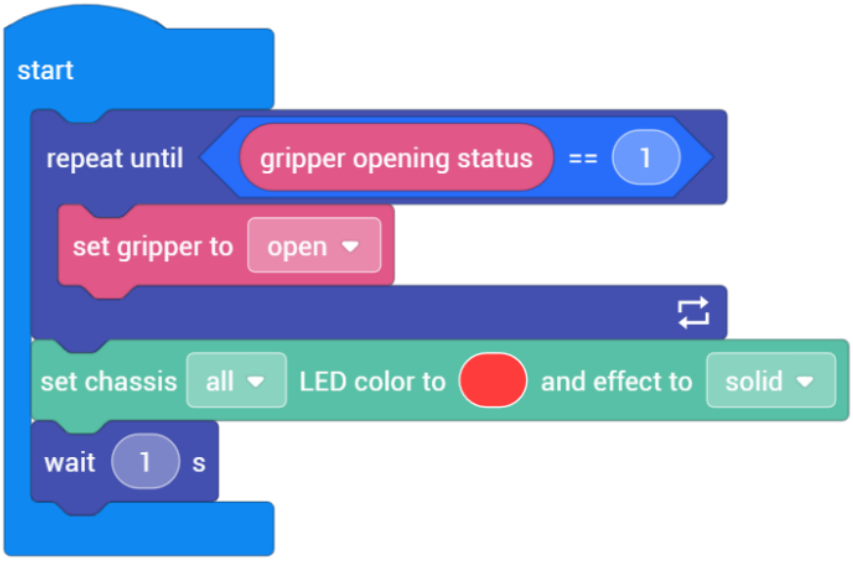
(1) Description: Learn whether the gripper is fully closed. If the return value is 1, the gripper is fully closed; if the value is 0, the gripper is not fully closed.
(2) Type: Information
(3) Example: Gripper closes
Control the gripper's movement according to its current state until it is fully closed.



5. gripper's open state



(1) Description: Learn whether the gripper is fully opened. If the return value is 1, the gripper is fully opened; if the value is 0, the gripper is not fully opened.
(2) Type: Information
(3) Example: Gripper opens
Control the gripper's movement until it's fully opened. When the gripper is fully opened, the chassis LED indicator is solid red for one second.

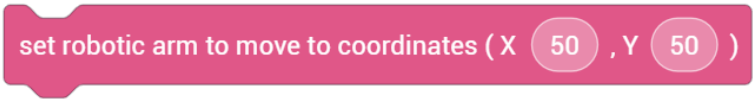


6.control the robotic arm to move (50) millimeters (forward)

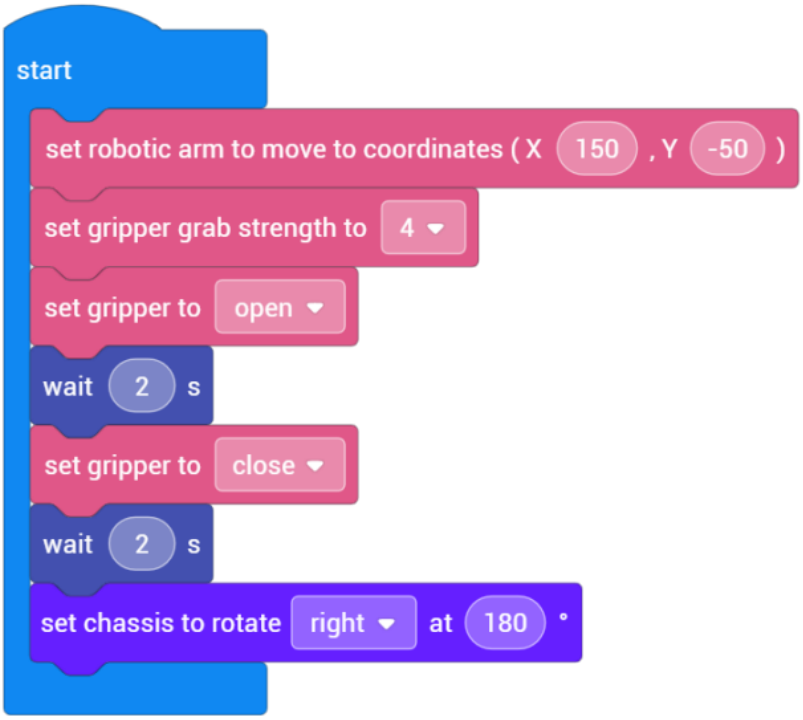
(1) Description: Using the robot body coordinate system as the reference frame, and control the robotic arm to move in a specified direction for a set distance.
(2) Type: Execution
(3) Example: Robotic arm dance
Control the robotic arm to move forward, backward, upward, and downward.

Note:
1) The robotic arm of this robot is a parallel robotic arm. It has a more complex structure which comes with a high load-bearing capacity, rigidity, and precision. It is driven by two servos installed at the bottom of the robotic arm.
2) The end executor of the robotic arm is the gripper. By controlling the connecting rod of the robotic arm, the gripper moves forward and backward along the direction the robot's heading, and upward and downward vertically along the robot body, thus flexibly and accurately delivering the gripper to the specified position.

7. control and move robotic arm to coordinate (X (50), Y (50))

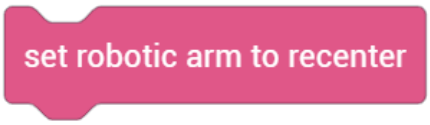


(1) Description: Control and move the robotic arm to a specified position. The front and back direction along the body is X while the vertical direction along the body is Y. The unit is millimeter.
(2) Type: Execution
(3) Example: Grasp objects
Place a water bottle in front of the robot and control the robotic arm so that it can accurately reach the target position. After that, use the gripper to grasp the bottle then finally turn the robot around.

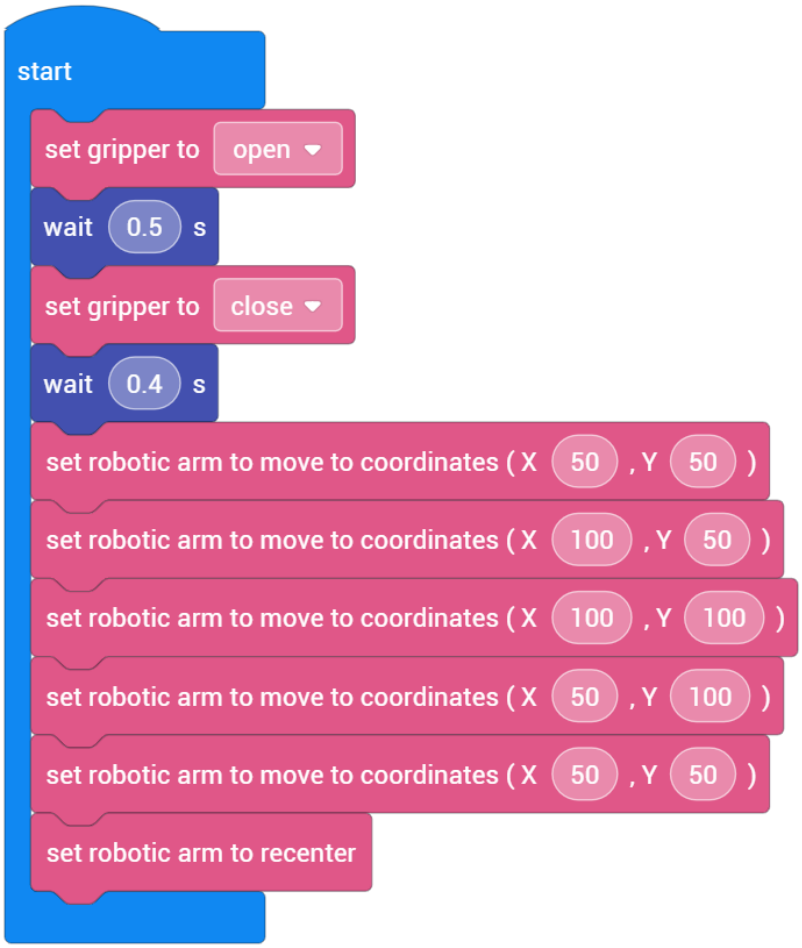


Note:
1) Mark the origin when connecting the robotic arm and proceed according to the instructions in the manual.
2) Flexibly control the robotic arm, the gripper, and the mobile chassis so that the robot can grasp, lift, transfer, and deliver objects.

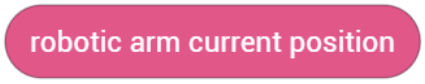
8. control robotic arm to return to original position



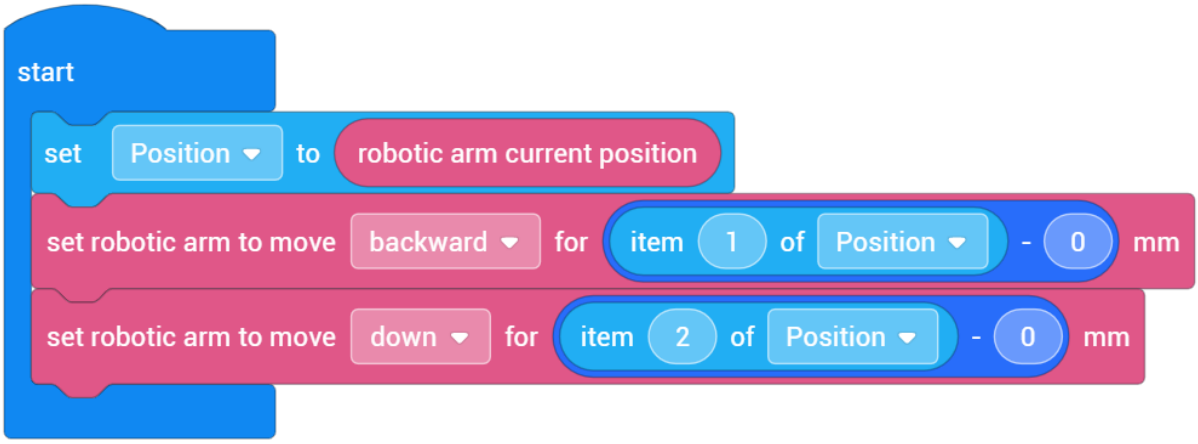
(1) Description: Control the robotic arm so that it can return to the original position, which is (0, 0) of the body coordinate system.
(2) Type: Execution
(3) Example: Draw a square
Control the gripper to grip a writing brush and draw a square by moving the robotic arm and then return to the original position.



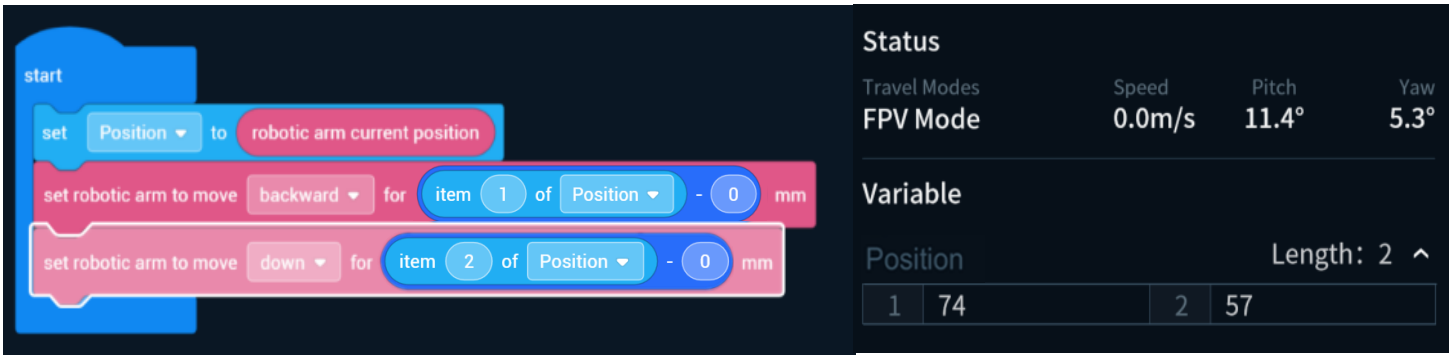
9. Current Position of Robotic Arm



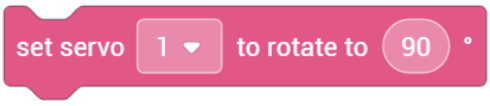
(1) Description: Obtain the current position of the robotic arm. The position parameter is (x(abscissa), y(ordinate)). The unit is millimeter.
(2) Type: Execution
(3) Example: Control the robotic arm to return it to the original position
Control the robotic arm to return to the original position through calculating the difference between the current coordinate and the origin.



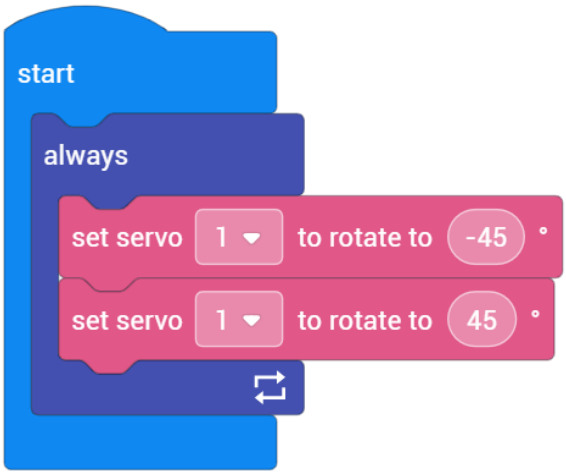
It can be observed through the FPV window:



10. control servo no.(1) to rotate to (90)°

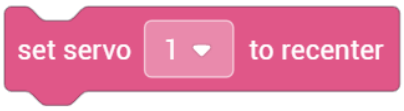


(1) Description: Control a specified servo (No.) to rotate to a set angle. If the set angle is a positive value, the servo rotates clockwise; if the set angle is a negative value, the servo rotates counterclockwise.
(2) Type: Execution
(3) Example: Pendulum
Make a pendulum using a servo, a servo gear, and adhesive tape, then control the servo so that it rotates back and forth between 45° and -45°.

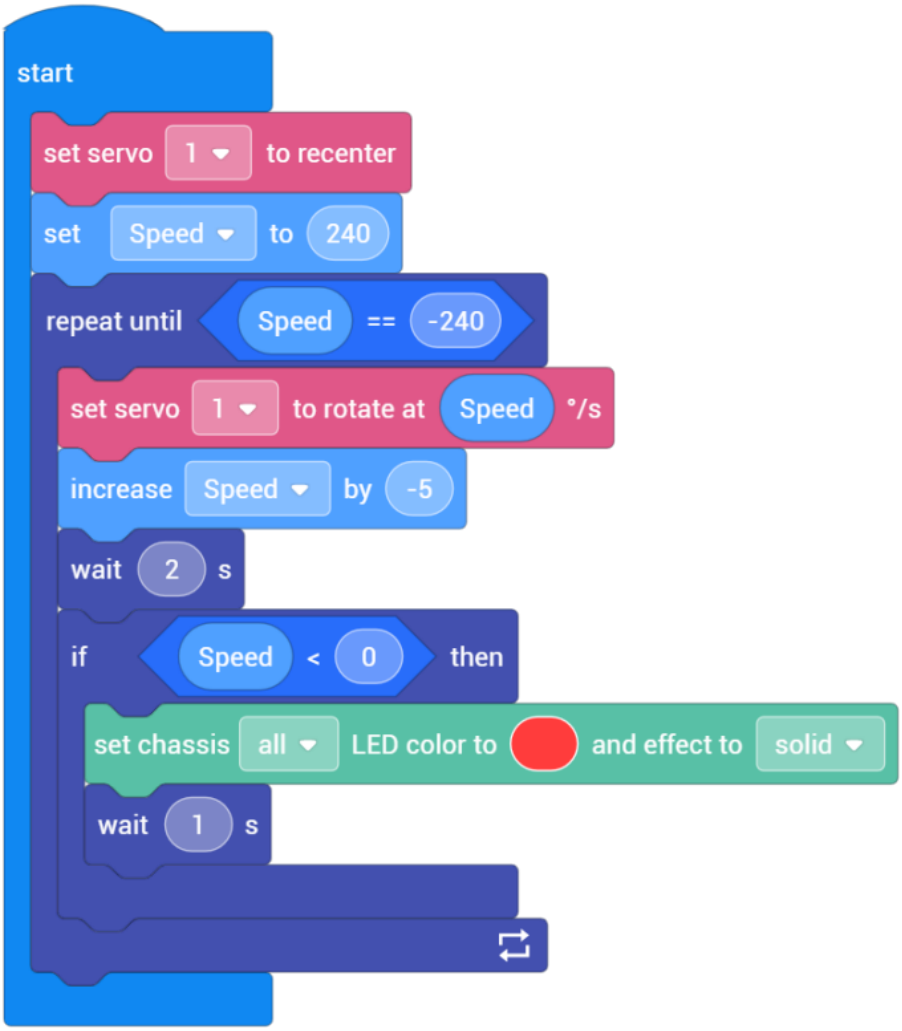


Note:
1) The servo supports two kinds of control: angle control and speed control. This module is applicable to situations with a high requirement for the rotation angle control of the servo.
2) When the servo is installed on the robotic arm, using servo-related modules in the "extension modules" to control the servo is not supported.

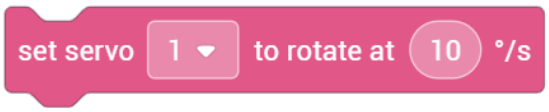
11. control servo no.(1) to return to its original position



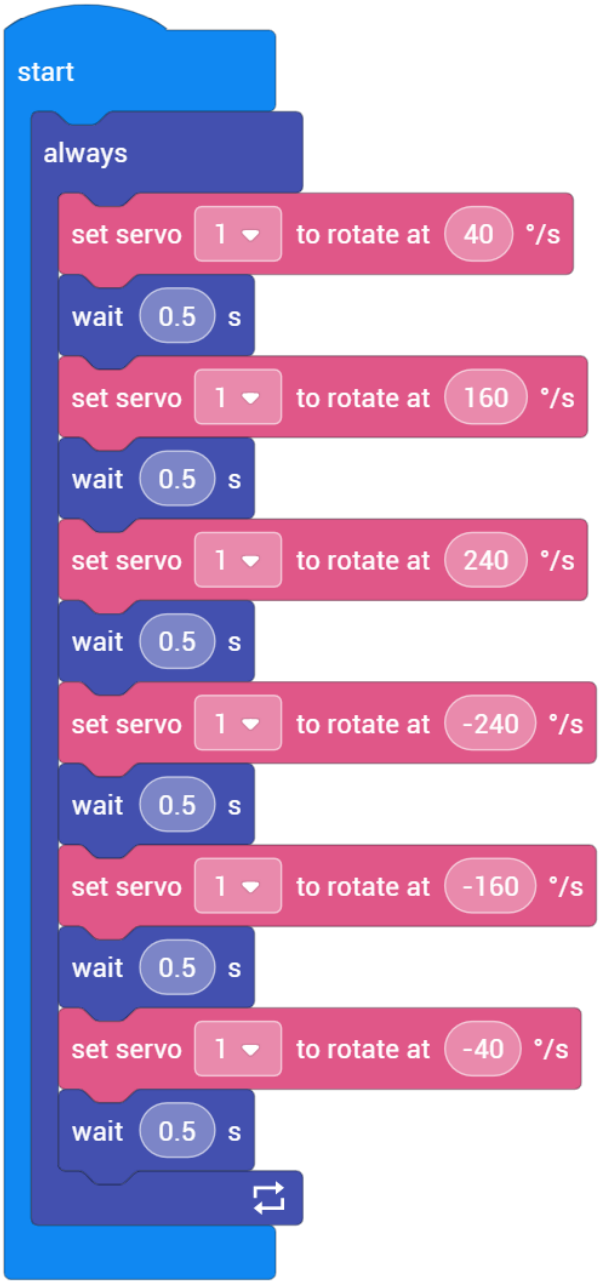
(1) Description: Control a specified servo (No.) to return to its original position.
(2) Type: Execution
(3) Example: Check servo performance
To check the performance of servo No.(1), control the servo to perform a clockwise rotation at the maximum speed, gradually decelerating to a stop, then control it to rotate counterclockwise, continuously accelerating. When the servo rotation direction is switched from clockwise to counterclockwise, the chassis LED indicator is a solid red.



12. rotate servo no.(1) at a speed of (10)° per second



(1) Description: Control a specified servo (No.) to rotate at a set speed. If the speed is a positive value, the servo rotates clockwise; if the speed is a negative value, the servo rotates counterclockwise.
(2) Type: Execution
(3) Example: The servo speed changes along with the direction.
Control the servo to accelerate while rotating clockwise and to decelerate while rotating counterclockwise.

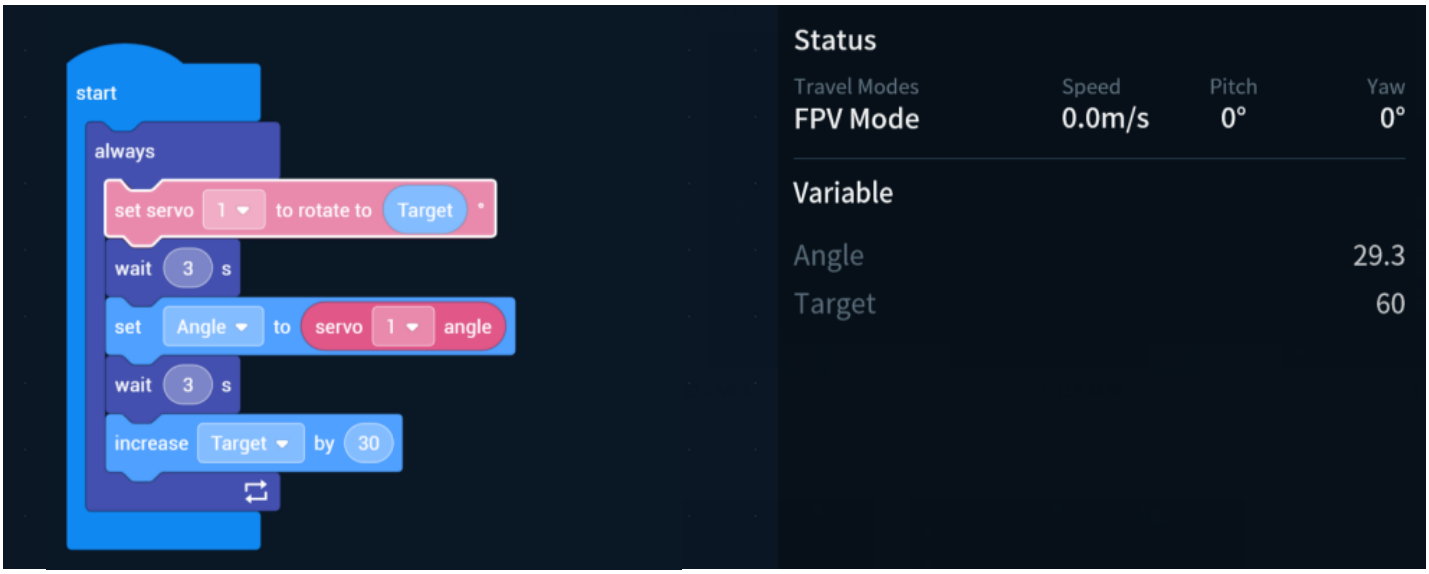


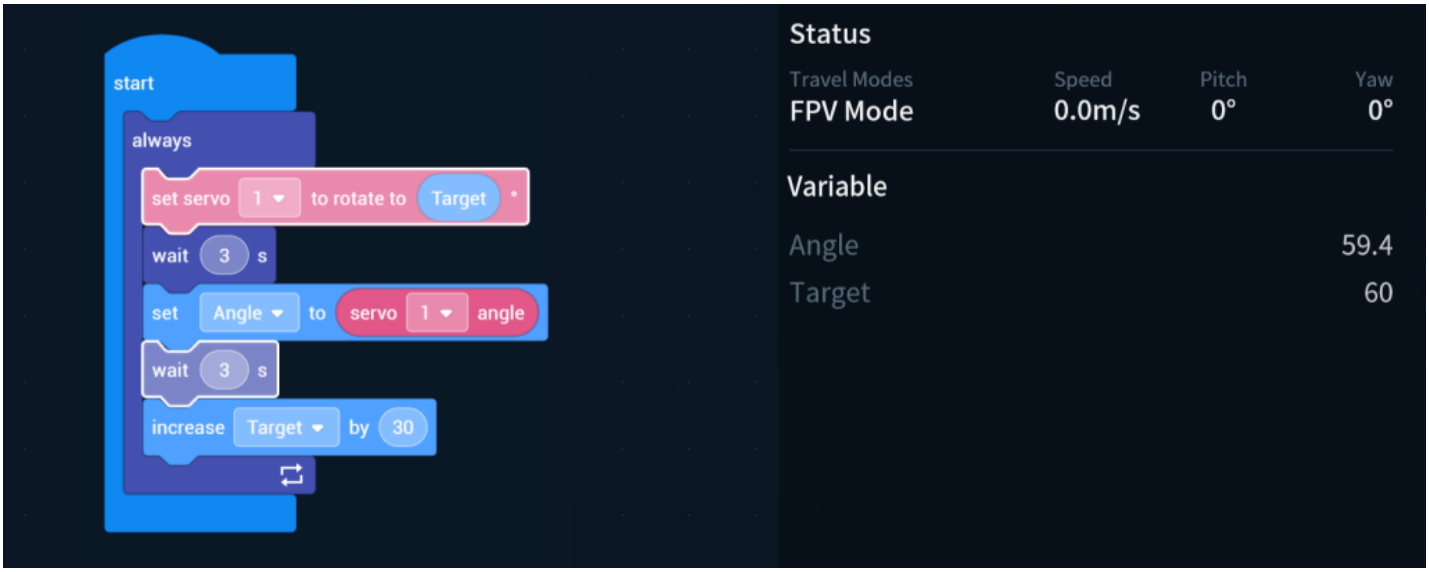
Note:
The servo supports two kinds of control: angle control and speed control. This module is applicable to cases with a high requirement for speed control of the servo, which will continuously rotate without any positioning limitations.

13. the angle of servo no.(1)

(1) Description: Obtain the angle of a specified servo (No.).
(2) Type: Information
(3) Example: Servo angle
Obtain the changes of the servo angle as the servo rotates continuously.

It can be observed through the FPV window.

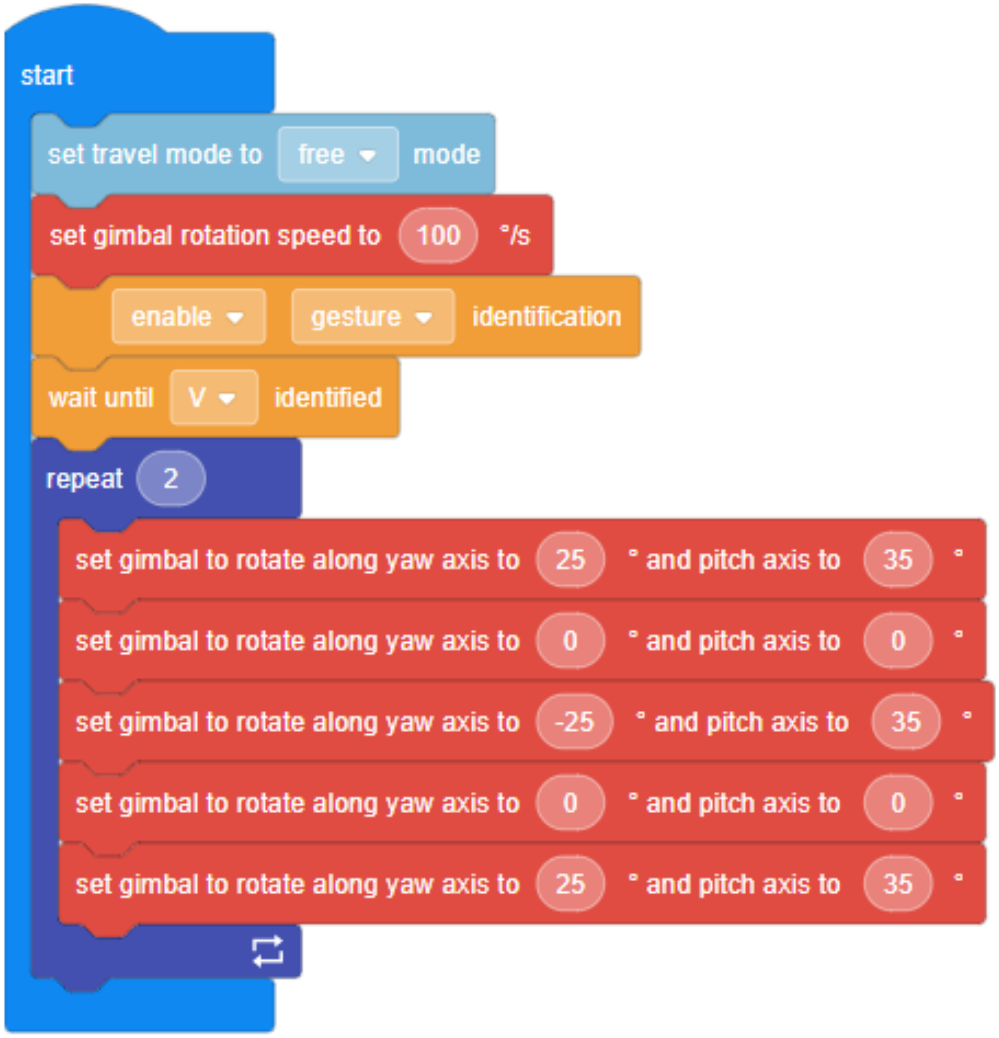



start
always
set servo 1 to rotate to Target °
wait 3 s
set Angle to servo 1 angle
wait 3 s
increase Target by 30
Status
Travel Modes
FPV Mode
Speed
0.0m/s
Pitch
0°
Yaw
0°
Variable
Angle
59.4
Target
60

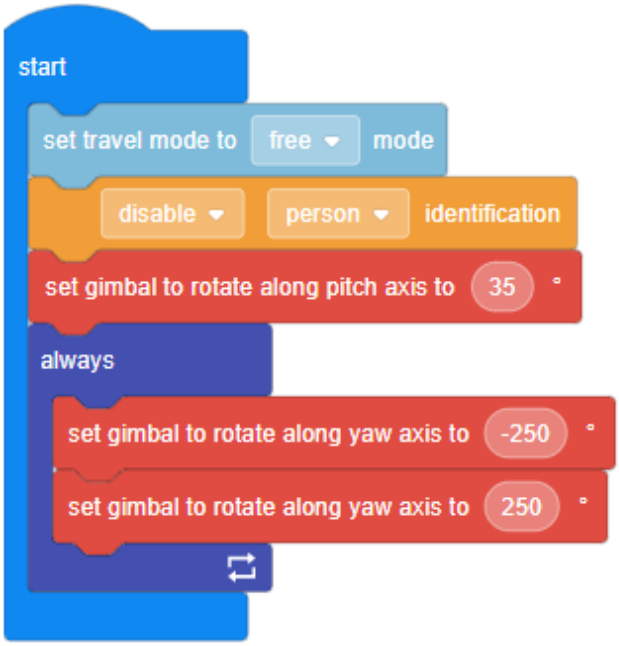
Smart

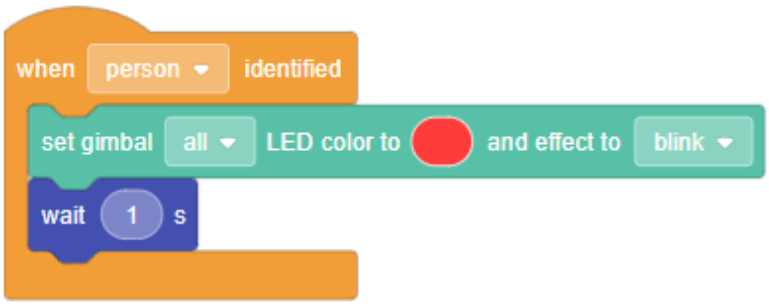
1. (enable) (vision marker) identification

(1) Objective: Enables or disables the EP's visual identification functions for vision marker, person, and other EP robot
(2) Type: Settings block
(3) Example: Imitate V-signal hand gesture
Sets the gimbal to translate in a V shape when the robot identifies the V-sign hand gesture.

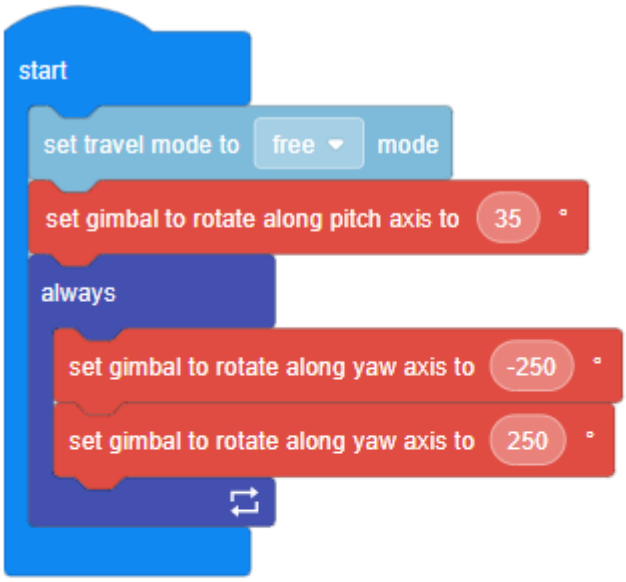


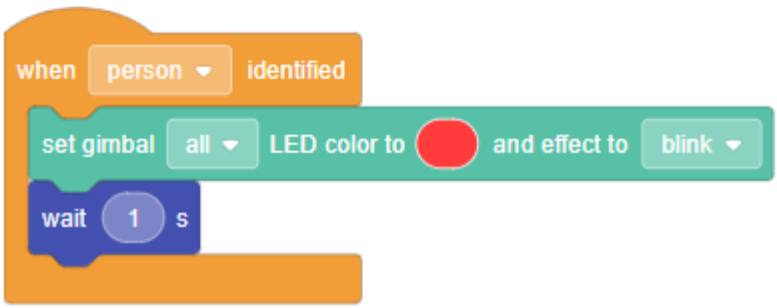
Note:
The robot's smart identification function is disabled by default. You must enable the identification function for the robot to be able to respond to identifiable objects. For example,


start
set travel mode to free mode
disable person identification
set gimbal to rotate along pitch axis to 35 °
always
set gimbal to rotate along yaw axis to -250 °
set gimbal to rotate along yaw axis to 250 °


when person identified
set gimbal all LED color to and effect to blink
wait 1 s

or


start
set travel mode to free mode
set gimbal to rotate along pitch axis to 35 °
always
set gimbal to rotate along yaw axis to -250 °
set gimbal to rotate along yaw axis to 250 °


when person identified
set gimbal all LED color to and effect to blink
wait 1 s

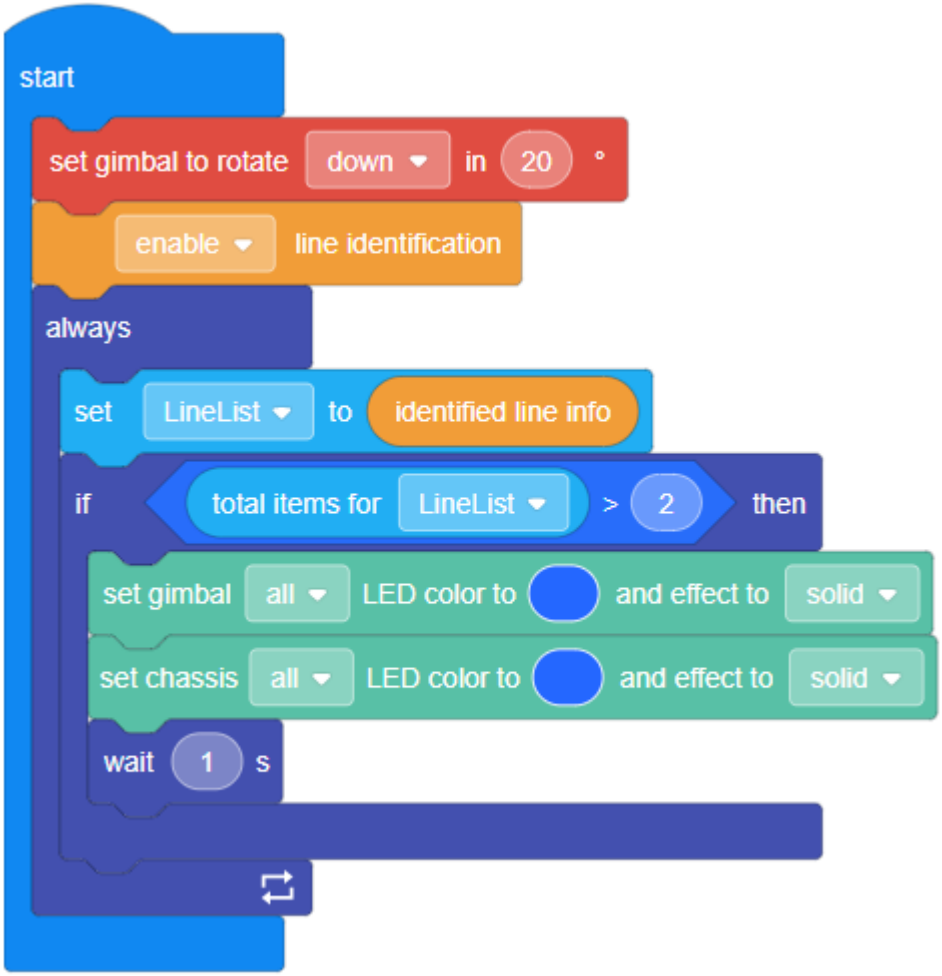
In the above two example programs, robot will be unable to identify people, meaning the gimbal's LED indicator will not blink.

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. (enable) line identification

(1) Objective: Enables/disables line identification
(2) Type: Settings block
(3) Example: Identify blue lines
When the robot identifies blue lines, all blue LED chassis and gimbal indicators will blink.



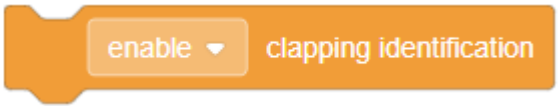
Note:
1) The line identification function is disabled by default. You must enable line identification for the robot to

be able to respond to line inspection commands.
2) The default line color that the robot is set to identify is blue.

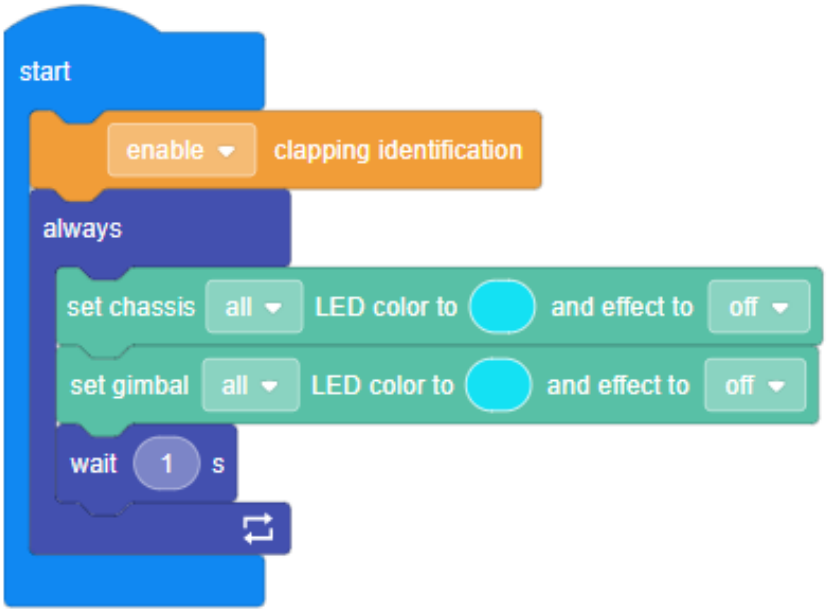
Python API：
Function: vision_ctrl.enable_detection(function_enum)
vision_ctrl.disable_detection(function_enum)
Parameters:

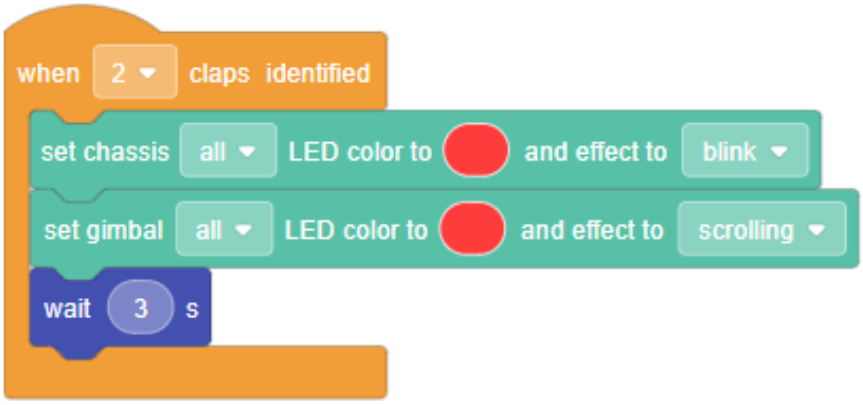
- function_enum(enum):
  - rm_define.vision_detection_line

3. (enable) clapping identification



(1) Objective: Enables/disables clapping identification
(2) Type: Settings block
(3) Example: Wake up the robot by clapping
Initially, the LED chassis and gimbal indicators are turned off. When the robot identifies a double clap, all red LED chassis and gimbal indicators will blink and the red LED gimbal indicators will scroll.





Note:
Clapping identification is disabled by default and must be enabled for the robot to be able to respond to any

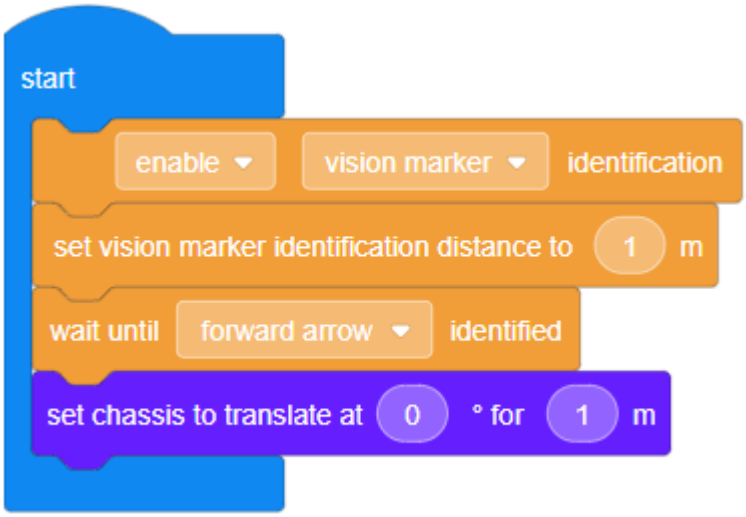
clapping.

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. set vision marker identification distance to (1) m

(1) Objective: Sets the maximum Vision Marker identification distance for the gimbal; beyond this distance, the robot will be unable to recognize Vision Markers.
(2) Type: Settings block
(3) Example: Set identification distance
When the robot identifies a forward arrow it will translate forward by 1 meter.



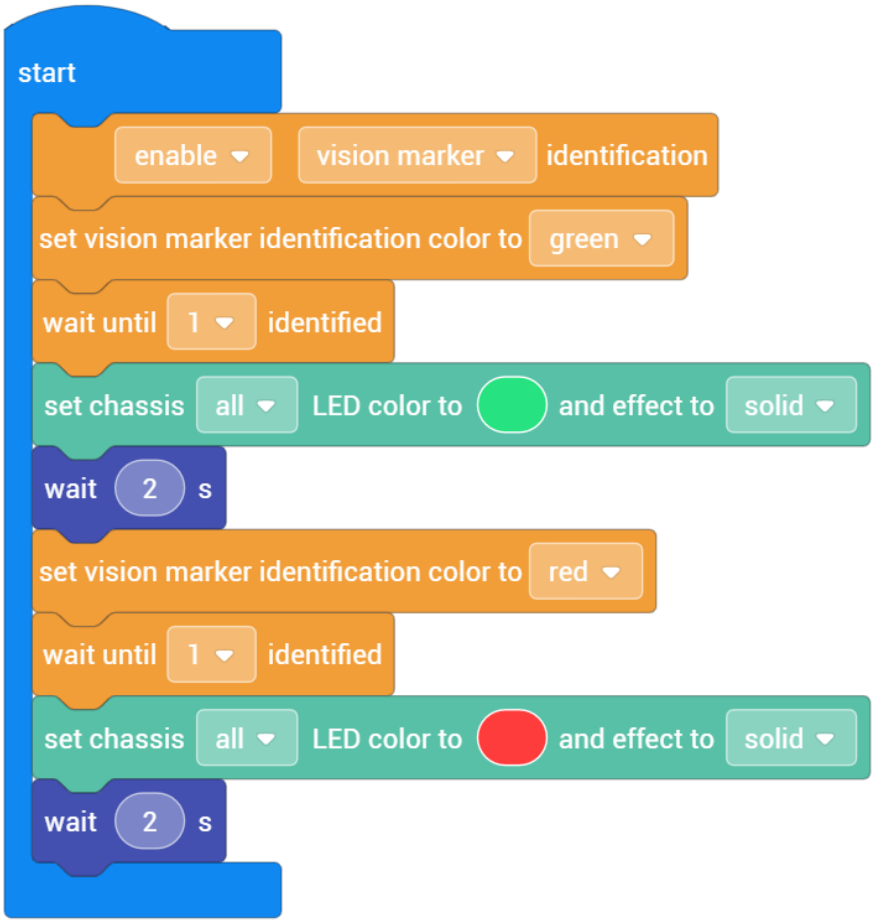
Note:
1) If the Vision Marker is placed more than 1 meter away from the robot (for example 1.5 or 2 meters away from the robot), the robot will not be able to identify it.
2) The identification distance only applies when using Vision Markers of the official standard size. The effective identification distance may vary if you print your own Vision Markers in non-standard sizes.

Python API:
Function: vision_ctrl.set_marker_detection_distance(distance)
Parameters:

- distance(float): [0.5, 3]

5. set the visual tag identification color as (red)

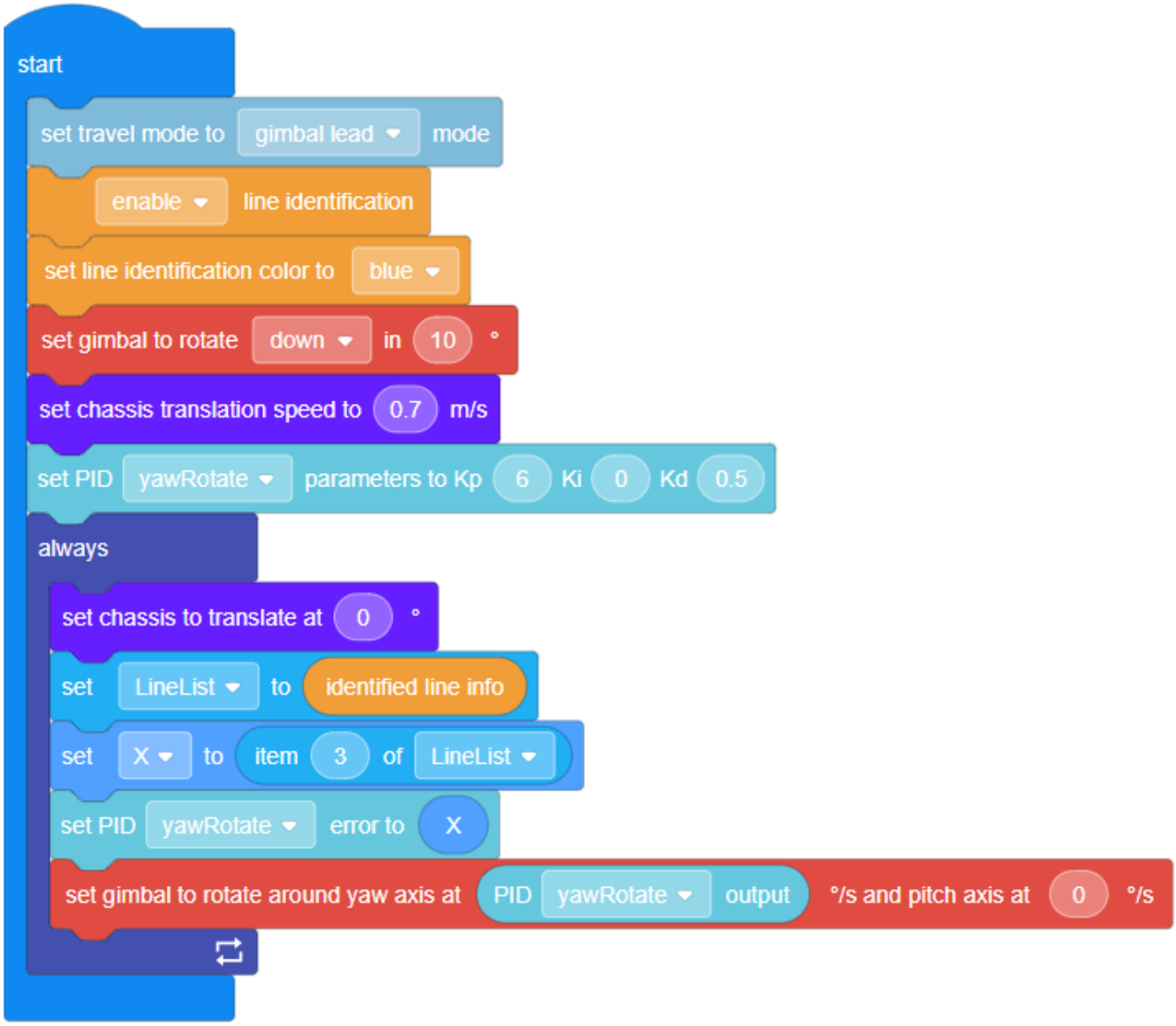
(1) Description: Set visual tag colors that the robot can identify.
(2) Type: Setting
(3) Example: Switch different visual tag identification colors.
After the robot identifies the red tag number 1, the chassis LED indicator stays solid red; after it identifies the green label number 1, the chassis LED indicator stays solid green.



Note:
The default identifiable visual tag color is red
.

6. set line identification color to (blue)

(1) Objective: Sets the specific line color the robot will be able to identify
(2) Type: Settings block
(3) Example: Identify blue lines
When blue tape is laid on the ground, the robot will translate along the blue lines.



Note:
The default line color that the robot can identify is blue.

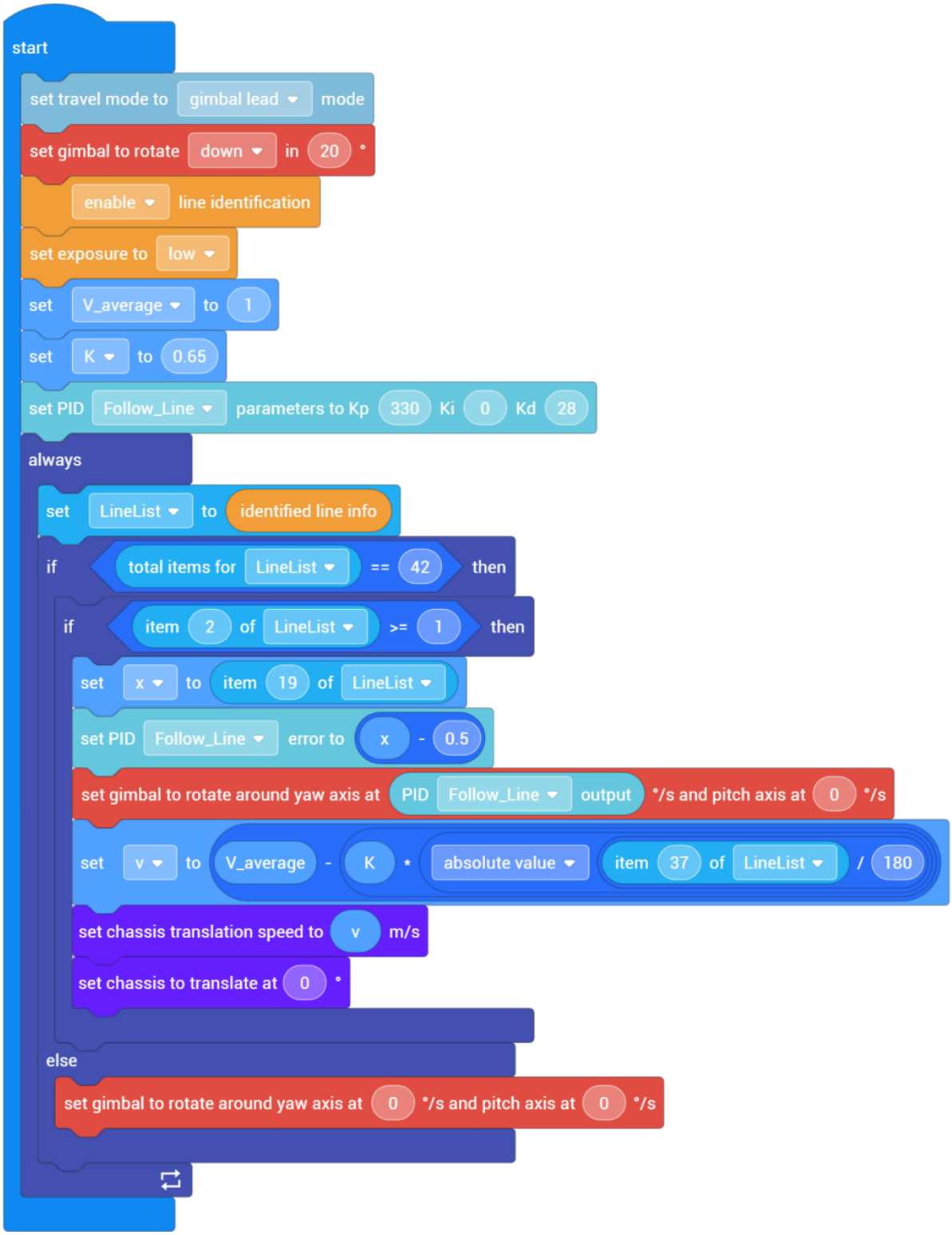
Python API:
Function: vision_ctrl.line_follow_color_set(color_enum)
Parameters:

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

7. exposure value to (high)

(1) Objective: Reduces the exposure in Line Follow Mode to prevent blurriness caused by fast turns, ensuring effective and accurate visual identification

(2) Type: Settings block
(3) Example: Identify lines

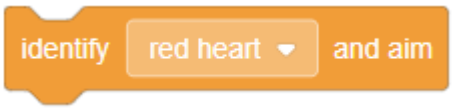


Python API:
Function: media_ctrl.exposure_value_update(exposure_value_enum)
Parameters:

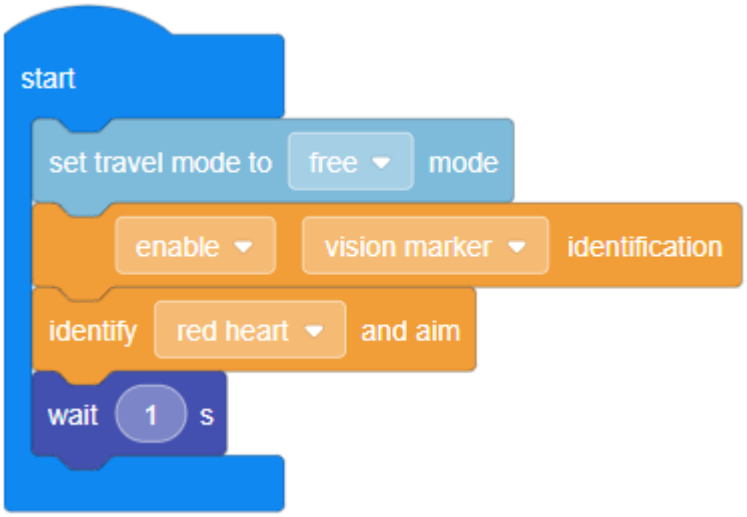
- exposure_value_enum(enum):
  - rm_define.exposure_value_large
  - rm_define.exposure_value_medium

■ rm_define.exposure_value_small

8. identify (red heart) and aim



(1) Objective: Sets the robot to identify and aim at the central position of the corresponding Vision Markers
(2) Type: Execution block
(3) Example: Set robot to aim at the red heart



Note:
1)You must enable Vision Marker identification for the robot to be able to identify Vision Markers.
2)In this block, when the robot identifies a red heart, it will automatically aim toward it; if the robot fails to identify the red heart within five seconds, it will exit the program and run the next program.

Python API：
Function: vision_ctrl.detect_marker_and_aim(marker_enum)
Parameters:

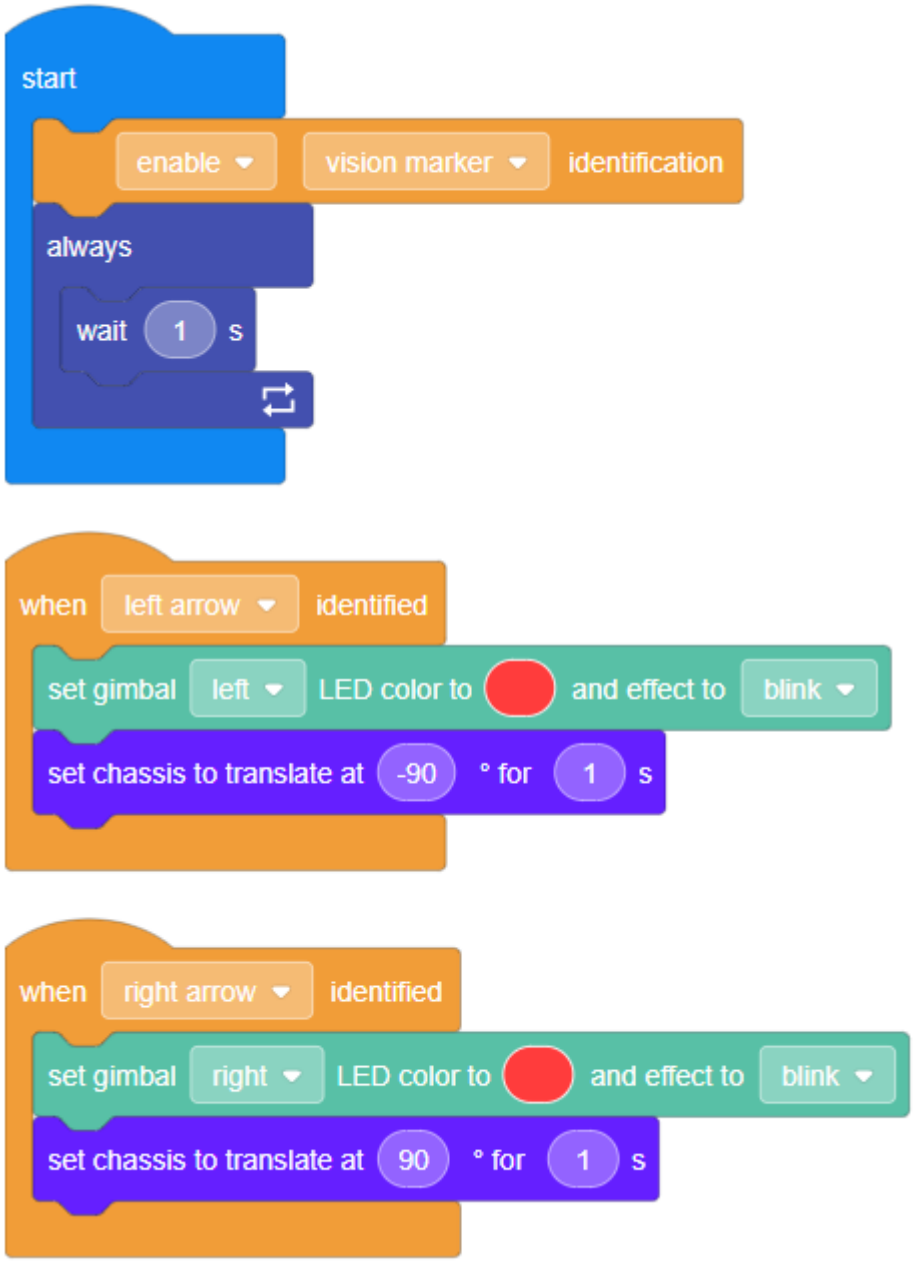
- rm_define.marker_trans_red_heart
- rm_define.marker_trans_target
- rm_define.marker_trans_dice
- rm_define.marker_number_[zero,..., nine]
- rm_define.marker_letter_[A,..., Z]

9. when (person) identified

(1) Objective: Sets the corresponding block to run its internal program when specified information is identified
(2) Type: Event block
(3) Examples: Indicate turn, Control attitude

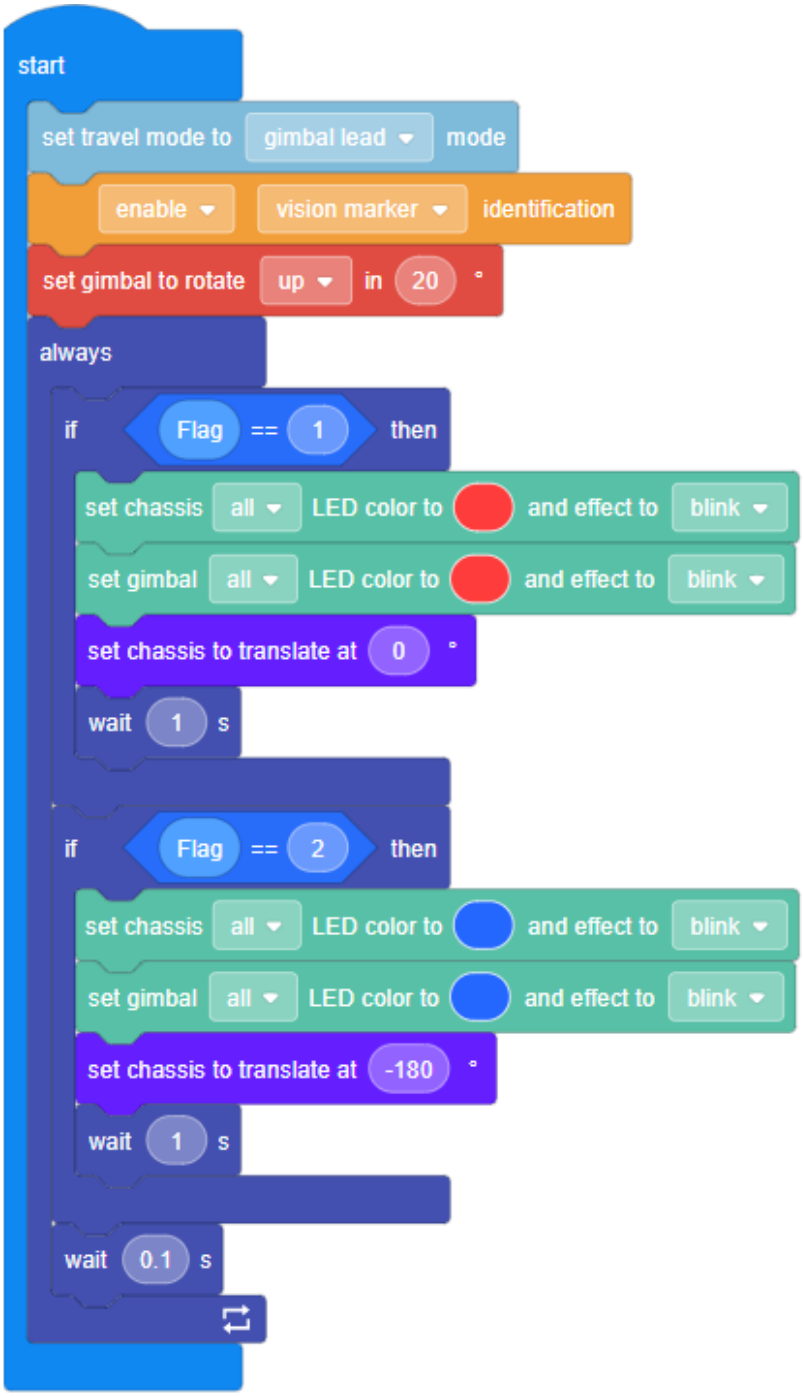
① Indicate turn

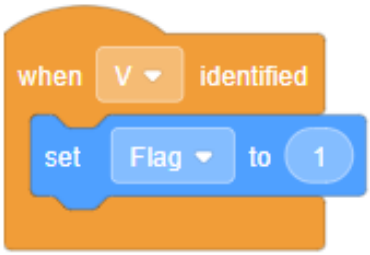
When the robot identifies a left arrow, the left turn LED indicators on the gimbal will blink and the robot will translate left for 1 second; when the robot identifies a right arrow, the right turn LED indicators on the gimbal will blink and the chassis will translate right for 1 second.

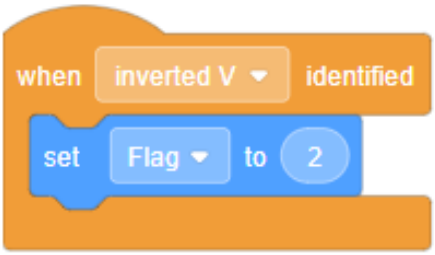


② Control attitude

A person can stand at a distance of 1 meter from the robot and control its movements using gestures. When the person makes a V sign, the robot will translate forward; when the person makes an inverted V sign, the robot will translate backward.







Note:

1) The Event is of high priority, which means that no matter where the main function runs, the main function will be suspended and the program in the Event will start running once the triggering conditions are fulfilled.

2) To have the robot identify a person, rotate the gimbal slightly upward, have the person stand at a distance of 1 meter, and ensure that the person stands upright within the robot's field of view.

3) The person signaling to the robot must make the V sign and the inverted V sign with their arms (as opposed to fingers).

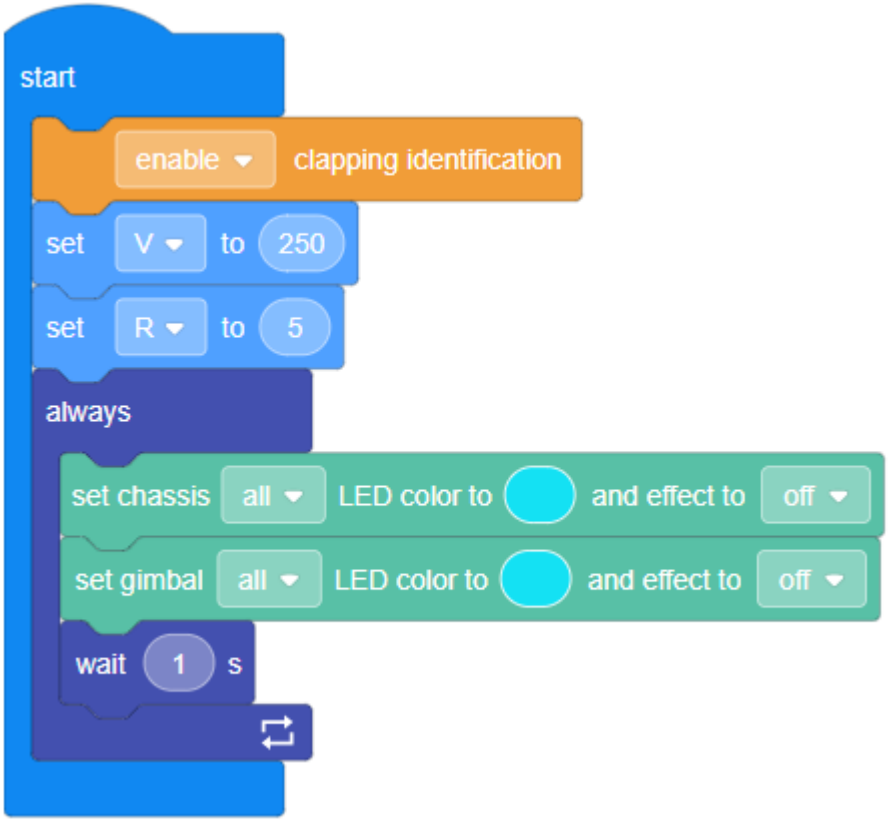
Python API:

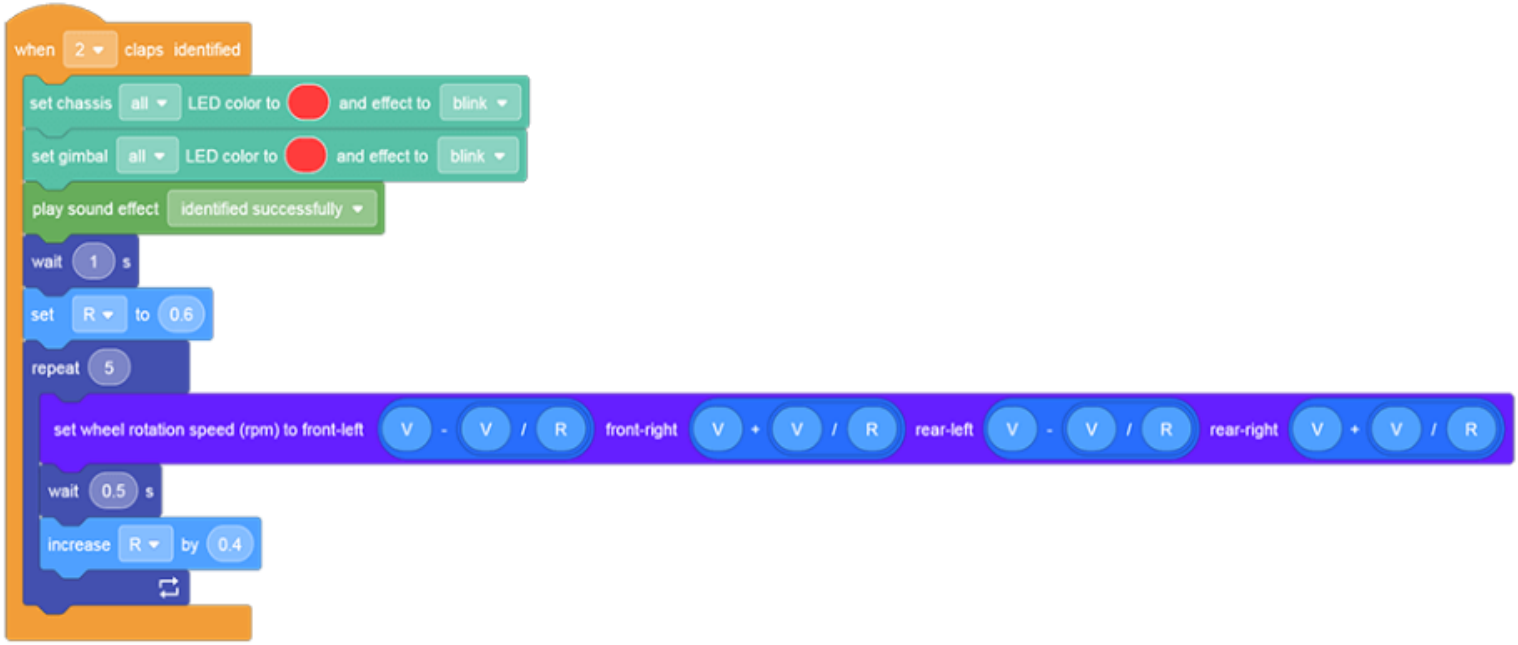
Function: def vision_recognized_people(msg)
def vision_recognized_car(msg)
def vision_recognized_pose_all(msg)
def vision_recognized_pose_victory(msg)
def vision_recognized_pose_give_in(msg)
def vision_recognized_pose_capture(msg)
def vision_recognized_marker_trans_all(msg)
def vision_recognized_marker_trans_left(msg)
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. when (2) claps identified

(1) Objective: Sets the block to run its internal program when a specific clapping pattern is identified
(2) Type: Event block
(3) Example: Translate in a spiral pattern
Initially, all chassis and gimbal LED indicators are turned off. When the robot identifies 2 claps, all red LED indicators will blink and the robot will translate in a spiral line.





Python API:

Function: def sound_recognized_applause_twice(msg)

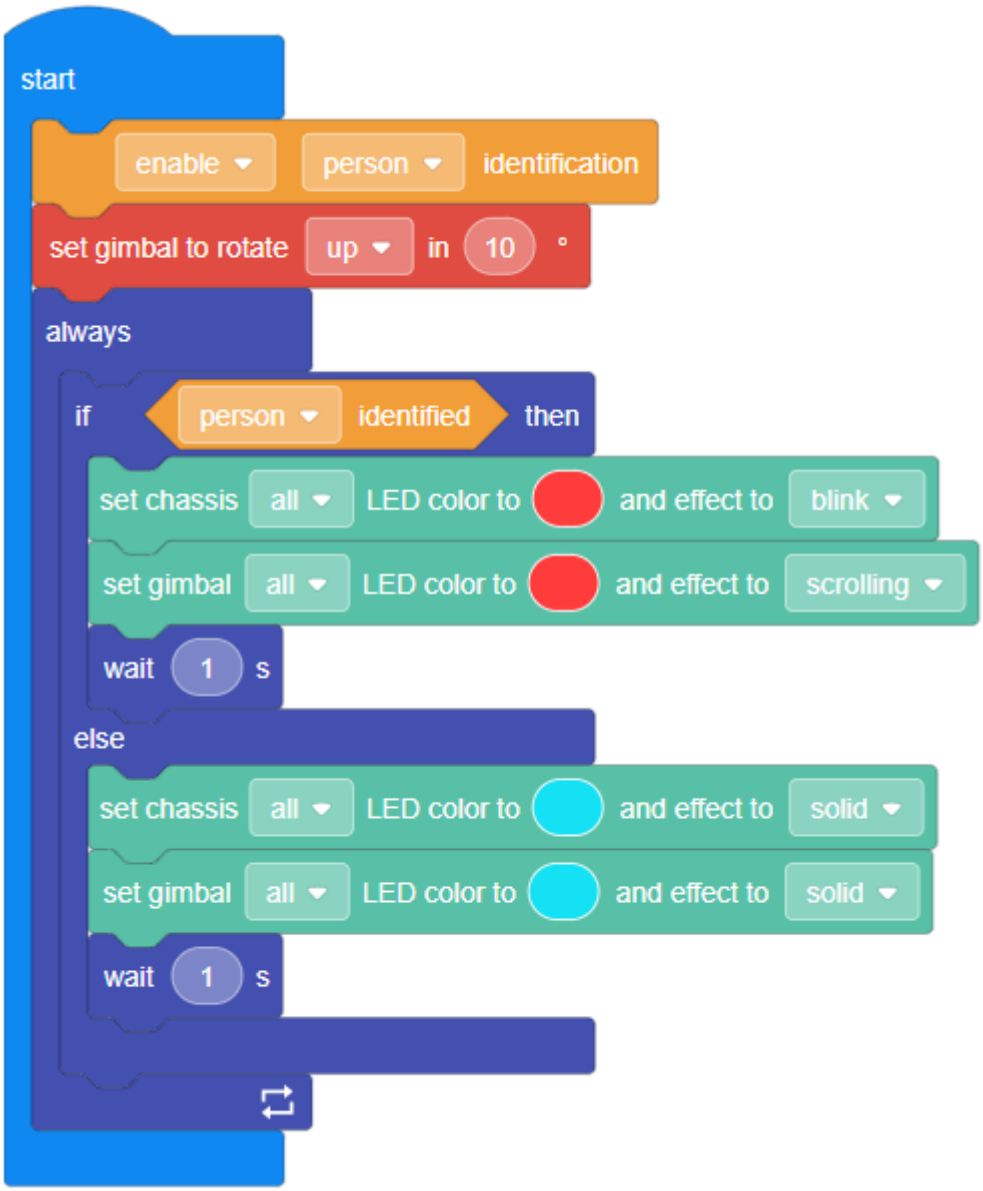
def sound_recognized_applause_thrice(msg)

Type: Event callback

11. (person) identified

(1) Objective: Sets the robot to return the condition “True” when specified information (such as an object,

Vision Marker, gesture, and so on) is identified; otherwise, the condition “False” is returned.
(2) Type: Boolean block
(3) Example: Identify people When the robot identifies a person, all red LED chassis indicators will blink and all red LED gimbal indicators will scroll; otherwise, all LED chassis and gimbal indicators will remain in the default color.



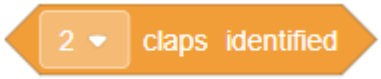
Note:
Try to use this block with conditional statements whenever possible.

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

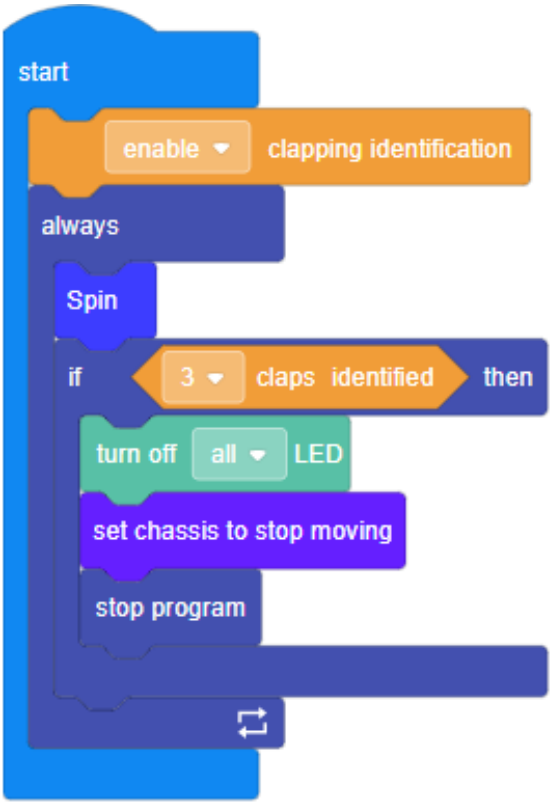
- condition_enum(enum):
  - rm_define.cond_recognized_people
  - rm_define.cond_recognized_car
  - rm_define.cond_recognized_marker_trans_all
  - rm_define.cond_recognized_marker_trans_left
  - rm_define.cond_recognized_marker_trans_right
  - rm_define.cond_recognized_marker_trans_forward
  - rm_define.cond_recognized_marker_trans_stop

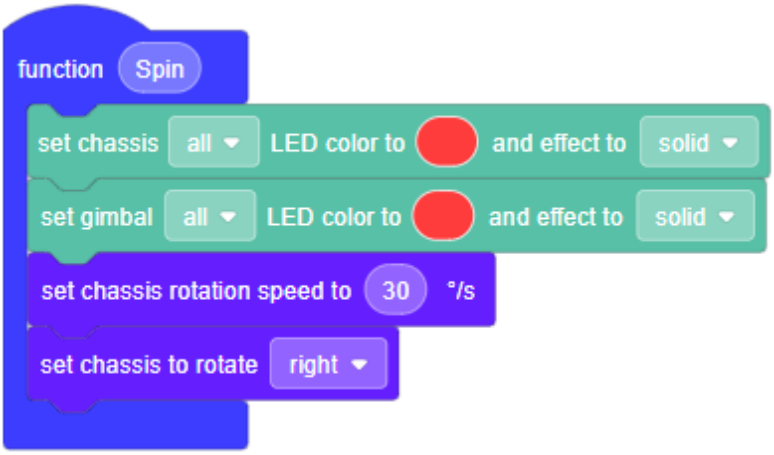
- rm_define.cond_recognized_marker_trans_red_heart
- rm_define.cond_recognized_marker_trans_target
- rm_define.cond_recognized_marker_trans_dice
- rm_define.cond_recognized_marker_number_all
- rm_define.cond_recognized_marker_number_[zero,..., nine]
- rm_define.cond_recognized_marker_letter_all
- rm_define.cond_recognized_marker_letter_[A,..., 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. (2) claps identified



(1) Objective: Sets the robot to return the condition “True” when a specific clapping pattern is identified; otherwise, the condition “False” is returned.
(2) Boolean block
(3) Example: Stop motion with clapping
This sets the robot to turn right and stop all motion when it identifies 3 claps.





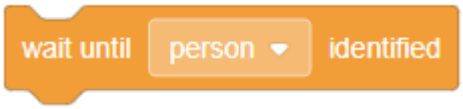
Python API:

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

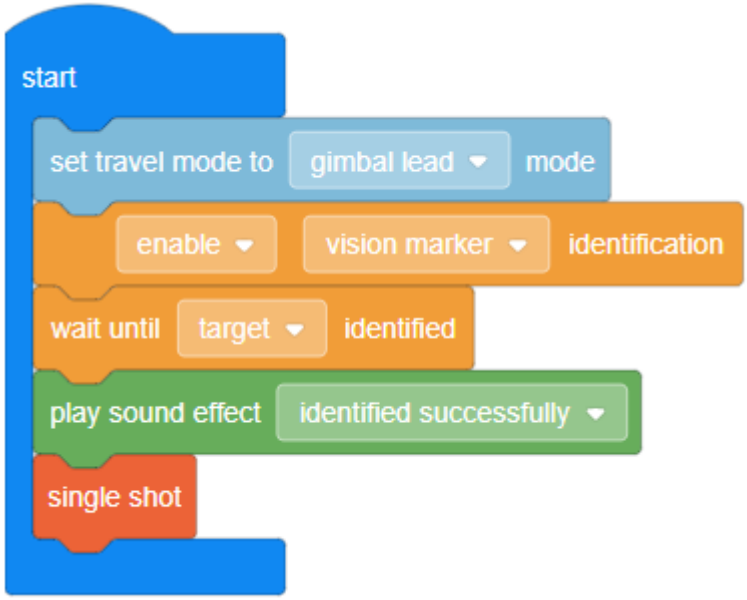
13. wait until (person) identified



(1) Objective: When specific information (such as an object, Vision Marker, gesture, and so on) is identified, the system will continue executing commands; otherwise, it will continue to wait.

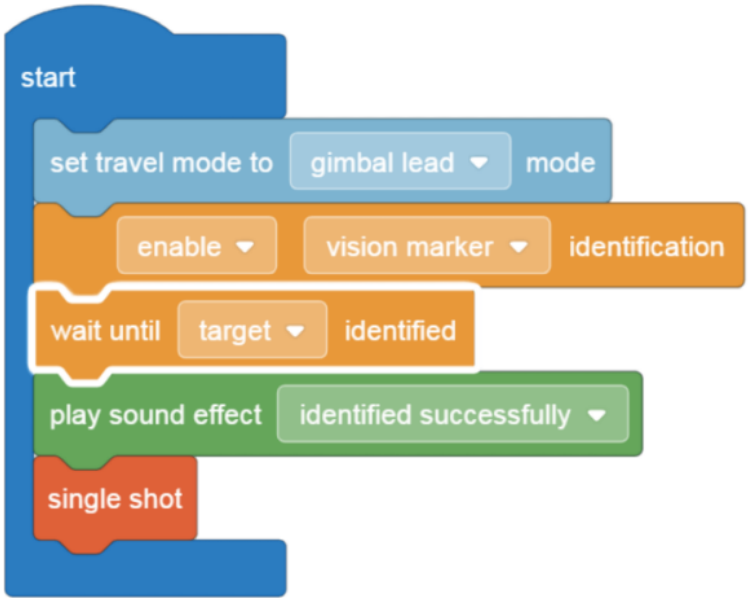
(2) Type: Execution block, Blocking block

(3) Example: Practice shooting



Note:

If the target marker is not identified, the program in this block will continue waiting and the block will be highlighted:



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. wait until (2) claps identified

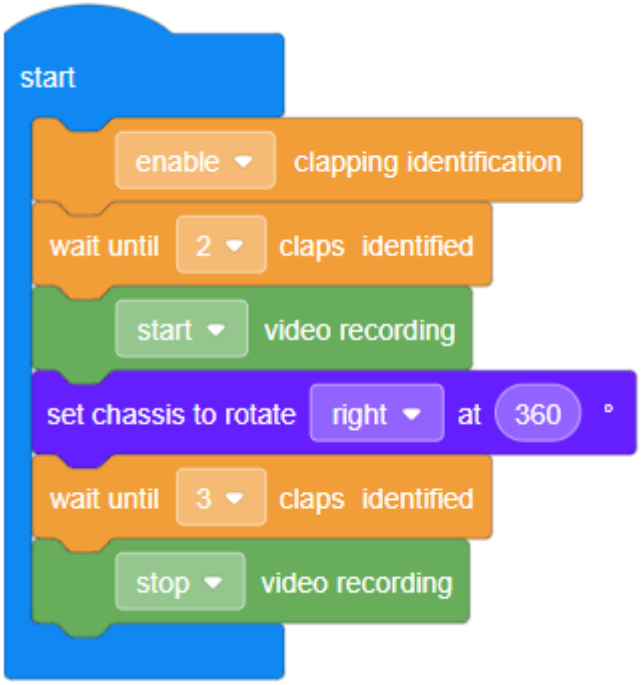
(1) Objective: Sets the system to continue to execute commands when a specific clapping pattern is identified; otherwise, it will continue to wait.

(2) Type: Execution block, Blocking block

(3) Example: Capture videos with clapping control

This sets the robot to start recording videos when it identifies 2 claps and to stop recording when it identifies

3 claps.



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

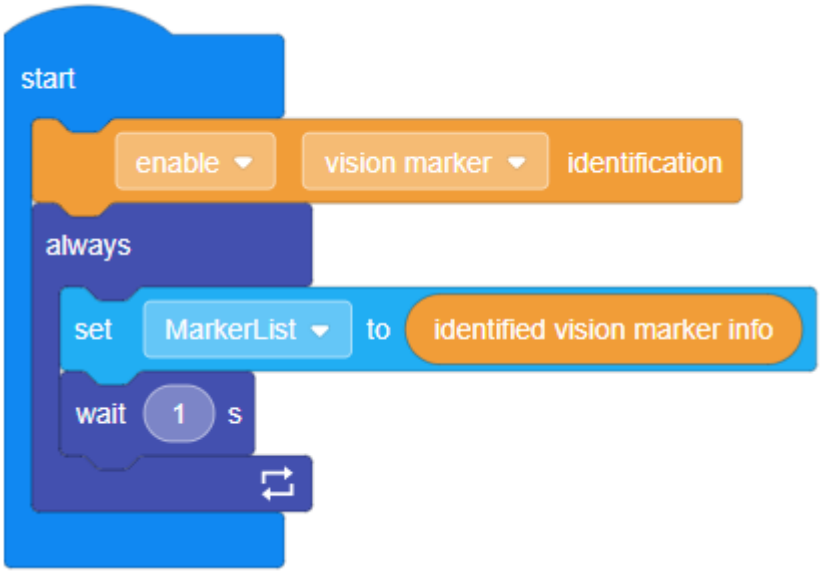
15. identified vision marker info

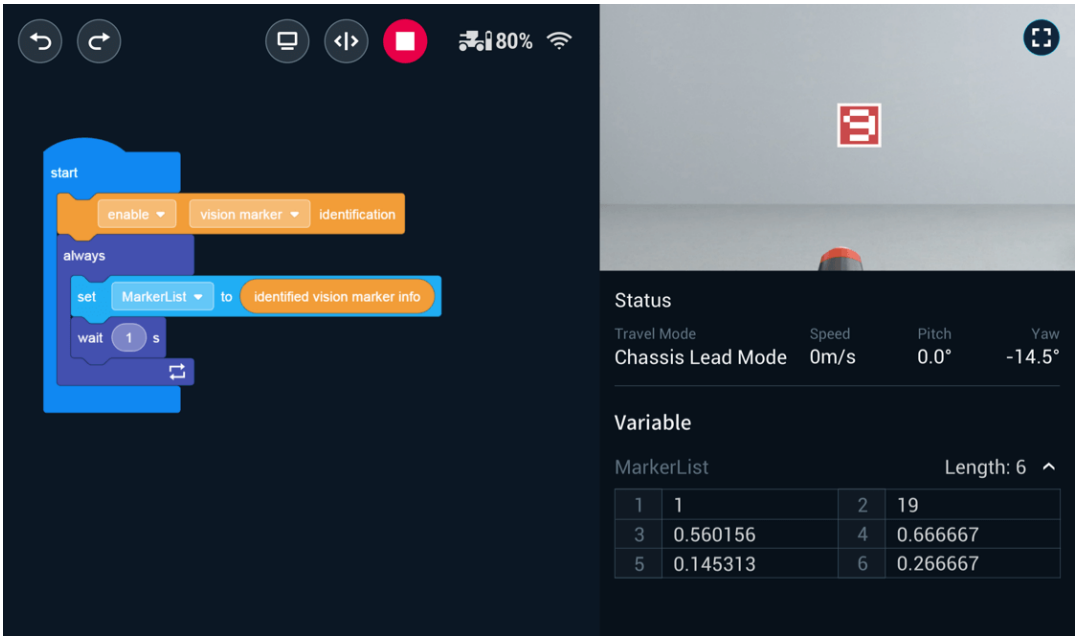
(1) Objective: Obtains information for an identified Vision Marker, returned as N,ID,X,Y,W,and H parameters.

(2) Type: Information block (list-type data)

(3) Example: Recognize Vision Markers

When a Vision Marker appears in the robot's field of view, you will be able to observe Vision Marker parameters in the FPV page. You can move the Vision Marker to observe changes to any of the 6 parameters.

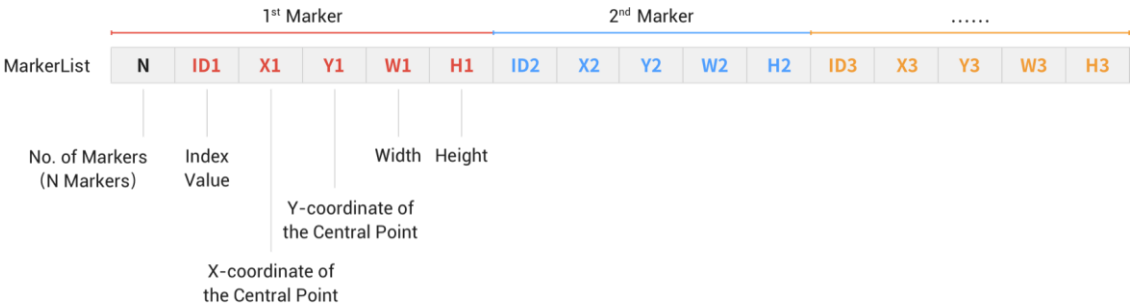




Note:

1) The vision marker format is as follows:

The first item is the number of markers the robot has detected. The second item is a group of 5 values: Vision Marker ID, X-axis of the center point, Y-axis of the center point, W-width and H-height. The same format is used for all following items, as shown below:



2) Descriptions for returned ID values identified by the robot:

ID=0: emit sounds

ID=1: stop

ID=2: dice

ID=3: target

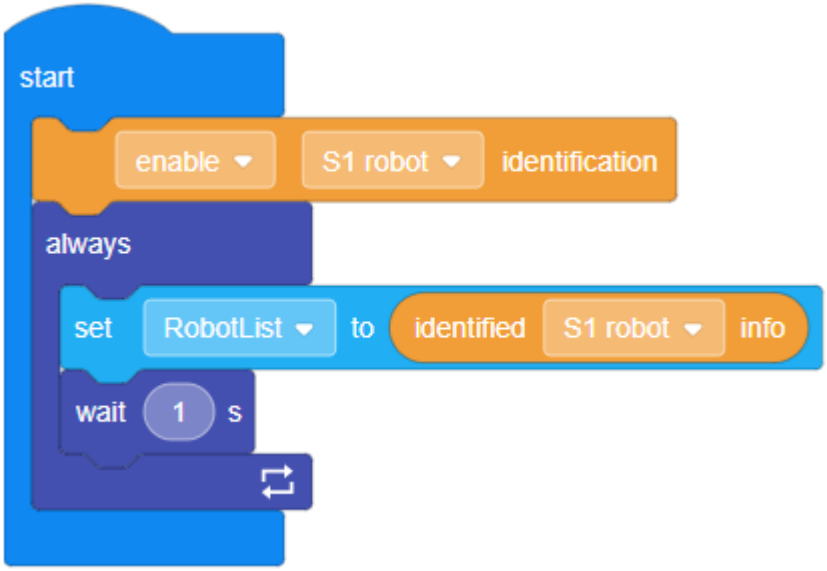
ID=4: left arrow
ID=5: right arrow
ID=6: forward arrow
ID=8: red heart
ID=10-19: 0-9
ID=20-45: A-Z

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

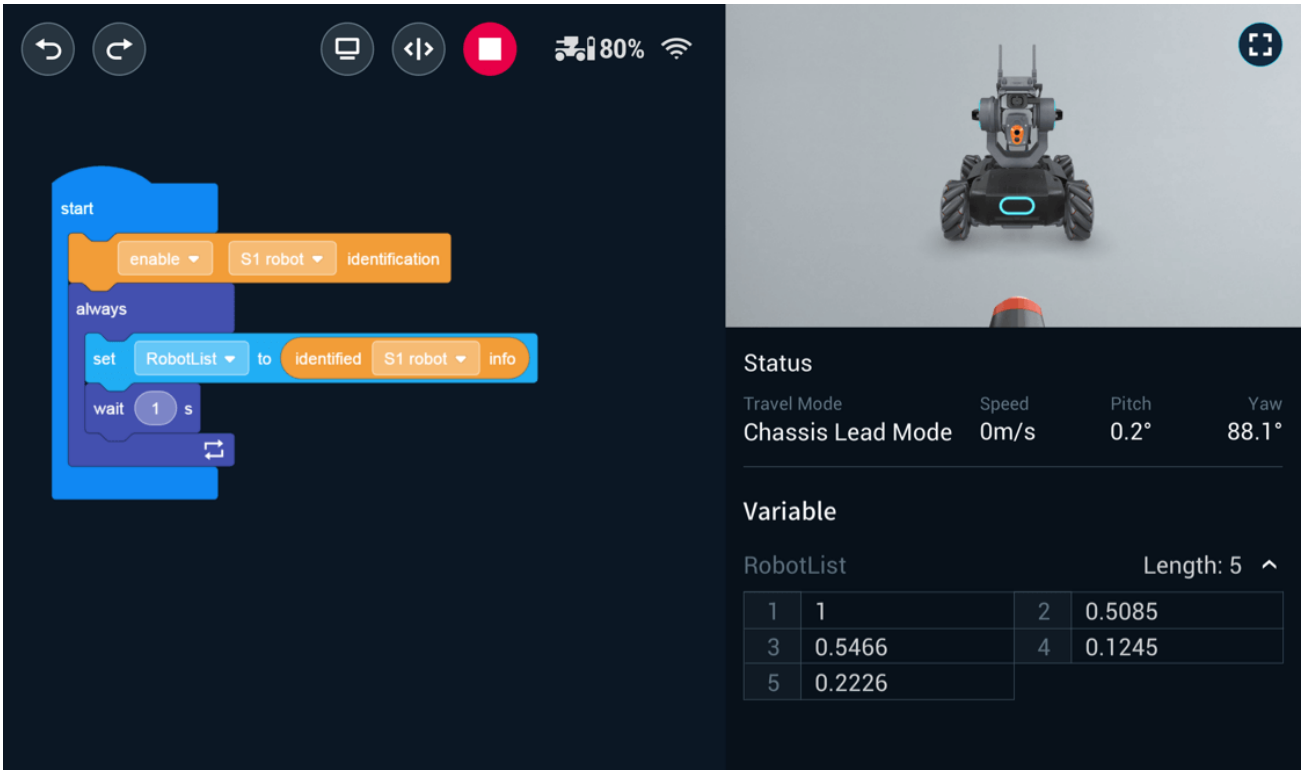
- detection_info(list)

16. identified (person) info

(1) Objective: Obtains an identified person or robot's information in terms of the parameters N, X, Y, W, and H
(2) Type: Information acquisition block (list-type data)
(3) Example: Identify another S1 robot
When another S1 robot appears in the robot's field of view, information about the robot will display in the FPV window. You can move the other robot within your robot's field of view and observe how these 5 values change.



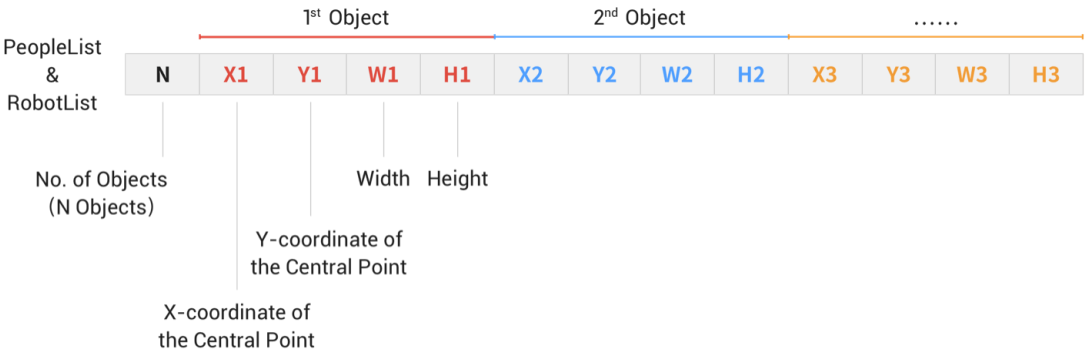
You can observe data changes using the FPV window:



Note:

The format of object information is as follows:

The first item is the number of objects the robot has identified. The second item is a group of 4 numbers: X-axis of the center point, Y-axis of the center point, W-width and H-height. The same format is used for all following items, as shown below:



Python API:

Function: vision_ctrl.get_people_detection_info()

vision_ctrl.get_car_detection_info()

Return value:

- detection_info(list)

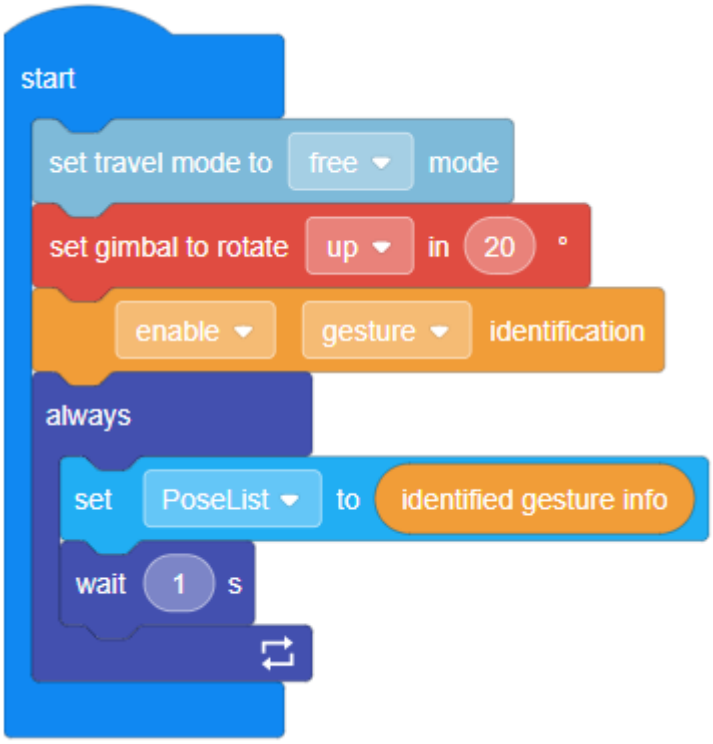
17. identified gesture info

(1) Objective: Acquires identified gesture information in terms of the parameters N, ID, X, Y, W, and H

(2) Type: Information block (list)

(3) Example: Identify posture

You can observe changes in gesture-related data using the FPV window.



Note:

1) The format for gesture information is as follows:

The first item is the number of gestures the robot has identified. The second item is a group of 5 numbers: gesture ID, X-axis of the center point, Y-axis of the center point, W-width and H-height. The same format is used for all following items, as shown below:

2) Descriptions for ID values:

ID=4: V

ID=5: inverted V

ID=6: take photo

Python API:

Function: vision_ctrl.get_pose_detection_info()

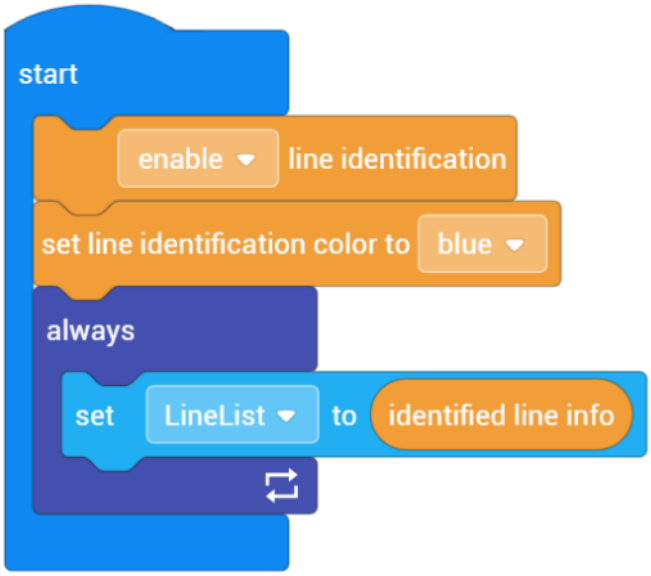
Return value:

- detection_info(list)

18. identifiable single line information

(1) Description: Obtain identifiable single line information. The parameters are N, Info, X, Y, θ, and C.

(2) Type: Information (table type data)
(3) Example: Single line information
Obtain the returned line information after a blue line appears in the robot's FOV.



You can observe the real time line data changes through the FPV window.

Status

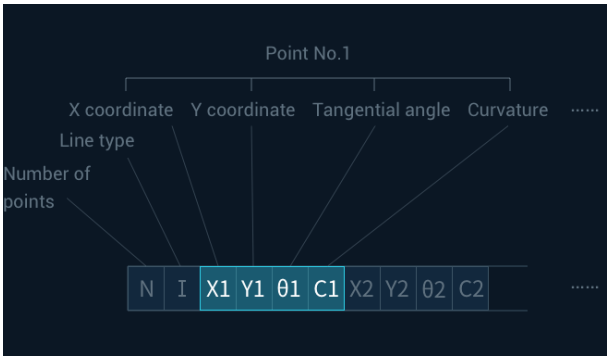
Travel Modes	Speed	Pitch	Yaw
Chassis Lead Mode	2.4m/s	-20.0°	0.0°

Variable

LineList Length: 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
25	-3.591615	26	0.056703
27	0.496875	28	0.627778

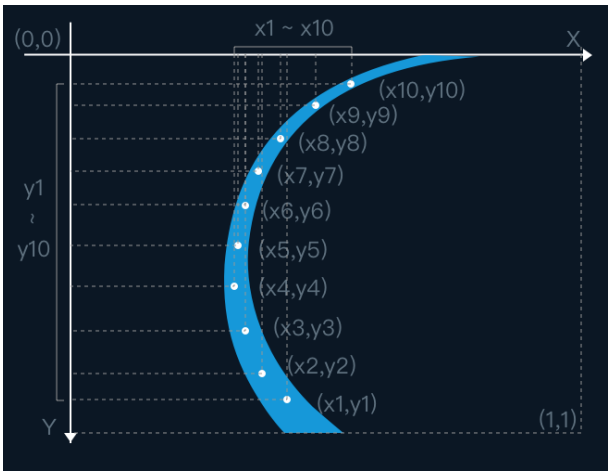
Note:
1) The single line information format that is identifiable by the robot is as follows:
The first item "N" is the number of points identified on the line, the second item 'Info' is the line type, and the subsequent items are grouped into 4 data groups: they are the (Coordinate X, ordinate Y, actual tangent angle θ, curvature C) of ten points that are equidistant from each other on the line (from the nearest to the farthest). There are altogether 42 sets of data.

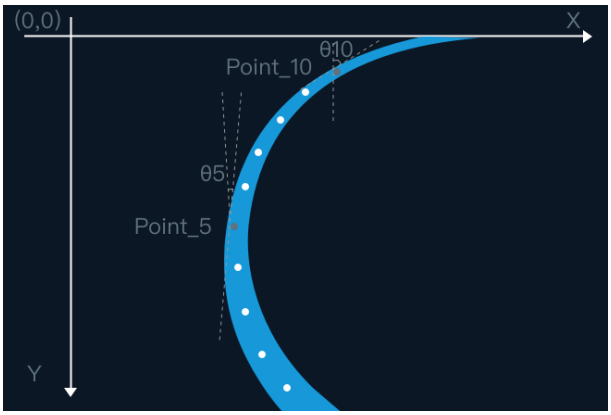


The first data set, N: Number of points The fixed value is 10 or 0 with 10 indicating line detection and 0 indicating the opposite.

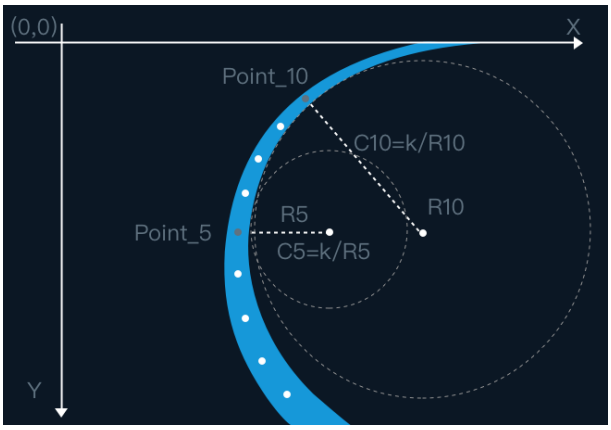
The second data set, I: line type 4 cases : "0", indicates no line is identified; "1", indicates one line within the FOV; "2", indicates a Y intersection; "3", indicates a crossroad.

The third and fourth data sets (X1 and Y1) indicate the coordinate information of the first point on the line. The first point is at the bottom of the FOV, which is the point closest to the robot.

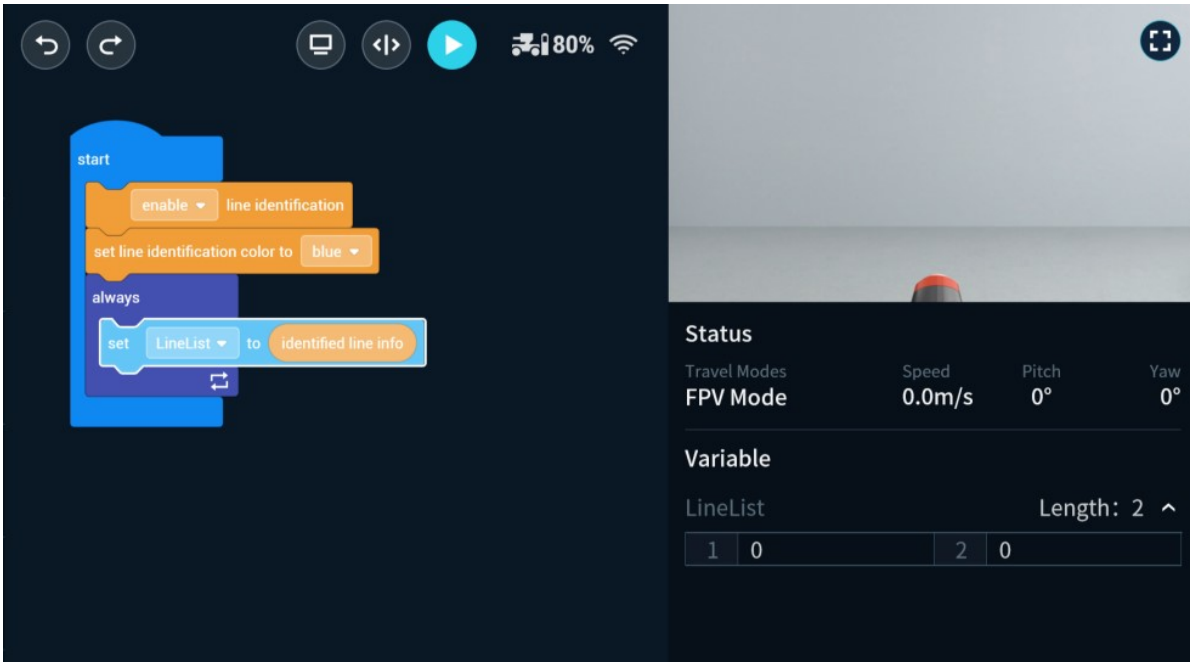




The fifth data set θ1: Tangent angle of the first point. When the tangent angle is 0, it means that the angle between the tangent and the vertical direction of this point is 90 degrees; when the tangent angle is 90, it proves that the line that this point is very curved.



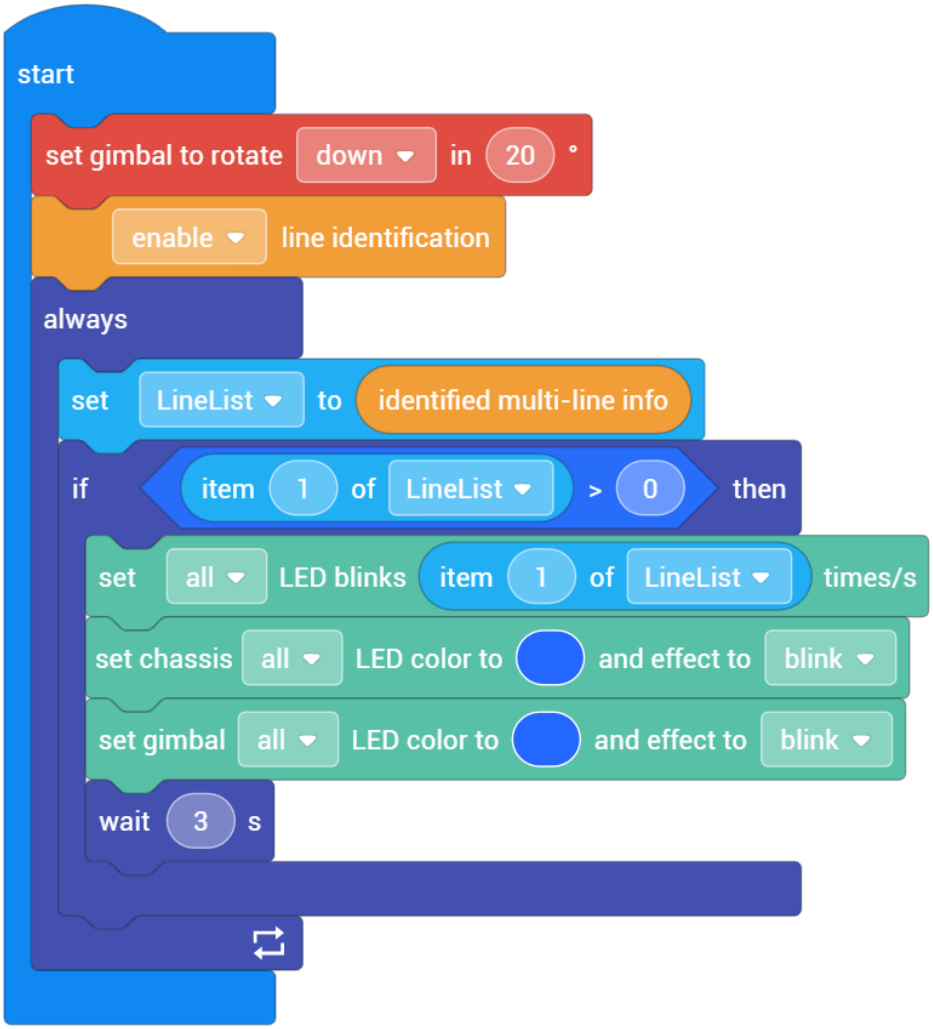
The sixth data set C1: The curvature of the first point. C=k/R. The value ranges from 0 to 10. 0 indicates that the line is a straight line and the radian increases correspondingly as the value increases from 1 to 10. When the circumscribed circle is larger, it proves that the radian of the line near this point is larger.
2) If no line is identified by the robot, the length of the returned line data is 2 while the value of the first and second items are all 0.



19. identifiable multi-line information

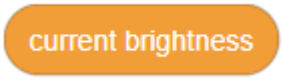
identified multi-line info

(1) Description: Obtain identifiable multi-line information . The parameters are n, X, Y, θ, and C.
(2) Type: Information (table type data)
(3) Example: Multi-line blink indication
When the robot identifies N lines, all the gimbal and chassis LED indicators blink blue N times

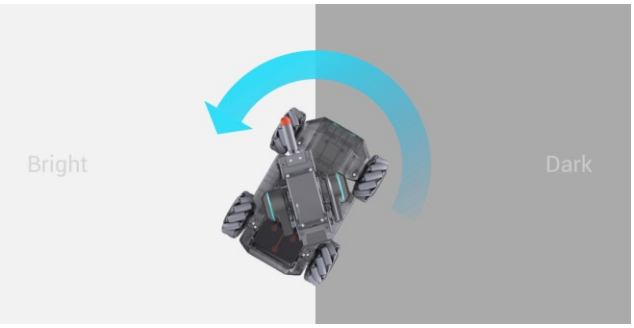


Note:
The multi-line information format identified by the robot is as follows:
The first item n (number of lines), from clockwise, the (Coordinate X, ordinate Y, actual tangent angle θ, curvature C) of ten points that are equidistant from each other on the first line (from the nearest to the farthest), the (X, Y, θ, C) of ten points on the second line ..., the (X, Y, θ, C) of ten points on the numbered n line. There are altogether 40n +1 values.
Compared with the "Identifiable Single Line Information" module, the "Identifiable Multi-Line Information" module has one feedback more on the number of lines.

20. current brightness



(1) Objective: Obtains information about the brightness of the current environment and returns a value of 0-10; the greater the value is, the brighter the environment.
(2) Type: Information block (variable)
(3) Example: Translate toward light

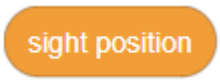


In dark environments, you can turn on the Blaster Trajectory Light and turn the robot to face the brighter area.

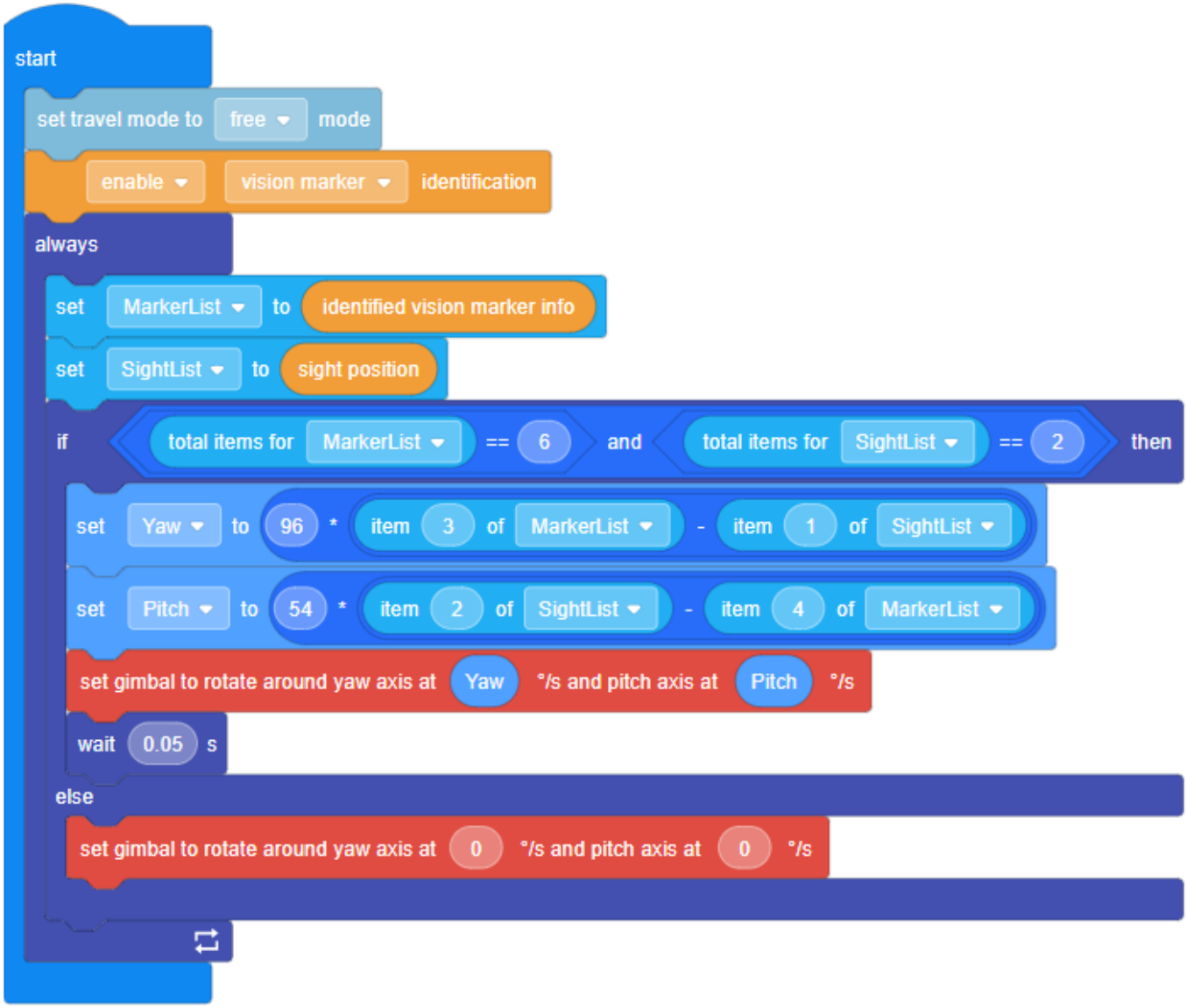
Python API:
Function: vision_ctrl.get_env_brightness()
Return value:

- brightness_value(int)

21. sight position

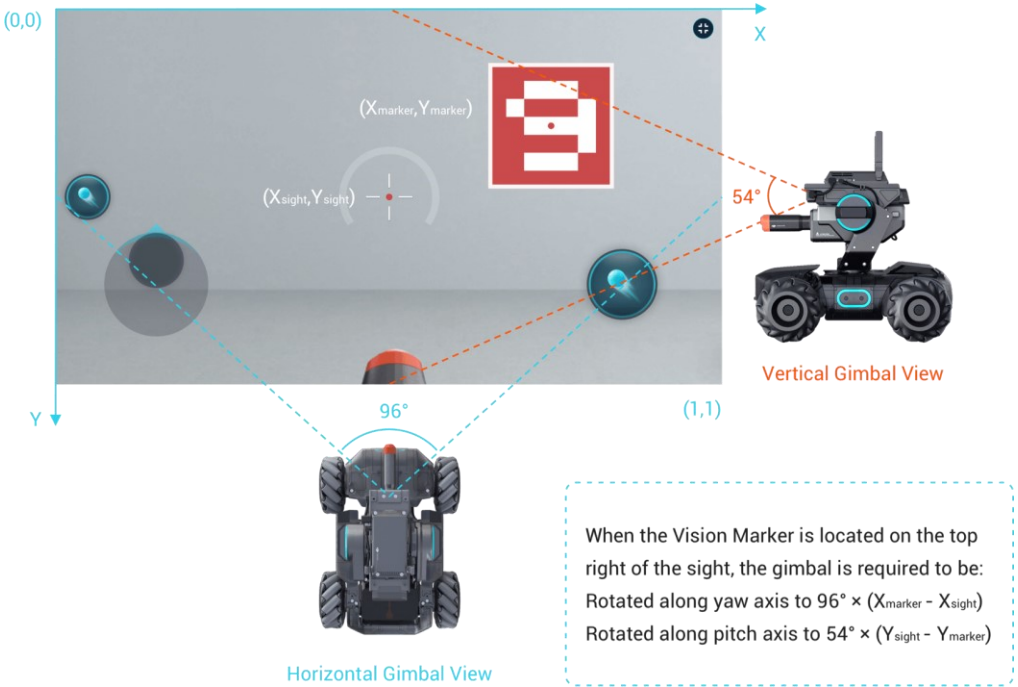


(1) Objective:Obtains information for sight position in terms of the parameters X and Y .
(2) Type: information block (list)
(3) Example: Follow visual markers
This will convert the difference between the sight position and a Vision Marker in the robot's field of view into a gimbal rotation angle value in order to direct the gimbal to move toward the Vision Marker.



Note:

Sign position format: X represents X-coordinate,and Y represents Y- coordinate.



Python API:

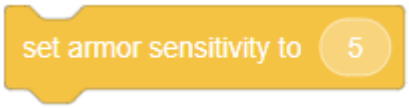
Function: media_ctrl.get_sight_bead_position()

Return value:

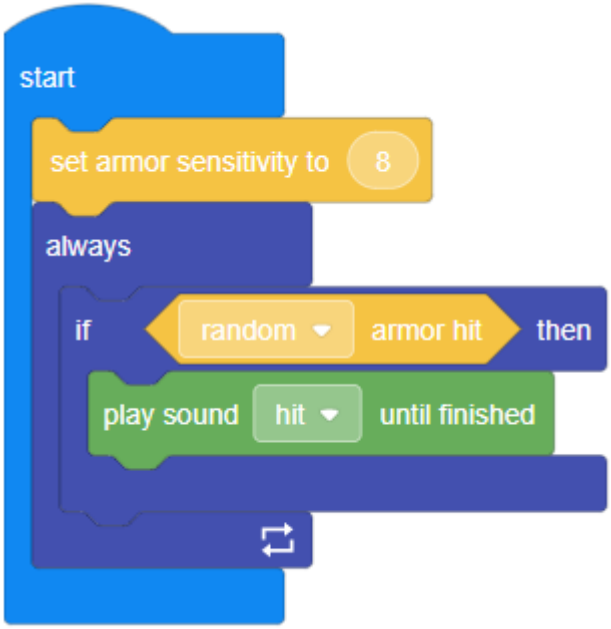
- sight_bead_position(list)

Armor

1. set armor sensitivity to (5)



(1) Objective: Sets armor sensitivity; the larger the value, the higher the armor sensitivity. For testing armor sensitivity with hard objects or by tapping, the recommended values are 6 and 8, respectively.
(2) Type: Settings block
(3) Response to tapping
Tap anywhere on the EP's armor to play the corresponding sound effect.



Note:
Configuring armor sensitivity is only available in laboratory testing. Any armor sensitivity settings will be restored to default during competition.

Python API:
Function: armor_ctrl.set_hit_sensitivity(value)
Parameters:

- value(int): [0, 10]

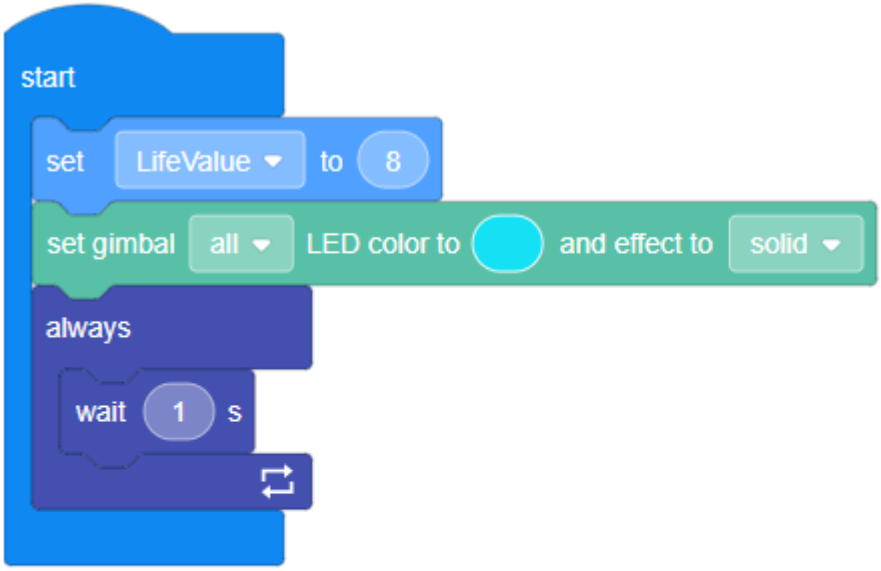
2. when (any) armor hit

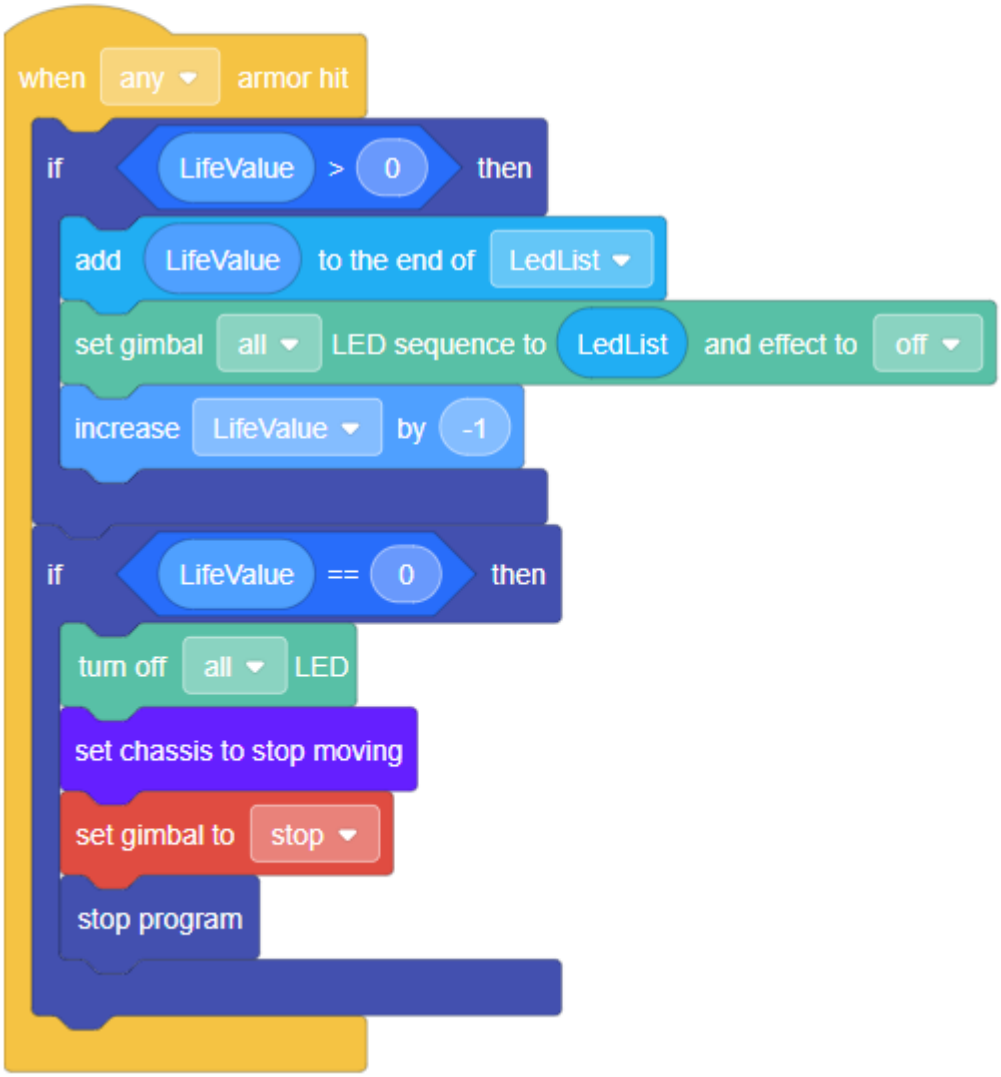
(1) Objective: Runs the block's program when the armor is hit at a specified section
(2) Type: Event block

(3) Example: Configure HP level

The gimbal LEDs indicate the current HP level of the robot; 8 LEDs indicates full HP.

When any point on the armor is hit, the robot's HP reduces by 1; when the HP level drops to 0, all LEDs will turn off and the robot will stop moving.





Note:

Event blocks have the highest priority and contain programs where main function will pop out and begin to run when certain conditional statements are met, regardless of the main function's current status.

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

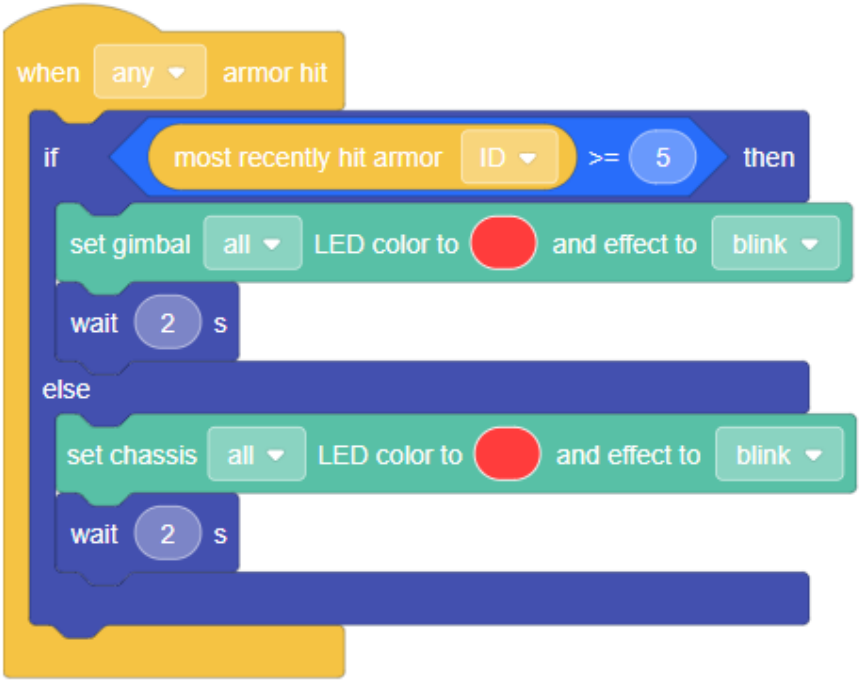
3. most recently hit armor (ID)

(1) Objective: Displays information for the last armor section hit; the ID value indicates the specific section that was hit, and the running time shows the time it was hit

(2) Type: Information block (variable-type)

(3) Example: Configure hit section indicator

If the gimbal is the last armor section hit, all LEDs on the gimbal will flash red. The same principle applies to chassis armor.



Note:
The returned value indicates the section of the armor that has been hit:
ID=1: Chassis rear
ID=2: Chassis front
ID=3: Chassis left side
ID=4: Chassis right side
ID=5: Gimbal left side
ID=6: Gimbal right side

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. (random) armor hit

(1) Objective: Continuously detects hit sections on a specific armor section. When the armor is hit, “True” is returned; otherwise, “False” is returned.
(2) Returned value: Boolean
(3) Example: Configure quick retreat
If the left side of the chassis is hit, the robot will retreat by moving to the right; if the front of the chassis is hit, the robot will retreat by moving backward.

Python API：

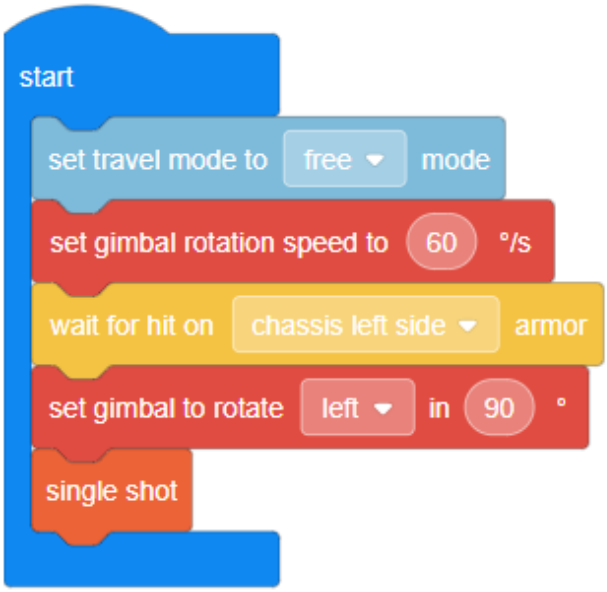
Function: armor_ctrl.check_condition(condition_enum)

Parameters:

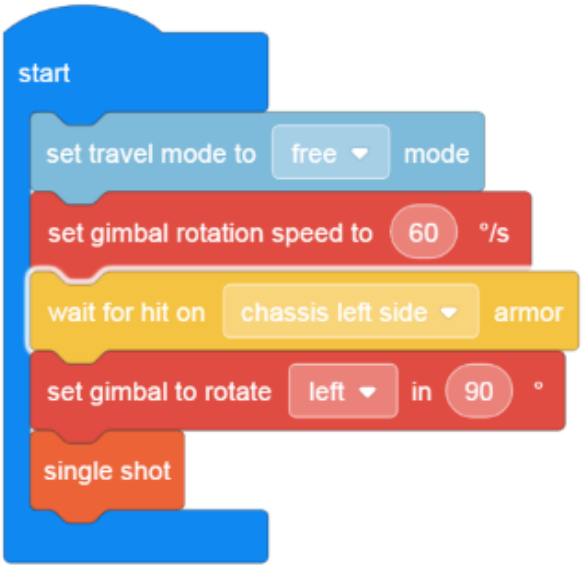
- condition_enum(enum):
  - rm_define.cond_armor_hit
  - rm_define.cond_armor_bottom_front_hit
  - rm_define.cond_armor_bottom_back_hit
  - rm_define.cond_armor_bottom_left_hit
  - rm_define.cond_armor_bottom_right_hit
  - rm_define.cond_armor_top_left_hit
  - rm_define.cond_armor_top_right_hit

5. wait for hit on (random) armor

(1) Objective: Executes the next command when the specified armor section is hit; otherwise, continues to wait
(2) Type: Execution block, Blocking block
(3) Example: Configure hit defense



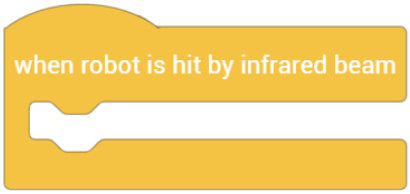
Note:
In this program, if the left side of the chassis is not hit, the program will wait and remain on this block, and the block will remain highlighted.



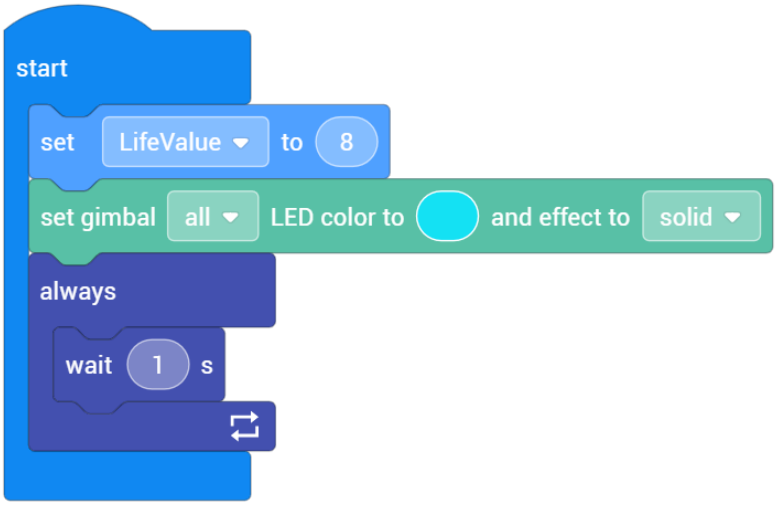
Python API:
Function: armor_ctrl.cond_wait(condition_enum)
Parameters:

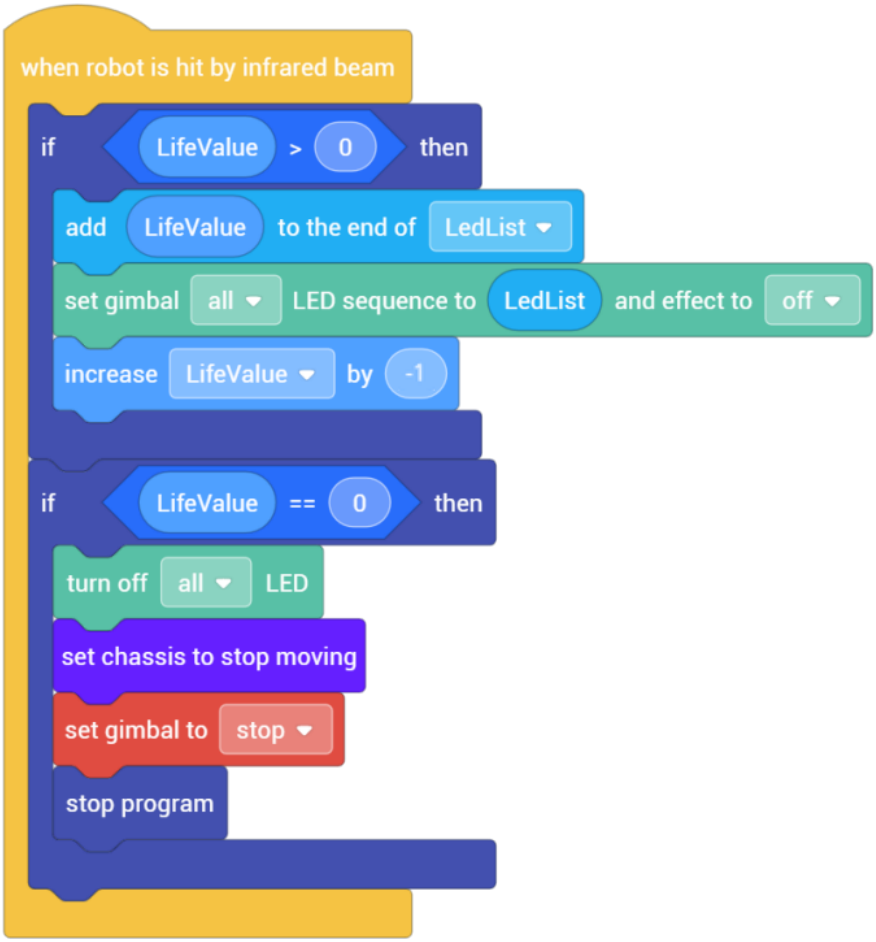
- condition_enum(enum):
  - rm_define.cond_armor_hit
  - rm_define.cond_armor_bottom_front_hit
  - rm_define.cond_armor_bottom_back_hit
  - rm_define.cond_armor_bottom_left_hit
  - rm_define.cond_armor_bottom_right_hit
  - rm_define.cond_armor_top_left_hit
  - rm_define.cond_armor_top_right_hit

5. when the robot is attacked by infrared beams



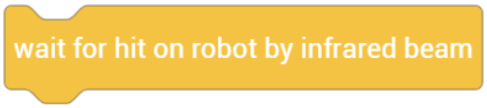
(1) Description: Run the program in this module when the infrared sensors on both sides of the robot's gimbal are attacked by infrared beams.
(2) Type: Execution
(3) Example: Health light bar
The gimbal LED indicator can indicate the current hit points of the robot, and the maximum hit points is 8. When the robot is attacked by infrared beams, the robot loses 1 hit point; when the hit point is 0, the LED indicator extinguishes and the robot stops moving.





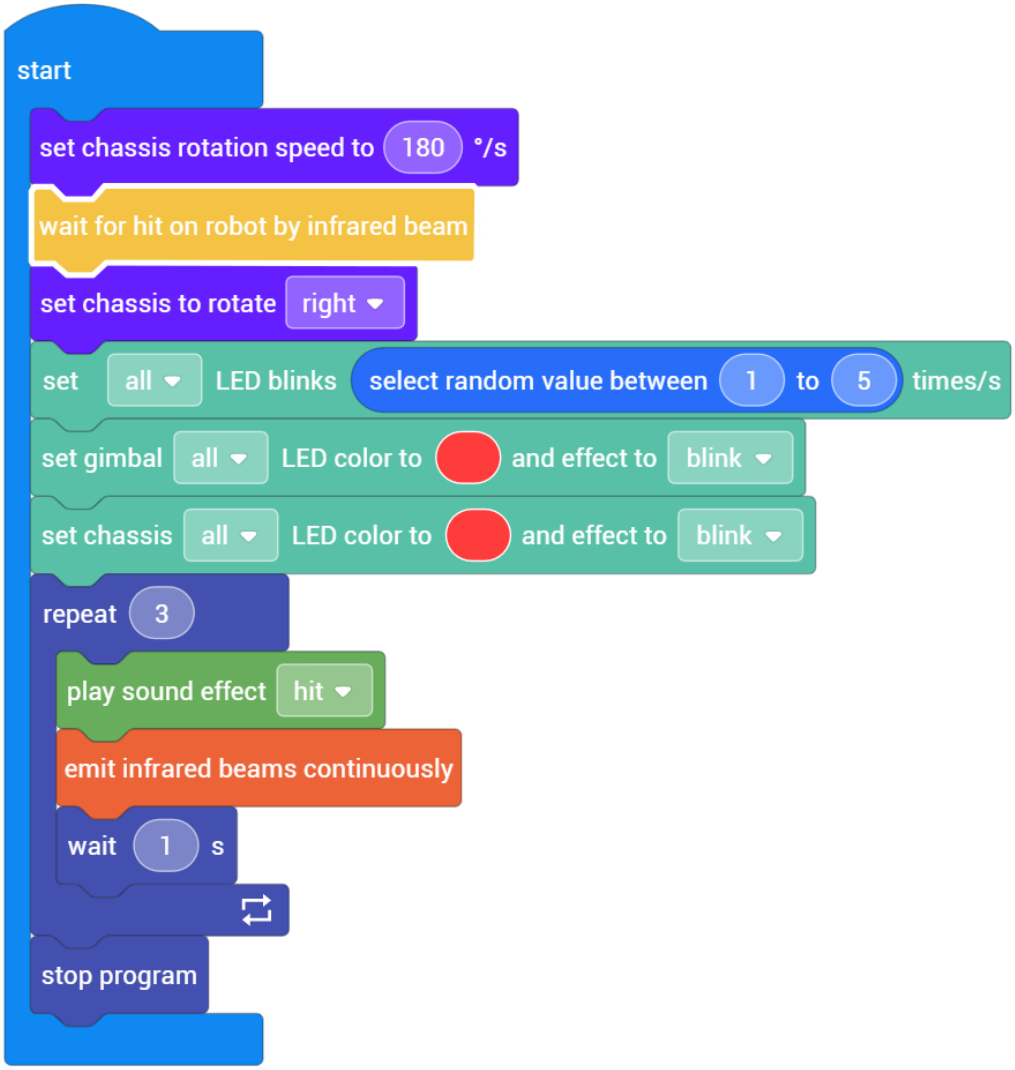
Note:
The event trigger module has a higher priority, which means that no matter which step the main function reaches, as long as the trigger condition is met, the robot will immediately jump out of the main function and start running the programs in the event module.

6. wait for hit on robot by infrared beam



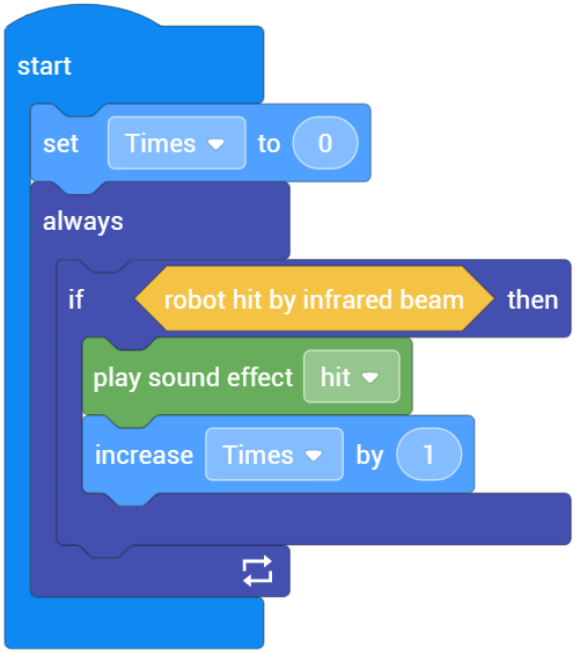
(1) Description: Execute the next instruction only after the infrared sensors on both sides of the gimbal are attacked by infrared beams; otherwise, continue to wait.
(2) Type: Execution, Blocking
(3) Example: Self-rotate for protection
After being attacked by infrared beams, the robot rotates in place to reduce the probability of being hit and fights back.

Note:
For this function, if no infrared beam attack is detected by the infrared sensor on the robot's gimbal, the program will wait on this module and the module will remain highlighted at this time.



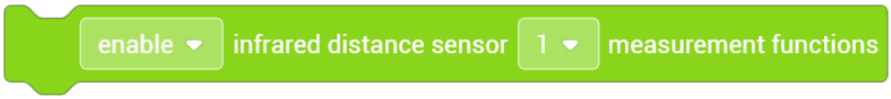
7. robot hit by infrared beam

(1) Description: Check whether the infrared sensors on both sides of the robot's gimbal have been attacked by infrared beams. Returns "True" if yes; otherwise, returns "False".
(2) Return value: Boolean
(3) Example: Number of hits
If the robot is attacked by an infrared beam, a "hit" sound is played and the number of hits is accumulated.

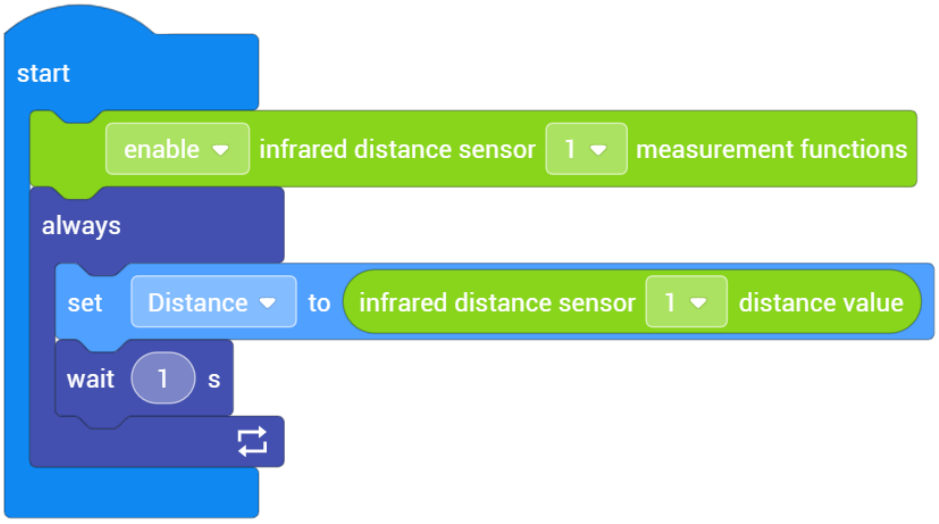

start
set Times to 0
always
if robot hit by infrared beam then
play sound effect hit
increase Times by 1

Sensor

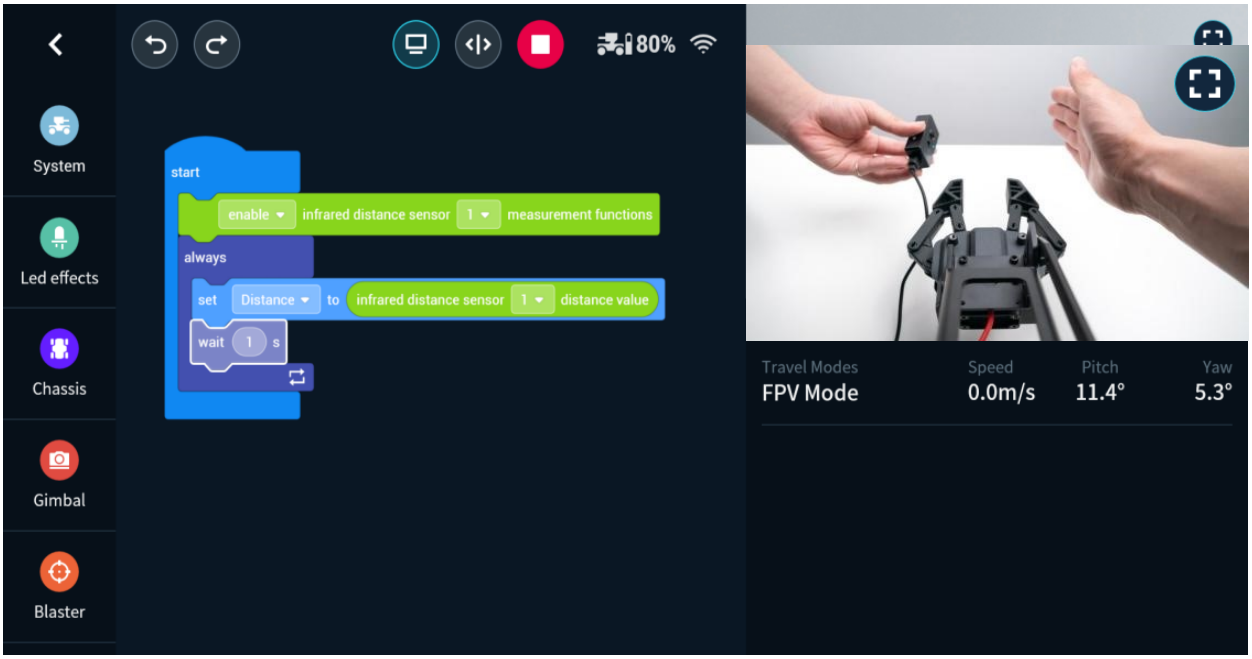
1. (turn on) infrared distance sensor (tof) no.(1) for distance measuring



(1) Description: Turn on or turn off the distance measuring function of a specified infrared distance sensor (No.).
(2) Type: Setting
(3) Example: Distance measuring using infrared beams
Measure the distance between the infrared distance sensor and your palm.



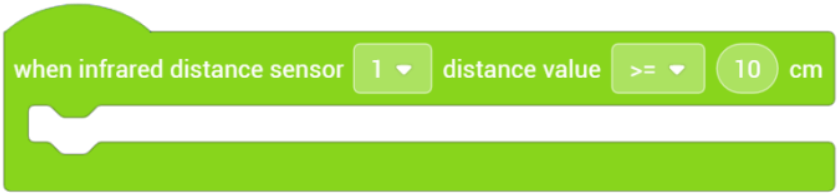
Move your palm forward and backward and observe the real time distance change through the FPV window.



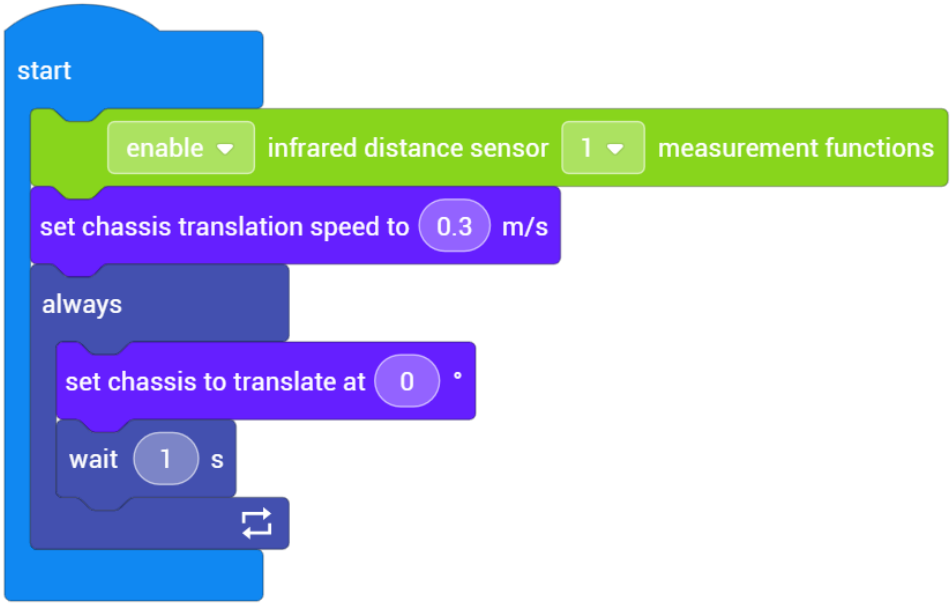
Note:

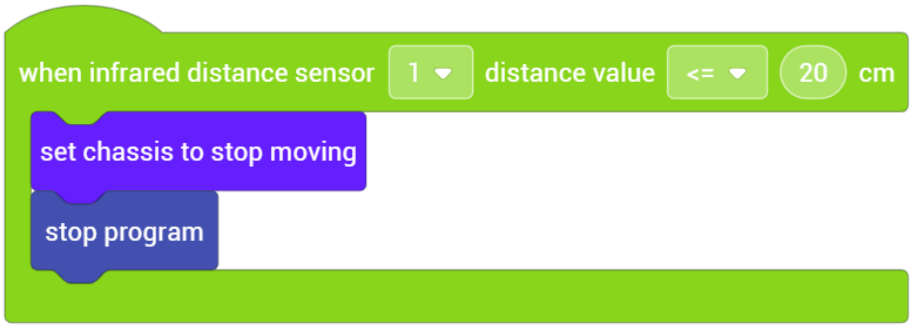
The distance measuring function of the infrared distance sensor is achieved mainly by the Time-of-Flight (ToF) camera. The sensor fires out a modulated light pulse (such as infrared beams shot by the infrared distance sensor) which then reflects off the object. The ToF camera measures the distance between the sensor and the measured object by calculating the time (flight time) the light pulse takes to reflect off the object.

2. when the distance measured by infrared distance sensor no.(1) is (⩾) (10) centimeters



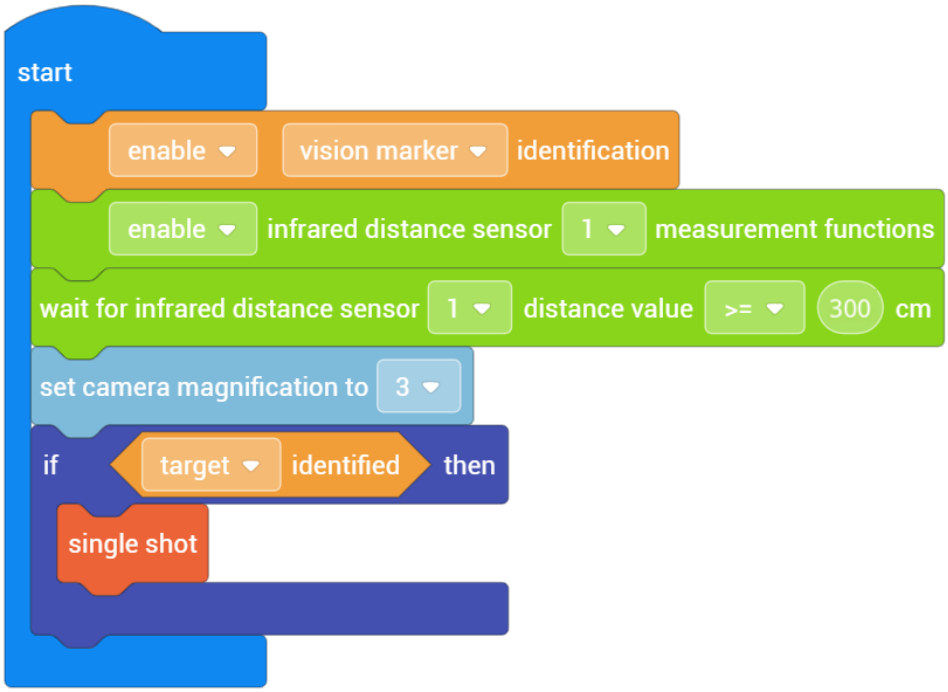
(1) Description: Run the program in this module when the distance measured by the specified infrared distance sensor (No.) meets the condition.
(2) Type: Event
(3) Example: Continuously approaching
Control the robot mobile chassis so that it continuously translates towards the wall until the distance measured by the infrared distance sensor is ≤ 20 cm, where it will then stop.





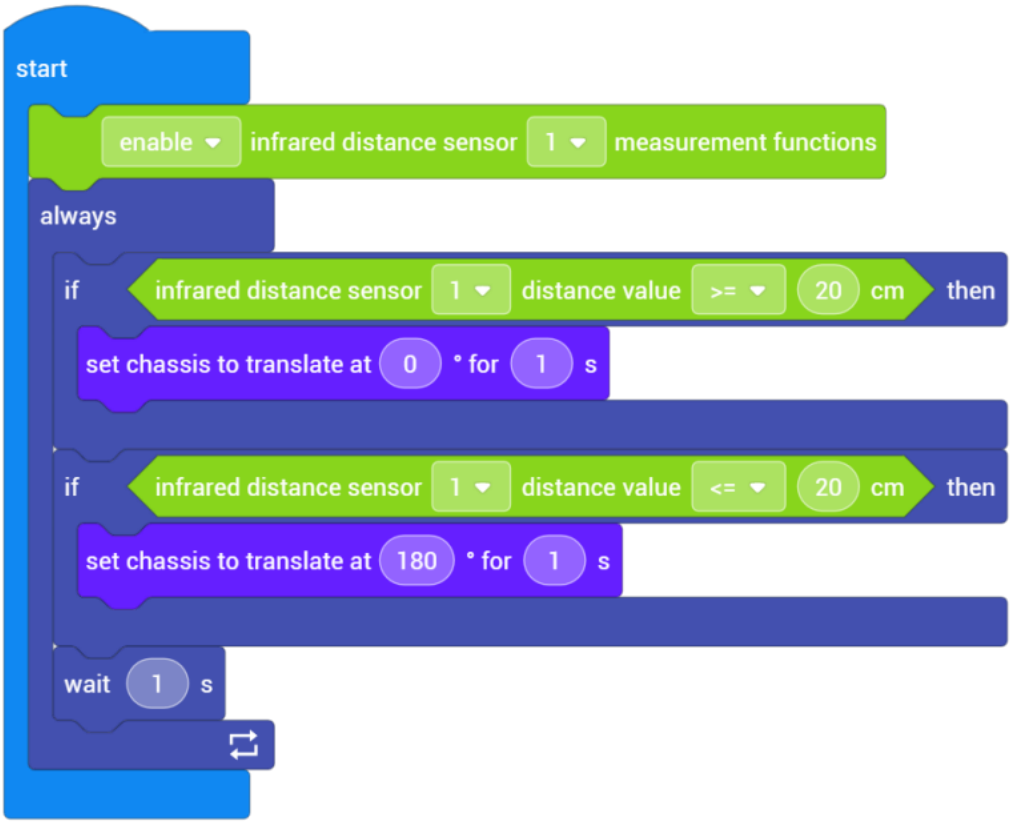
3. wait until the distance measured by infrared distance sensor no.(1) is (⩾) is (10) centimeters

(1) Description: Execute the next instruction when the distance measured by the specified infrared distance sensor (No.) meets the condition; otherwise, it will stay still and wait.
(2) Type: Execution, Blocking
(3) Example: Long-distance shooting
Hold a visual tag and back away from the robot. When the infrared distance sensor detects that the "target" is already 3 meters away from the robot, the robot will fire a long range shot by zooming in with the camera.



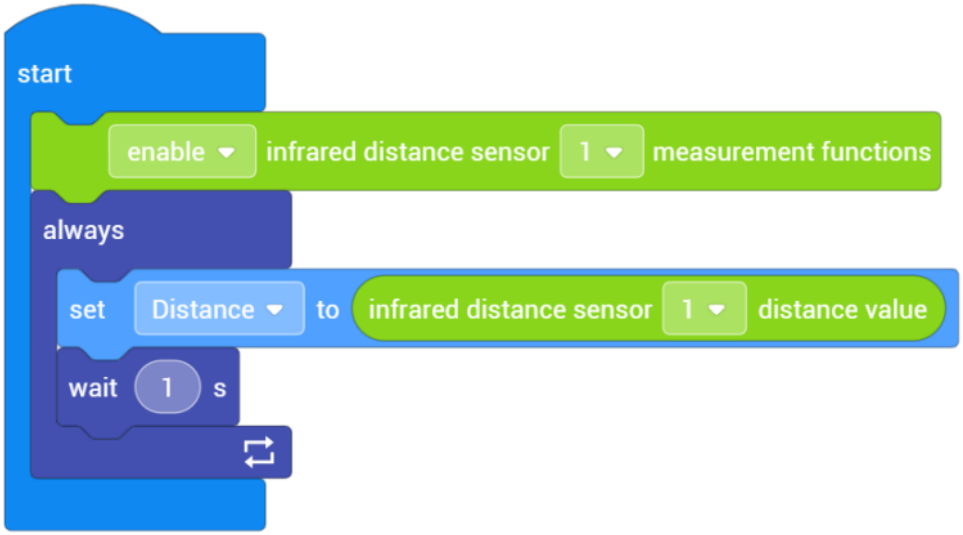
4. when distance measured by infrared distance sensor no.(1) is (⩾) (10) centimeters

(1) Description: Returns "True" when the distance measured by the specified infrared distance sensor (No.) meets the condition; otherwise, returns "False".
(2) Return value: Boolean
(3) Example: I retreat as you advance
In order to maintain an appropriate distance with the palm, the robot has mastered the "I retreat as you advance, I advance as you retreat" strategy: if the distance measured by the infrared distance sensor is ≤ 20 cm, the robot translates backward; if the distance is ≥ 20 cm, the robot translates forward.

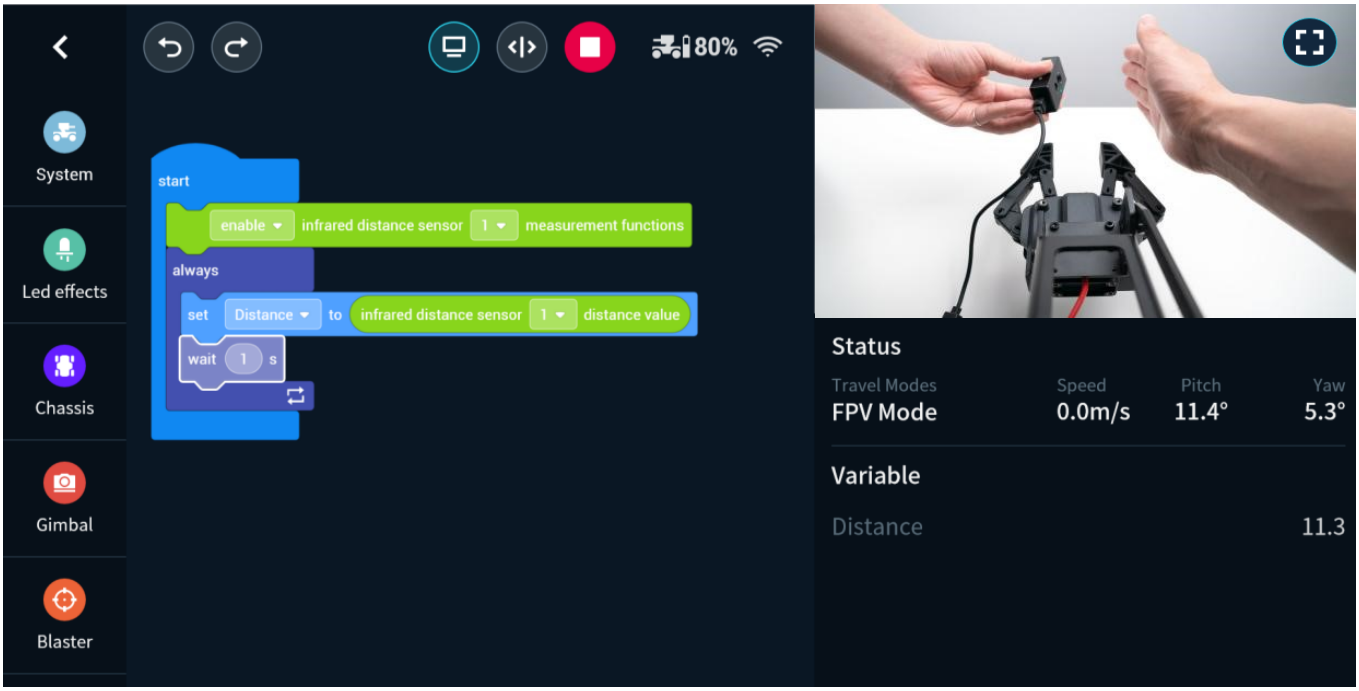


5. Distance Measured By Infrared Distance Sensor No.(1)

(1) Description: Obtain the distance measured by the specified infrared distance sensor (serial No.). The unit is centimeter.
(2) Type: Information
(3) Example: Distance measuring using infrared beams
Measure the distance between the infrared distance sensor and your palm.



Move your palm forward and backward and observe the real time distance change through the FPV window.

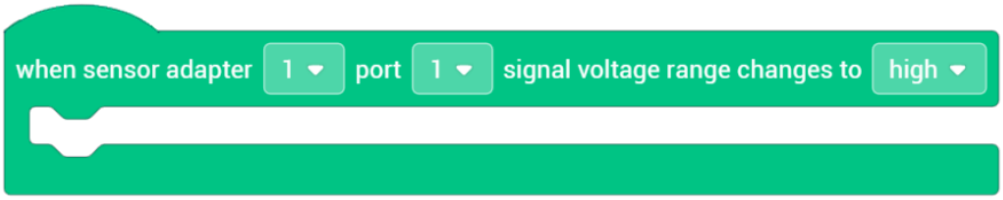


Note:

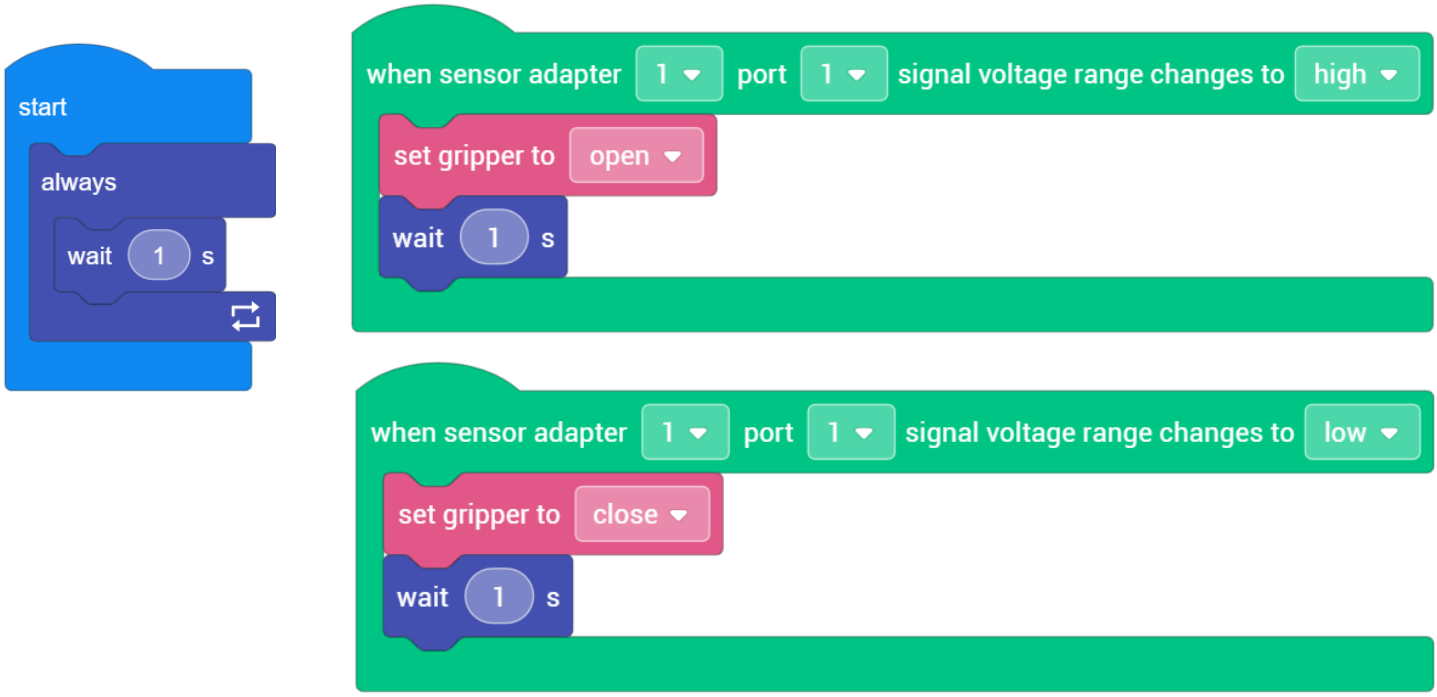
The distance that the infrared distance sensor can read ranges from 10 centimeters to 10 meters.

Sensor Adaptor

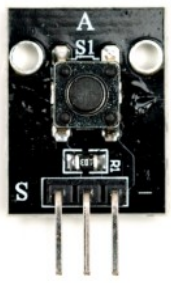
1. when the level transition of no.(1) port of sensor adaptor no.(1) becomes (high)



(1) Description: Execute the program in this module when the I/O pin level of the specified port of the specified sensor adaptor meets the conditions.
(2) Type: Event
(3) Example: Use a button to control the opening or closing of the gripper.
Connect a button to sensor adaptor No.(1). When the button is pressed, the robot gripper opens. When the button is released, the robot gripper closes.

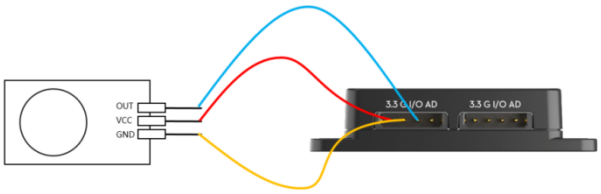


Note:



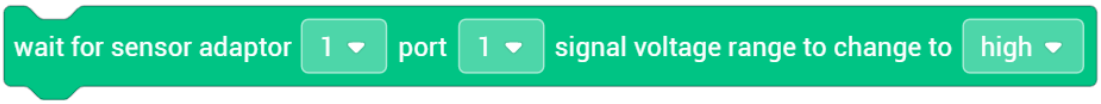
1) The output level of the button can be high or low. When the button is "pressed" and "released", the output level changes, and the I/O pin of the sensor adaptor receives the level signal change so as to control the opening or closing of the gripper.
2) The three pins on the button need to be connected to the designated spots of the sensor adaptor through DuPont wires:

Button Pin	Sensor Adaptor Port
OUT (Output)	I/O (Input/Output)
VCC (Power supply)	3.3 (3.3 Volt power supply)
GND (Ground)	G (Ground)

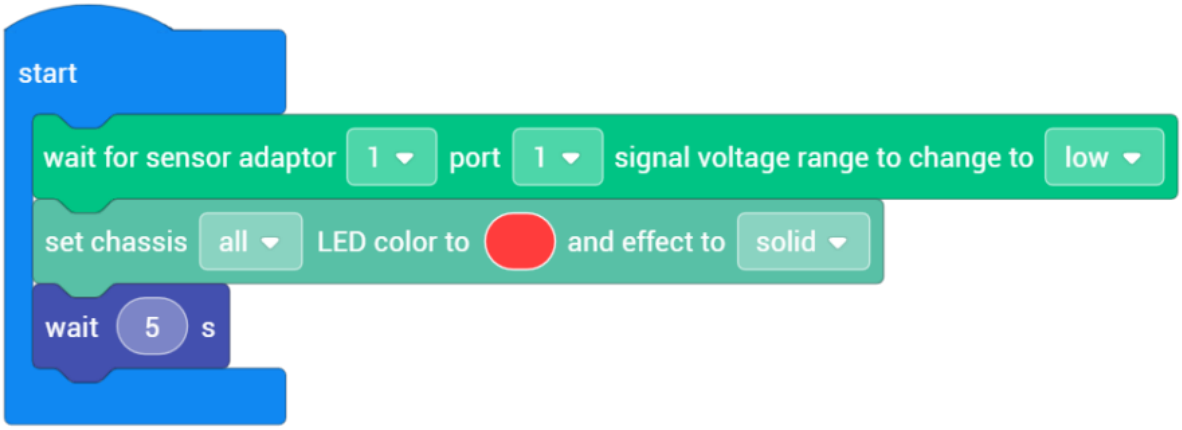


3) A maximum of six sensor adaptors can be connected to the robot at the same time.

2. wait for the level of no.(1) port of sensor adaptor no.(1) to become (high)



(1) Description: Execute the follow-up instructions when the I/O pin level of the specified port of the specified sensor adaptor meets the conditions; otherwise it will continue to wait.
(2) Type: Execution, blocking
(3) Example: Give a warning when the decibel level is too high.
After a sound sensor is connected to sensor adaptor No.(1), if the detected sound is too loud, the chassis LED indicator will turn solid red.

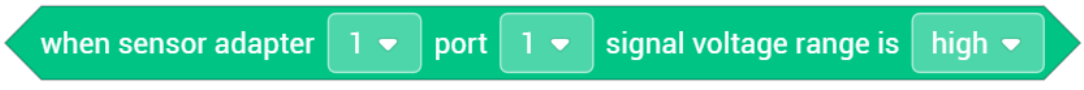


Note:
1) The sound signal received by the sound sensor used in this function design will output a high electrical level if the signal does not reach the set threshold of the sensor, and a low level if the threshold is exceeded, but the situation for different types of sound sensors varies.
2) Output levels by infrared obstacle avoidance sensors and touch sensors are only classified as high or low, therefore, the robot can obtain functional area acquisition through the I/O port of the adaptor module that

detects high and low electrical levels.

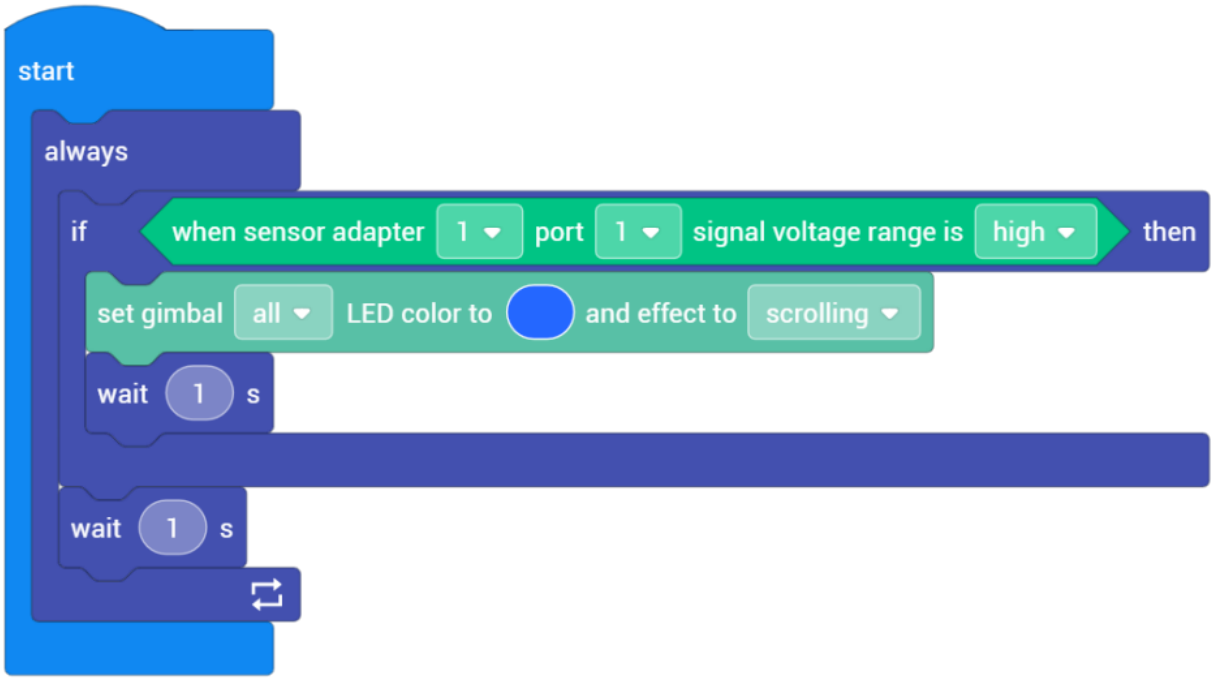
	Trigger		
Infrared obstacle avoidance sensor		VCC(5V) GND OUT(TTL)	High or low electrical level
Vibration sensor		VCC(5V) GND OUT(TTL)	High or low electrical level
Tilt sensor		VCC(5V) GND OUT(TTL)	High or low electrical level
Sound sensor		VCC(5V) GND OUT(TTL)	High or low electrical level
Disk encoding technology		VCC(5V) GND OUT(TTL)	High or low electrical level
Touch sensor		VCC(5V) GND OUT(TTL)	High or low electrical level

3. the electrical level of no.(1) port of sensor adaptor no.(1) is (high)



(1) Description: Returns "True" when the I/O pin level of the specified port of the specified sensor Adaptor

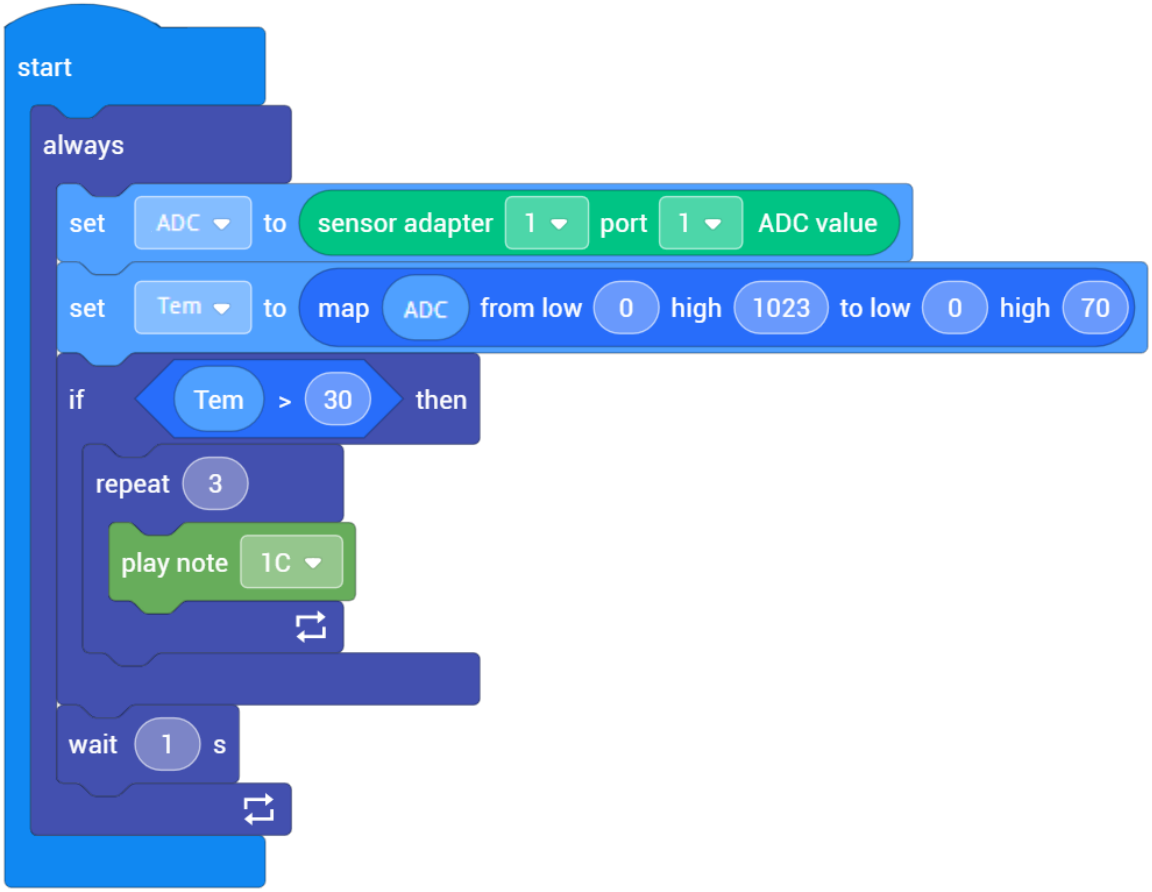
meets the conditions; otherwise, returns "False".
(2) Return value: Boolean
(3) Example: Lighting effect upon touch
Connect a touch sensor to sensor adaptor No.1 and after the finger touches the sensor, the gimbal displays a blue marquee lighting effect.



Note:
The touch sensor used in this function design will output a high electrical level if a finger touches it, and a low level when it is not touched, but the situation for different types of touch sensors varies.

4. the adc value of no.(1) port of sensor adaptor no.(1)

(1) Description: Read the ADC pin value of a specified port of a specified sensor adaptor.
(2) Return value: Information
(3) Example: Temperature alarm
Connect a temperature sensor to sensor adaptor No.1. A sound alert is played when the temperature sensor approaches a cup filled with hot water.



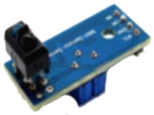
Note:

1) The sensor adaptor ADC is a 10-bit ADC which can take data of up to 1024 unique values, so the ADC range is from 0 to 1023.

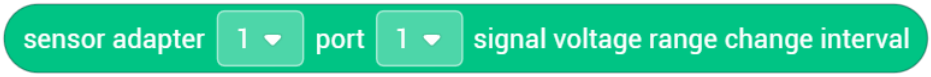
2) ADC collects the analog voltage value. Devices such as pressure sensors, potentiometers, and others have ADC output.

Trigger+ADC

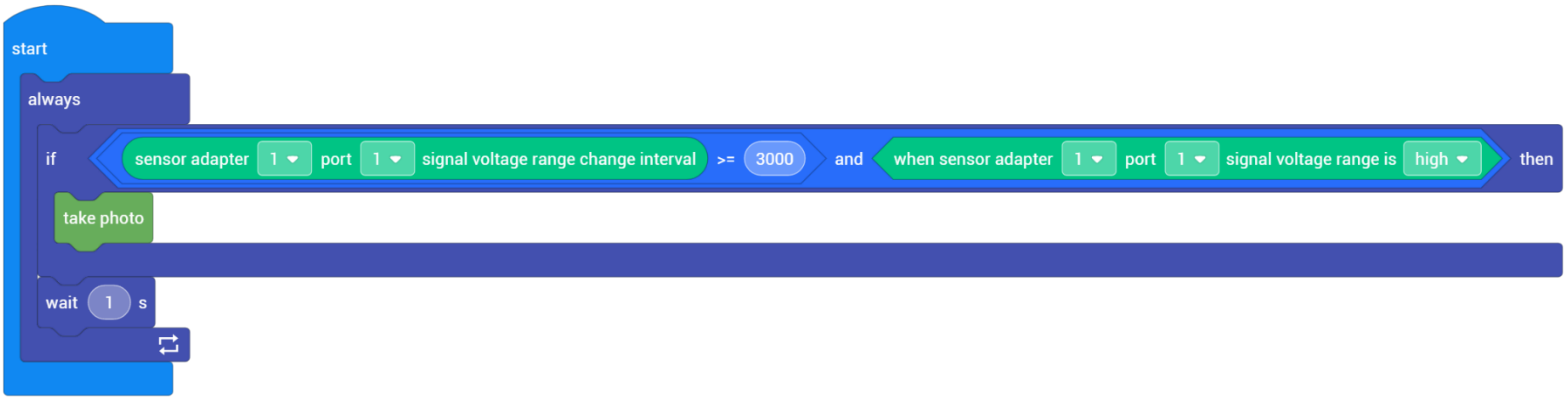
Photo resistor sensor module		VCC(5V) GND OUT(TTL) OUT(AD)	High or low electrical level +ADC
Water level sensor		VCC(5V) GND OUT(AD)	ADC
Hall sensor		VCC(5V) GND OUT(AD) OUT(TTL)	High or low electrical level +ADC
Flame sensor module		VCC(5V) GND OUT(AD) OUT(TTL)	High or low electrical level +ADC

Tracking sensor		VCC(5V) GND OUT(AD) OUT(TTL)	High or low electrical level +ADC
Smoke sensor		VCC(5V) GND OUT(AD) OUT(TTL)	High or low electrical level +ADC

5. pulse duration of no.(1) port of sensor adaptor no.(1)



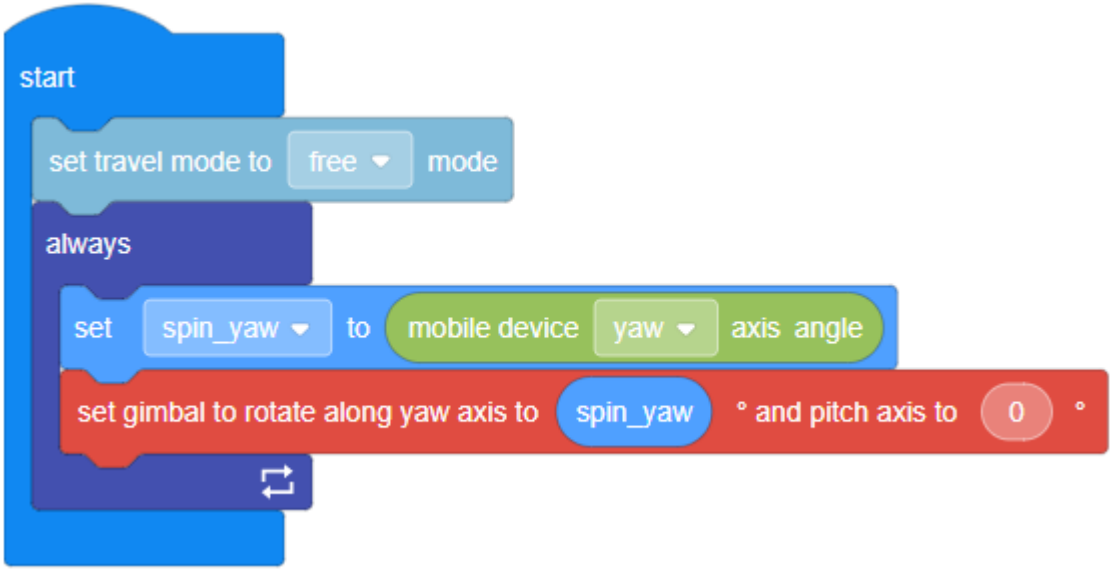
(1) Description: Obtain the duration from when the pulse of the specified I/O pin of the specified sensor adaptor jumps till it is obtained. The unit is millisecond.
(2) Return value: Information
(3) Example: Photo-taking
Connect a button to sensor adaptor No.(1). Starts taking photos if the button is pressed for more than 3 seconds.



Mobile Device

1. mobile device (yaw) axis angle

(1) Objective: Obtains the current angle value of a mobile device
(2) Type: Information block (variable-type)
(3) Example: Dynamic rotation control
Rotate the mobile device to the left or right and observe whether the EP's gimbal rotates accordingly.



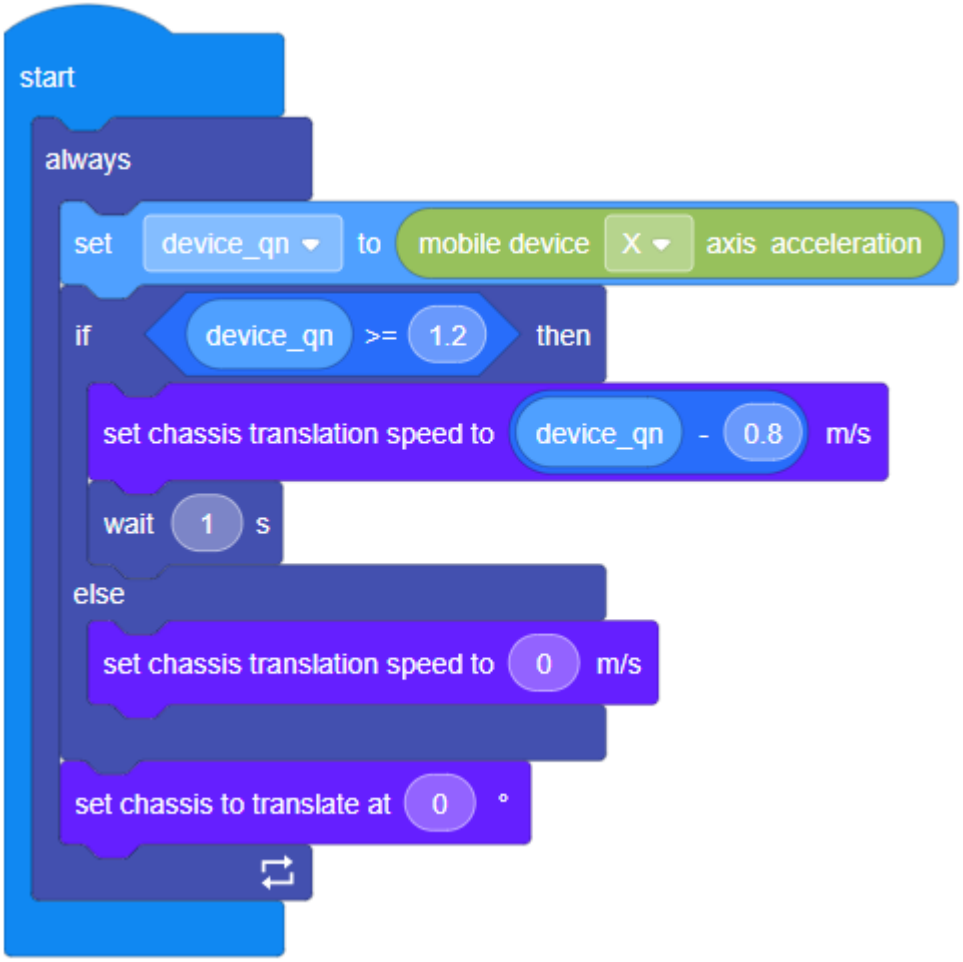
Note:
1) The EP defaults to Chassis Lead Mode. To control the gimbal separately you will need to first set the robot to Free Mode.
2) Mobile device refers to devices such as smartphones and tablets.
3) The gesture angle for mobile devices ranges from approximately -180 to 180°.
For the yaw-axis: Motion to the right is positive (0-180)
For the pitch-axis: Upward motion is positive (0-180)
For the roll-axis: Motion downward and to the right is positive (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. mobile device (X) axis acceleration

(1) Objective: Obtains the exact rate of acceleration for a mobile device
(2) Type: Information block (variable-type)
(3) Example: Dynamic forward control
Wave your mobile device up and down to control the robot's motion in a forward direction.



Note:
1) The faster the robot's velocity changes, the higher the rate of acceleration that will be obtained.
2) Mobile device refers to devices such as smartphones and tablets.

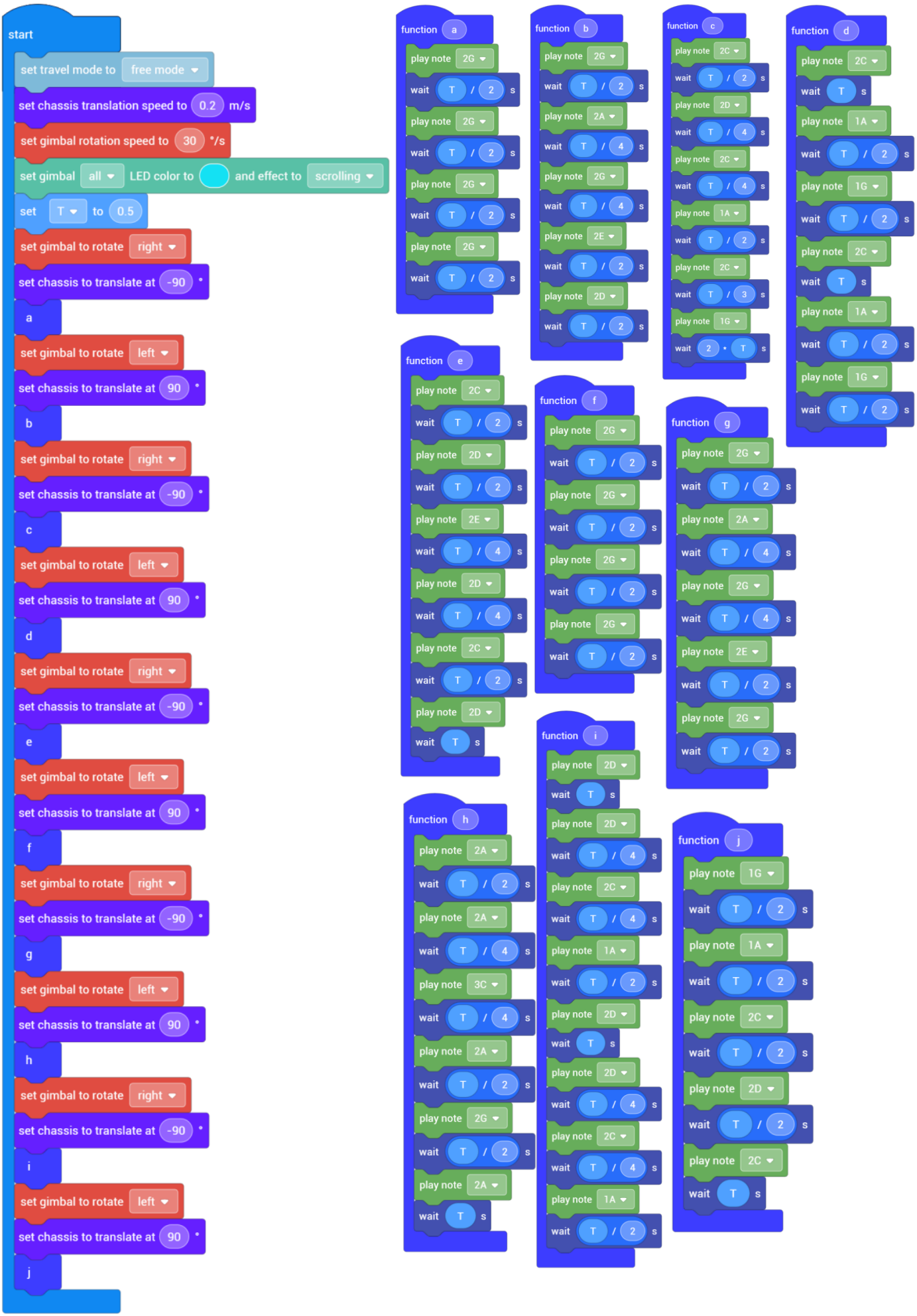
Python API:
Function: mobile_ctrl.get_accel(axis_enum)
Parameters:

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

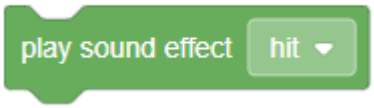
Media

1. play notes (1c)

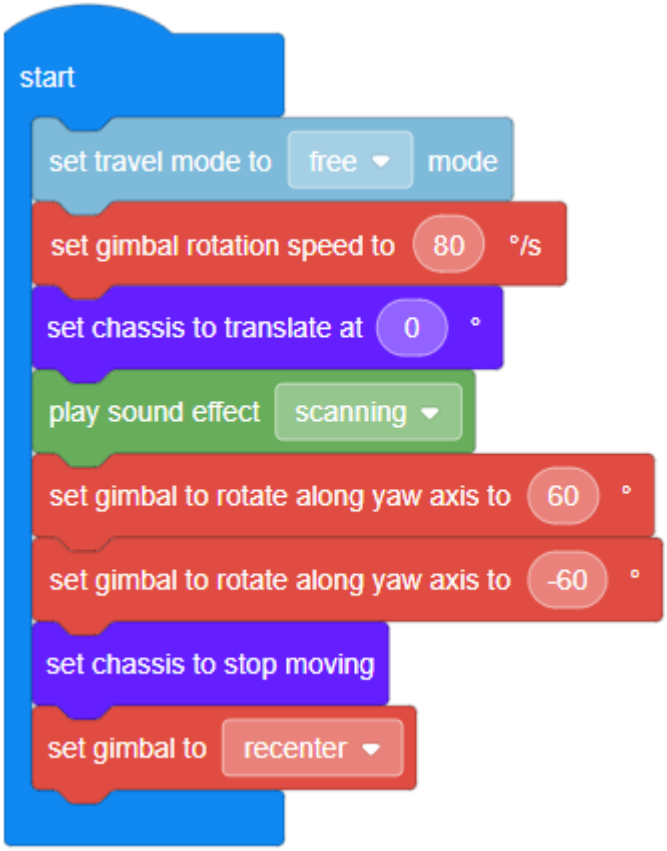
(1) Description: Choose notes and play them in a piano tone.
(2) Type: Execution, non-blocking
(3) Example: Pink Memory
Control the robot to play the notes of the song "Pink Memory".


start
set travel mode to free mode
set chassis translation speed to 0.2 m/s
set gimbal rotation speed to 30 °/s
set gimbal all LED color to and effect to scrolling
set T to 0.5
set gimbal to rotate right
set chassis to translate at -90 °
a
set gimbal to rotate left
set chassis to translate at 90 °
b
set gimbal to rotate right
set chassis to translate at -90 °
c
set gimbal to rotate left
set chassis to translate at 90 °
d
set gimbal to rotate right
set chassis to translate at -90 °
e
set gimbal to rotate left
set chassis to translate at 90 °
f
set gimbal to rotate right
set chassis to translate at -90 °
g
set gimbal to rotate left
set chassis to translate at 90 °
h
set gimbal to rotate right
set chassis to translate at -90 °
i
set gimbal to rotate left
set chassis to translate at 90 °
j
function a
play note 2G
wait T / 2 s
play note 2G
wait T / 2 s
play note 2G
wait T / 2 s
play note 2G
wait T / 2 s
function b
play note 2G
wait T / 2 s
play note 2A
wait T / 4 s
play note 2G
wait T / 4 s
play note 2E
wait T / 2 s
play note 2D
wait T / 2 s
function c
play note 2C
wait T / 2 s
play note 2D
wait T / 4 s
play note 2C
wait T / 4 s
play note 1A
wait T / 2 s
play note 2C
wait T / 3 s
play note 1G
wait 2 * T s
function d
play note 2C
wait T s
play note 1A
wait T / 2 s
play note 1G
wait T / 2 s
play note 2C
wait T s
play note 1A
wait T / 2 s
play note 1G
wait T / 2 s
function e
play note 2C
wait T / 2 s
play note 2D
wait T / 2 s
play note 2E
wait T / 4 s
play note 2D
wait T / 4 s
play note 2C
wait T / 2 s
play note 2D
wait T s
function f
play note 2G
wait T / 2 s
play note 2G
wait T / 2 s
play note 2G
wait T / 2 s
play note 2G
wait T / 2 s
function g
play note 2G
wait T / 2 s
play note 2A
wait T / 4 s
play note 2G
wait T / 4 s
play note 2E
wait T / 2 s
play note 2G
wait T / 2 s
function h
play note 2A
wait T / 2 s
play note 2A
wait T / 4 s
play note 3C
wait T / 4 s
play note 2A
wait T / 2 s
play note 2G
wait T / 2 s
play note 2A
wait T s
function i
play note 2D
wait T s
play note 2D
wait T / 4 s
play note 2C
wait T / 4 s
play note 1A
wait T / 2 s
play note 2D
wait T s
play note 2D
wait T / 4 s
play note 2C
wait T / 4 s
play note 1A
wait T / 2 s
function j
play note 1G
wait T / 2 s
play note 1A
wait T / 2 s
play note 2C
wait T / 2 s
play note 2D
wait T / 2 s
play note 2C
wait T s

2. play sound effect (hit)



(1) Objective: Sets the EP to emit a sound and executes the next command
(2) Type: Execution block, Non-blocking block
(3) Example: Emit scanning tone
This will set the robot to emit a “scanning” sound when the chassis is translating forward and the gimbal is rotating around the Yaw-axis.



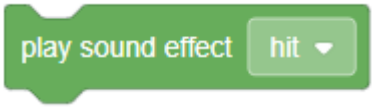
Python API：
Function: media_ctrl.play_sound(sound_enum, wait_complete_flag=False)
Parameters:

- 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. play sound (hit) until finished

(1) Objective: Ensures the next command will not be executed until a specified sound has been emitted
(2) Type: Execution block, Blocking block
(3) Example: Remote control photograph
When the EP identifies a “take photo” gesture, it will take a photo after playing the “start countdown” chime. If it does not identify a “take photo” gesture, it will continue to rotate and search for the gesture while playing a “scanning” sound.

Note:



“play sound” refers to simultaneous execution of the music and the next command, similar to a musical accompaniment;

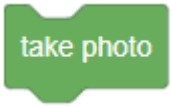
“play sound until finished” refers to one being executed after the other, making the sound more like a musical solo.

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

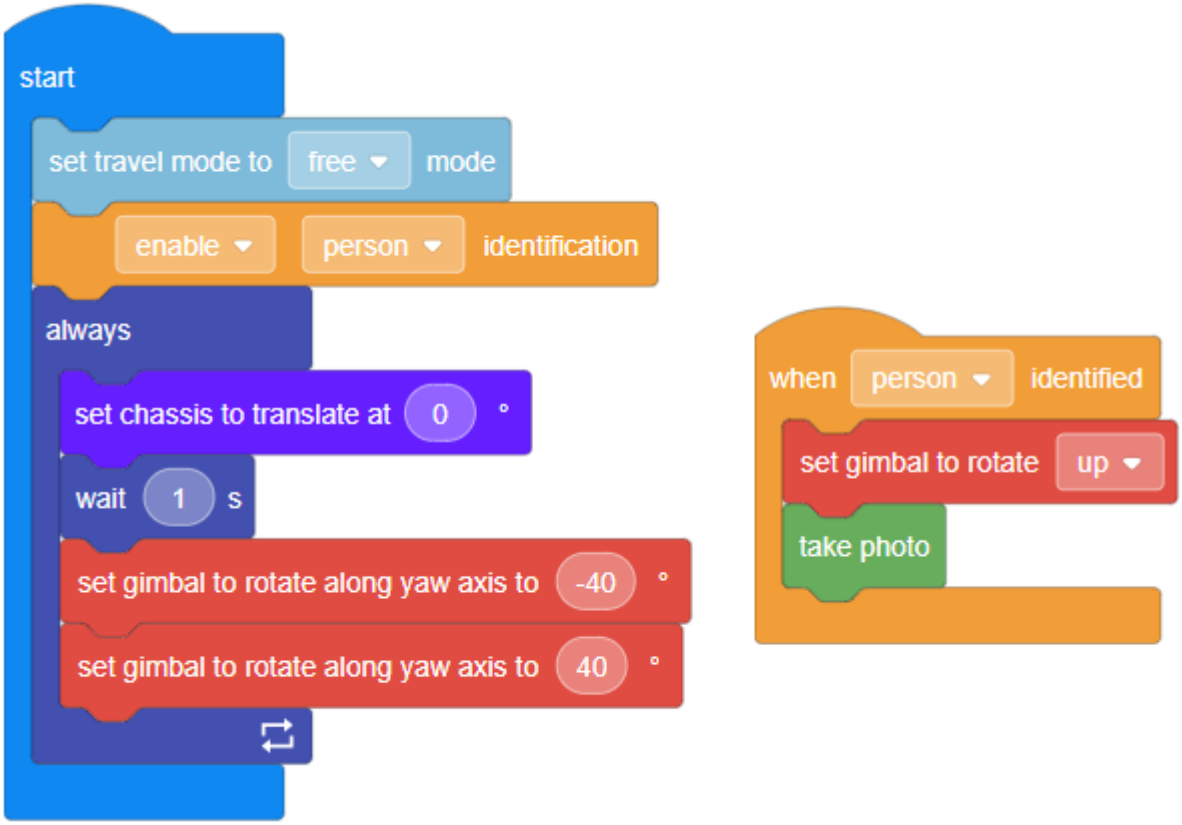
Parameters:

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

4. take photo



(1) Objective: Sets the robot to emit a specified shutter sound when it takes photos.
(Photos will appear in an album).
(2) Type: Execution block
(3) Example: Capture moving person

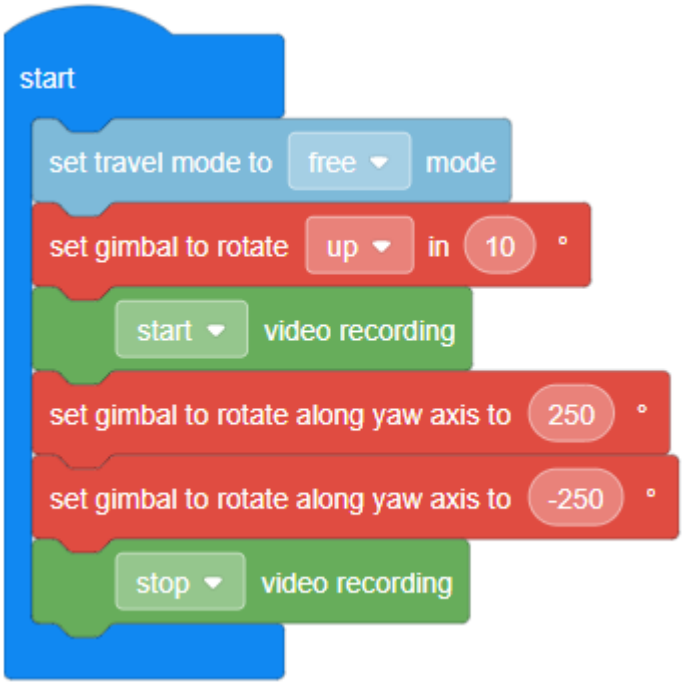


Note:
Users will need to insert an SD card with an available memory of more than 2G to operate the "take photo" command.

Python API:
Function: media_ctrl.capture()

5. (start) video recording

(1) Objective: Starts/stops recording; recorded videos will be saved in the SD card
(2) Type: Execution block
(3) Example: Video recording



Note:
Users will need to insert an SD card with an available memory of more than 2G to operate the “(start) video recording” command.

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

- enable_enum(enum):
  - 1
  - 0

Commands

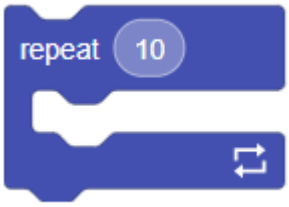
1. wait (1) s

(1) Objective: Executes the next command after waiting for a specified duration
(2) Type: Execution block
(3) Example: Complete a square
This will command the EP to translate in the following order to complete a square every two seconds by moving as translating: “right→forward→left→backward.”

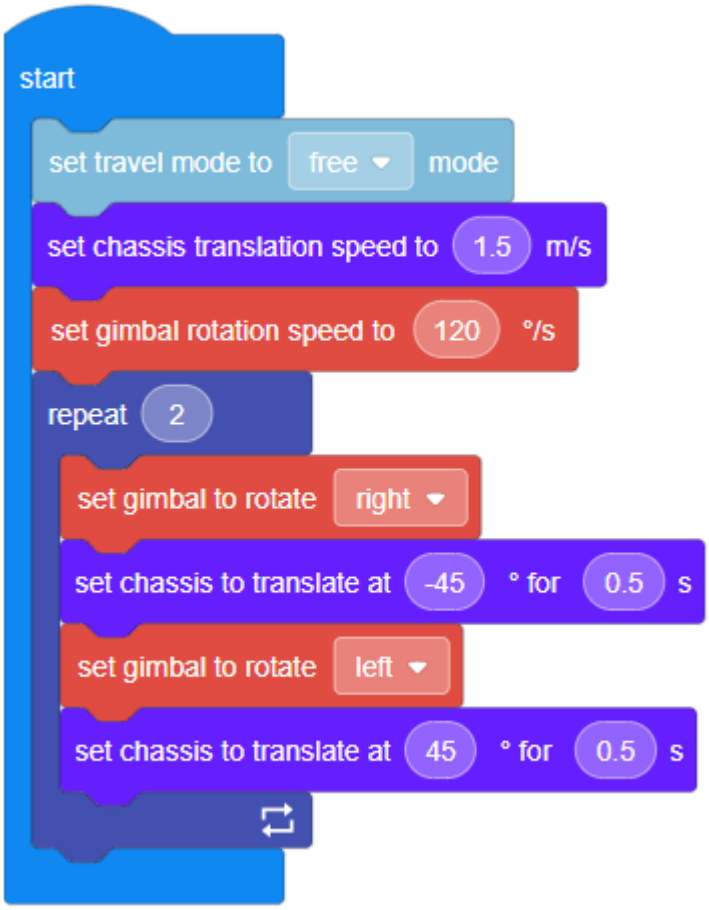
Python API
Function: time.sleep(t)
Parameters:

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

2. repeat (10)

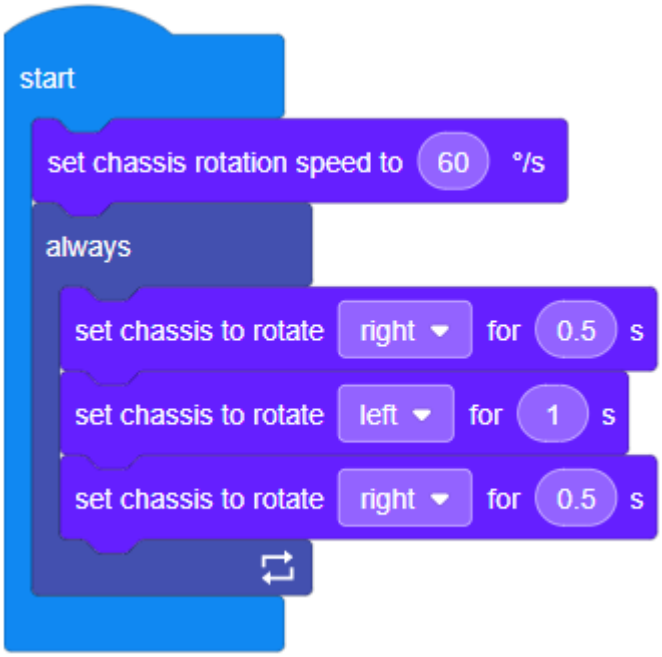


(1) Objective: Configures a program to repeat the same command several times (finite loop block)
(2) Type: Execution block
(3) Example: Translate along a zigzag line
This will control the robot to translate along a zigzag line.

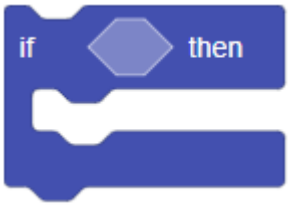


3. always

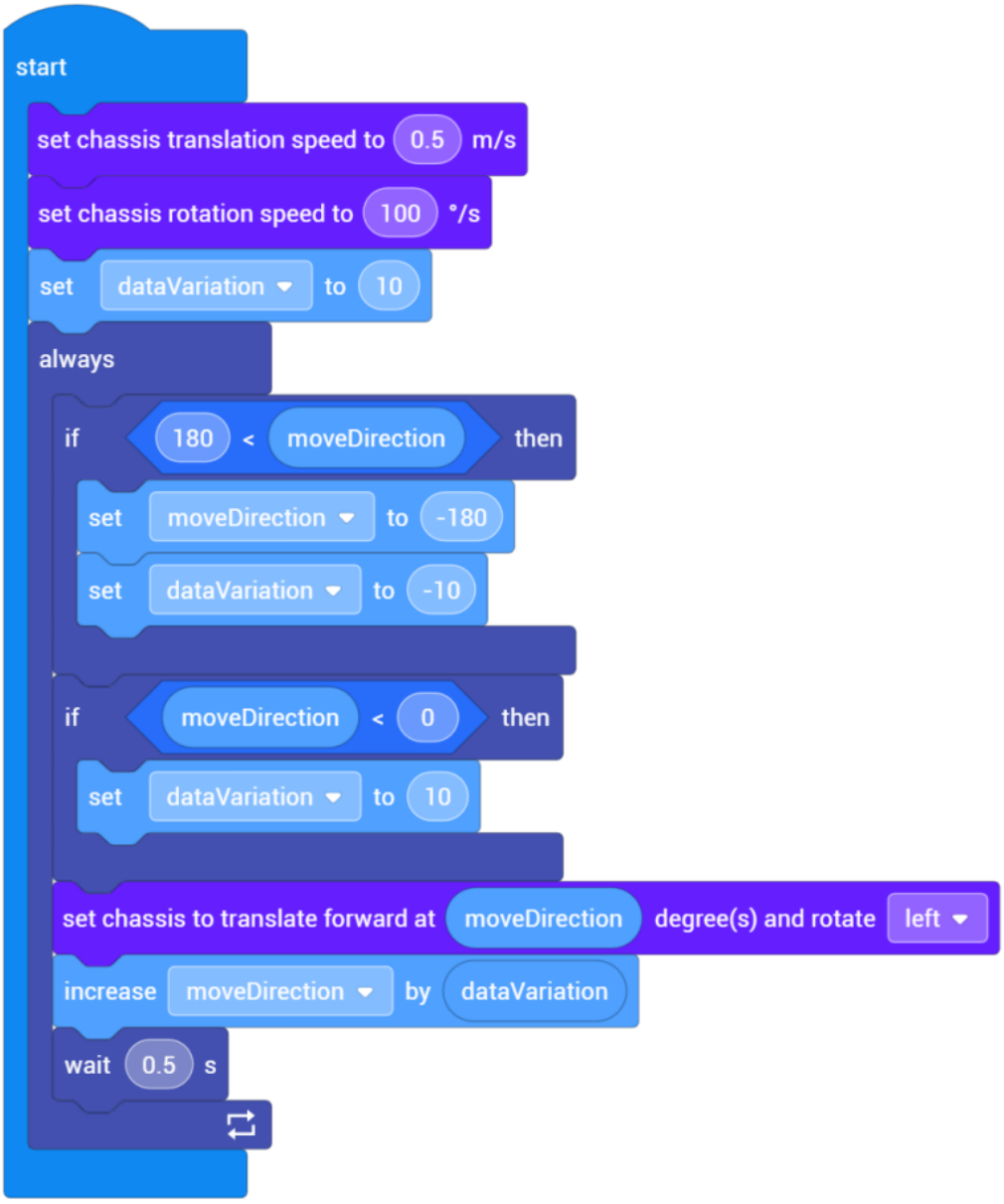
(1) Objective: Sets an internal program to repeat continuously (infinite loop)
(2) Type: Execution block
(3) Example: Rotate chassis left and right



4. If... then...

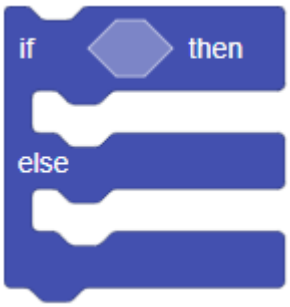


(1) Objective: Runs an internal program when the condition is “True”
(2) Type: Conditional statement block
(3) Example: Rotate and revolve
This will control the robot to revolve while rotating



Note:
The difference between conditional statement blocks and event blocks is that event blocks have higher priority while running the program when certain conditional statements are met regardless of the main function's status, whereas conditional statement blocks will not run unless the main function preceding it and corresponding conditions are met.

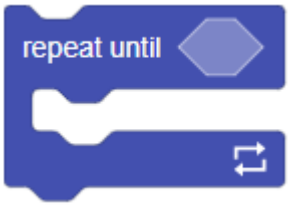
5. If... then... else...



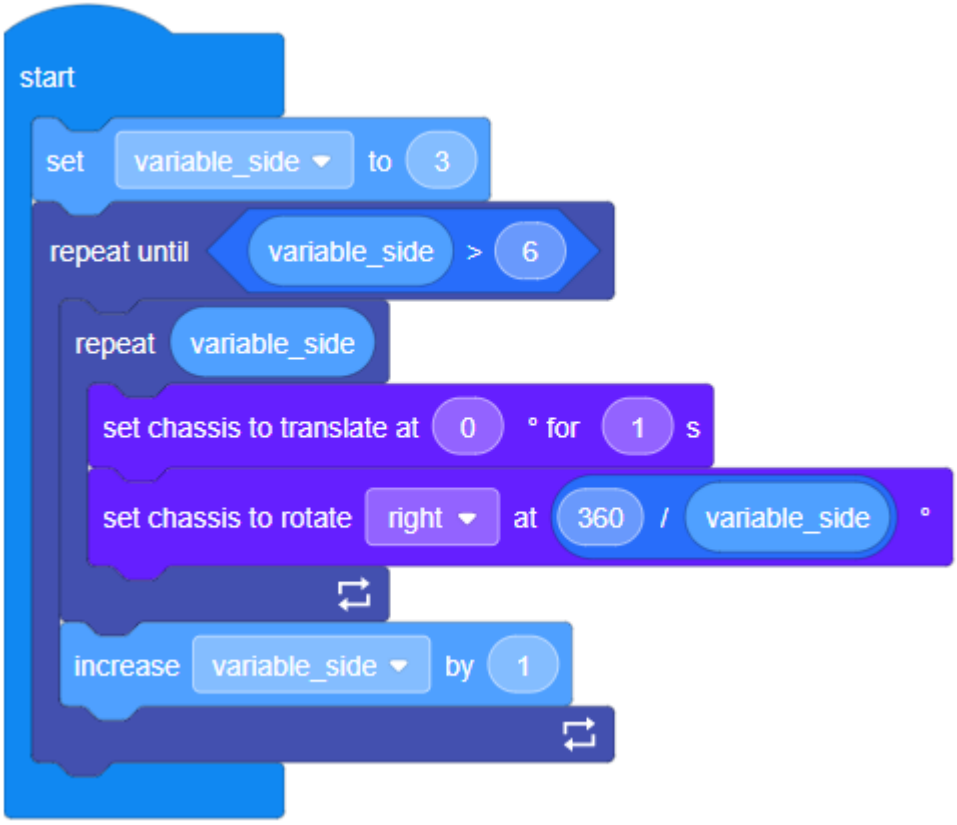
(1) Objective: Runs the "then" program when the condition is "True," and runs the "else" program otherwise
(2) Type: Conditional statement block

(3) Example: Limit the number of shots
This will control the robot to gradually increase the number of shots (with the maximum shots being "8 beads/time"), then restart from "1 bead/time" again.

6. repeat until...

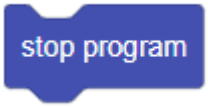


(1) Objective: Repeats a program until the condition is “True,” and then executes the next command
(2) Type: Conditional statement block
(3) Example: Complete different paths
This will control the robot to complete paths based on the following shapes: triangle, square, pentagon, and hexagon.

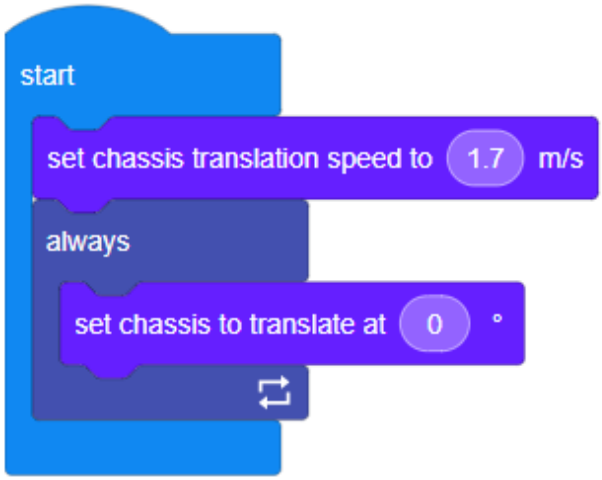


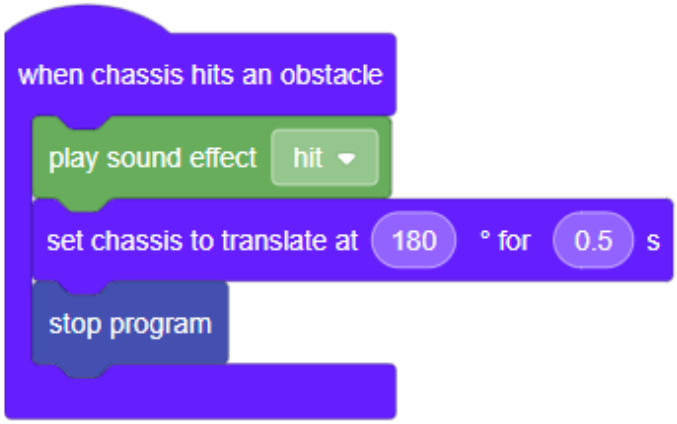
Note:
Strategize and plan the criteria for each path to obtain the ideal number of loops.

7. stop program



(1) Objective: Stops a running program for the current block and exits
(2) Type: Execution block
(3) Example: Stop a running program
This will control the chassis to translate backward for 0.5 second and then stop the program when it hits an obstacle.



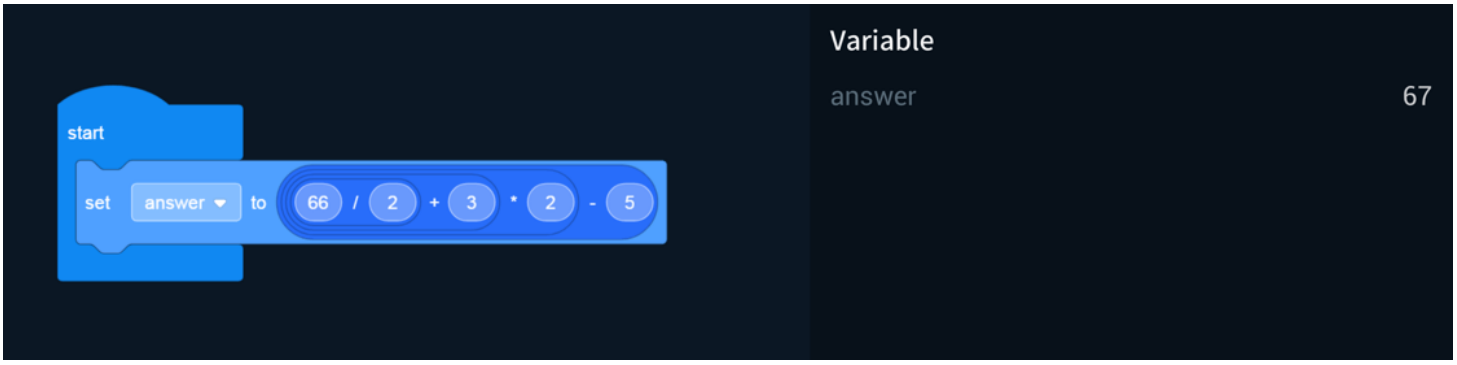


Operators

1. (O) + (O)

(1) Objective: Adds two numbers
(2) Type: Variable-type block
(3) Example: Perform basic arithmetic calculations
This will calculate 【(66÷2) +3】×2-5=?

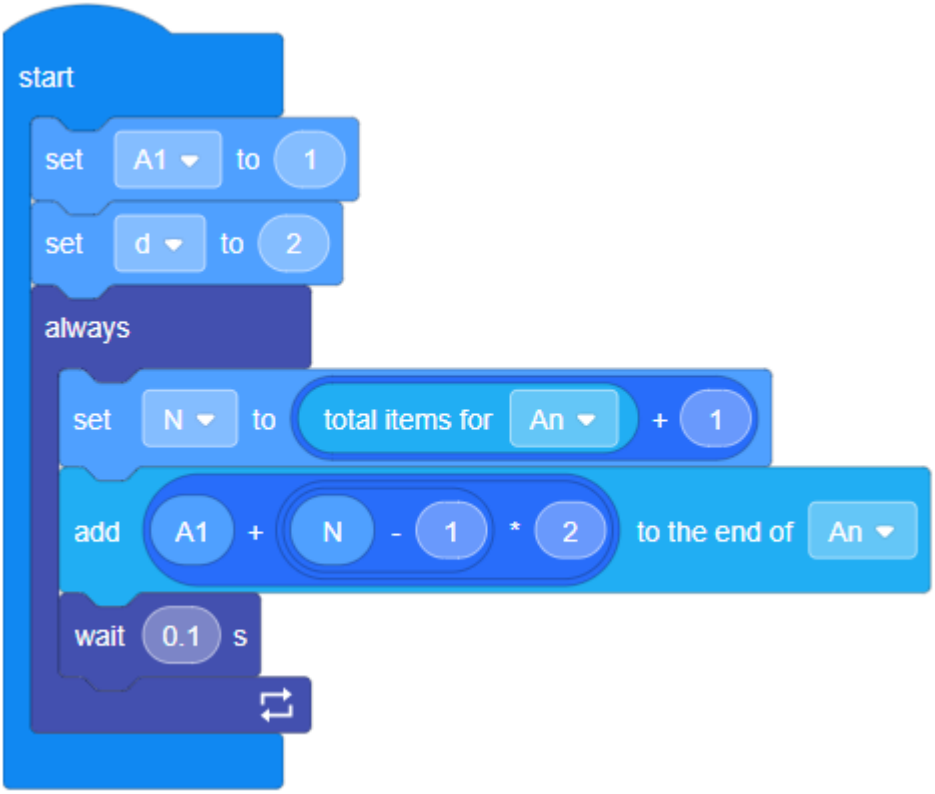
You can check the result using the FPV window:



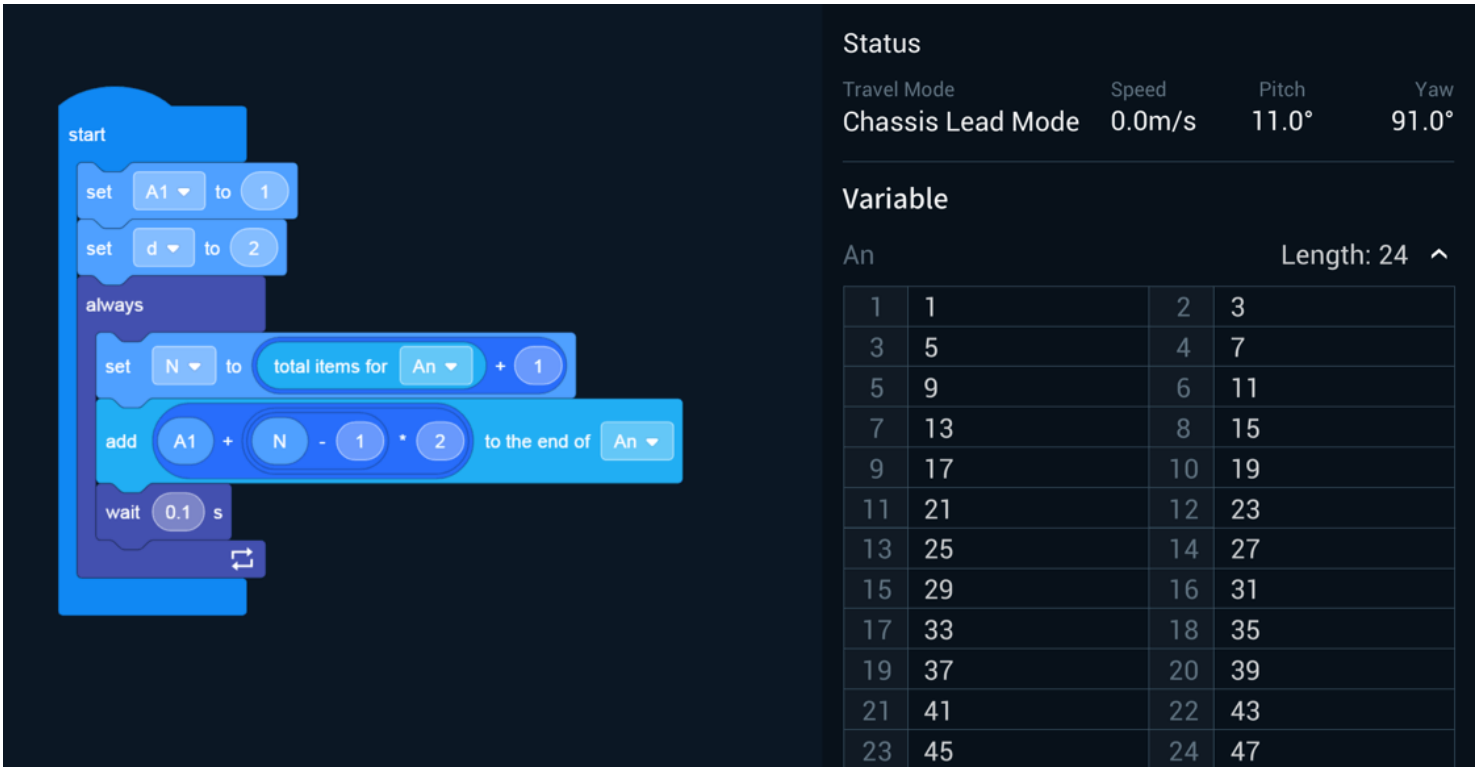
Note:
Arithmetic calculations cannot be performed on lists.

2. (O) - (O)

(1) Objective: Subtracts one number from another
(2) Type: Variable-type block
(3) Example: Calculate an arithmetic progression
This will calculate an arithmetic progression according to the formula An=A1+ (n-1) *d, with the first calculate item as 1 and a tolerance of 2.



You can check the result using the FPV window:



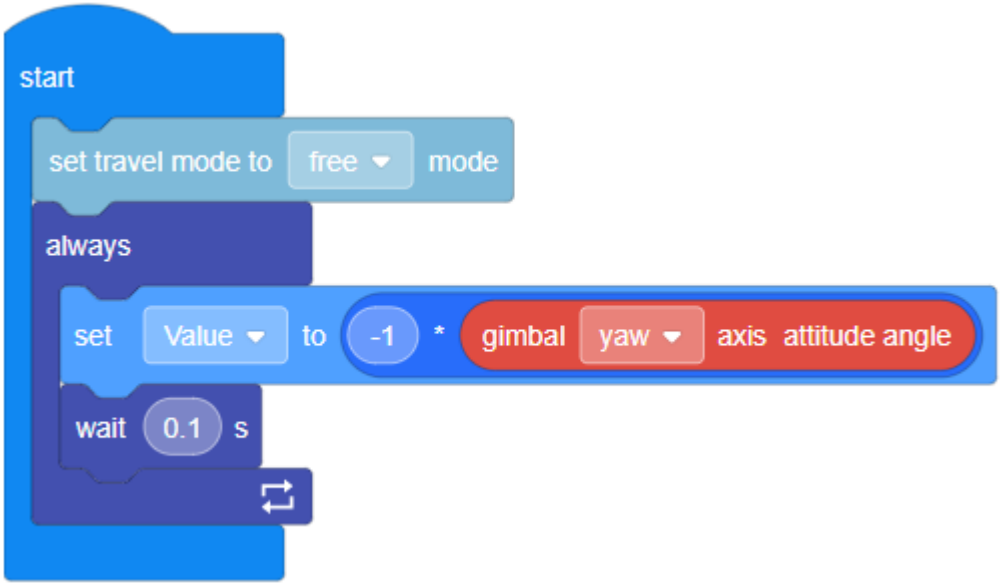
1	1	2	3
3	5	4	7
5	9	6	11
7	13	8	15
9	17	10	19
11	21	12	23
13	25	14	27
15	29	16	31
17	33	18	35
19	37	20	39
21	41	22	43
23	45	24	47

Note:
Arithmetic calculations cannot be performed on lists.

3. (0) * (0)

(1) Objective: Multiply two numbers

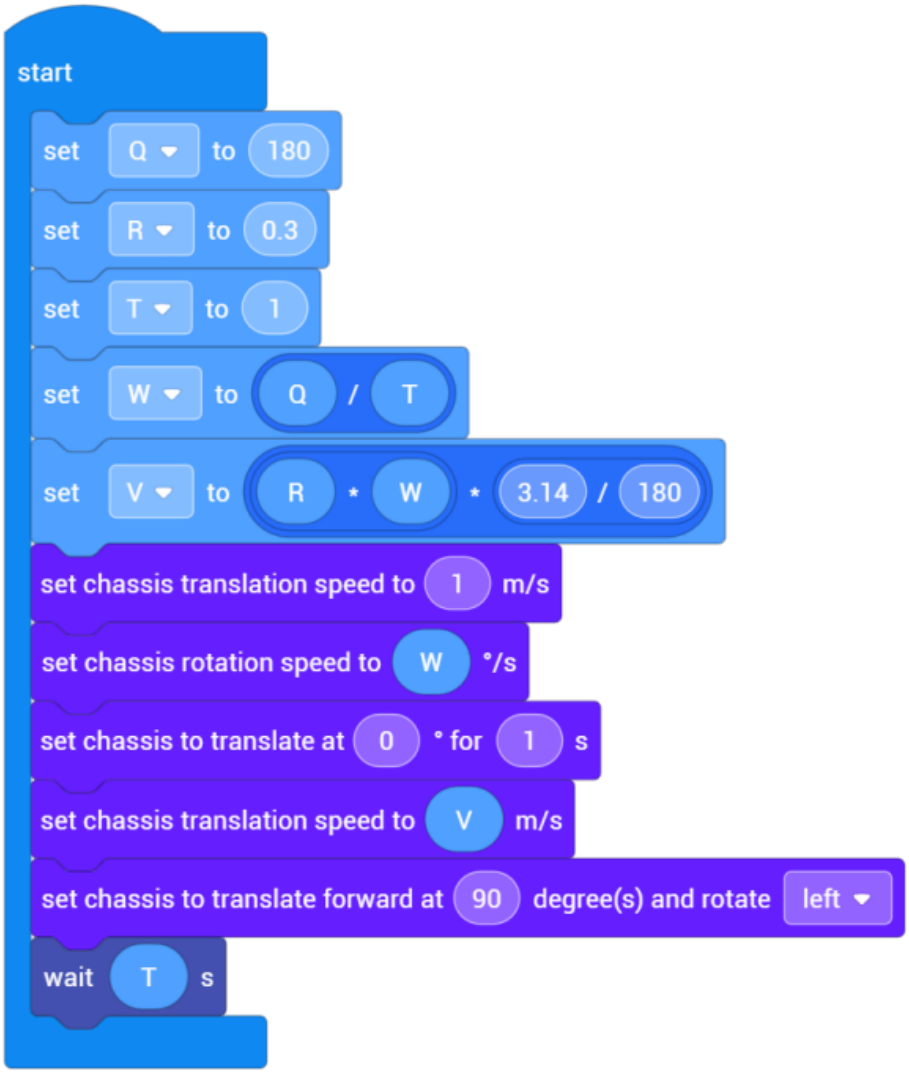
(2) Type: Variable-type block
(3) Example: Generate reverse display
This allows you to rotate the gimbal manually and observe how the value changes in the FPV window. Because this is the reverse display, when the gimbal is rotating on the right side of the chassis, the value will always be negative; when the gimbal is rotating on the left side of the chassis, the value will always be positive.



Note:
Arithmetic calculations cannot be performed on lists.

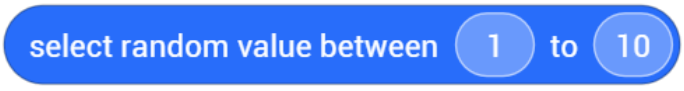
4. (0) / (0)

(1) Meaning: Divide one number by another
(2) Type: Variable-type block
(3) Example: Drift
This will control the robot to drift and pull up.

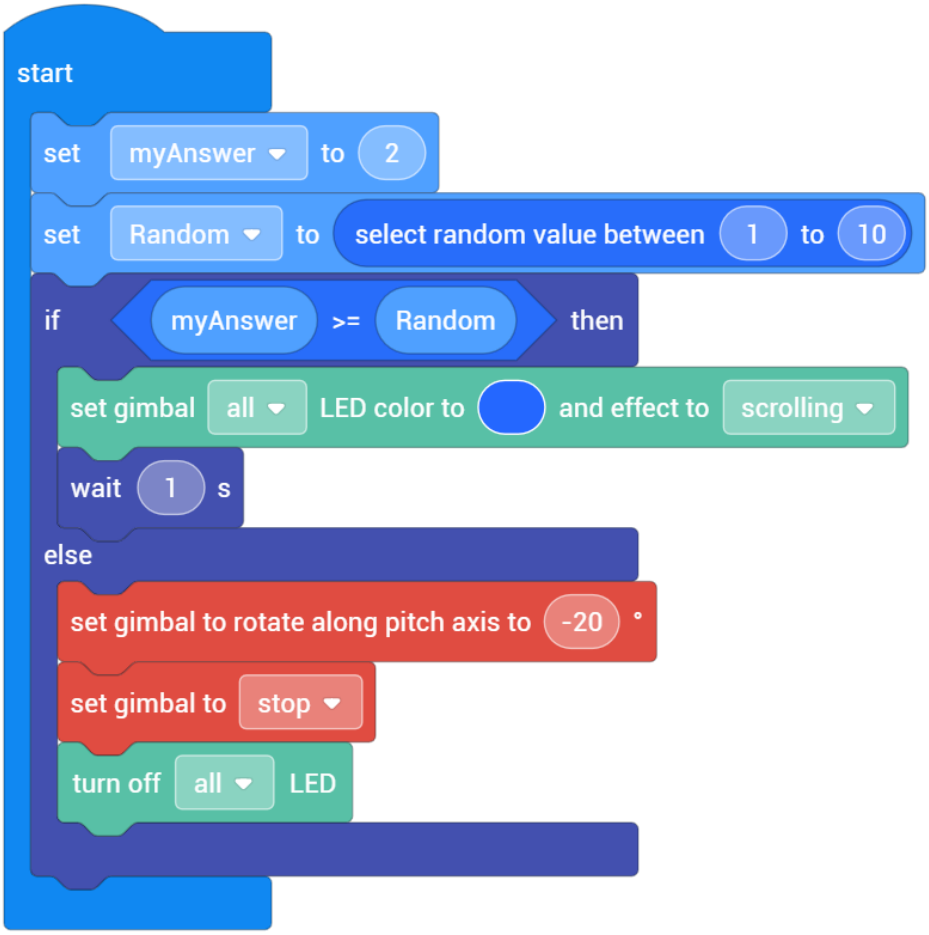


Note:
Arithmetic calculations cannot be performed on lists.

5. select random value between (1) to (10)

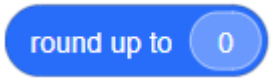


(1) Objective: Selects a random value from a specified range
(2) Type: Information block (variable-type)
(3) Example: Play a number guessing game
Guess a number to enter and compare it to the random value generated by the robot. If your answer is greater than or equal to the robot's number, the gimbal will celebrate with a lighting effect; if your answer is less than the number, the gimbal makes an effect to show disappointment.

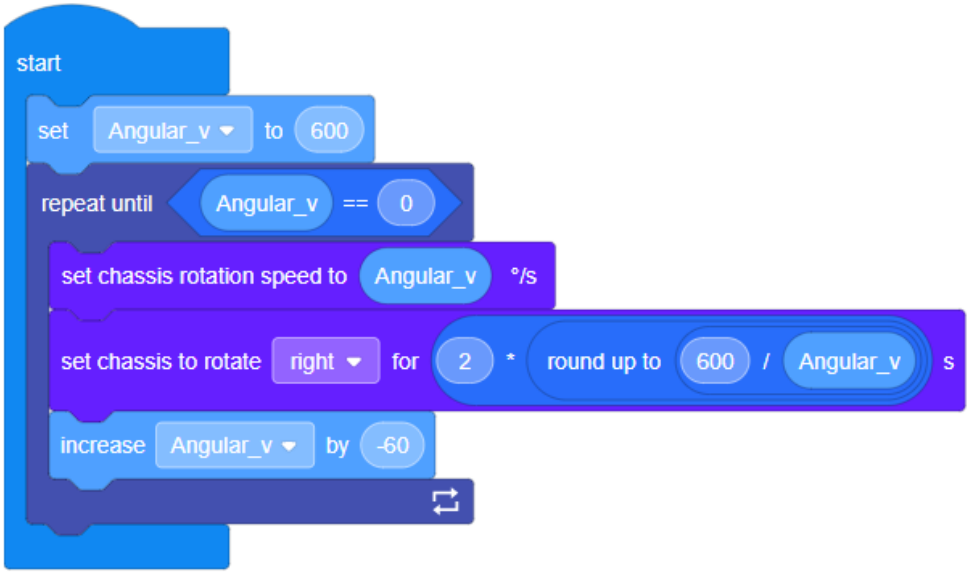


Note:
If the values you enter within the specified range are all integers, the output will always be an integer.
If any of the values you enter within the specified range is a decimal (whether it is the first, last, or any in between), the output will also be a decimal.

6. round up to (0)

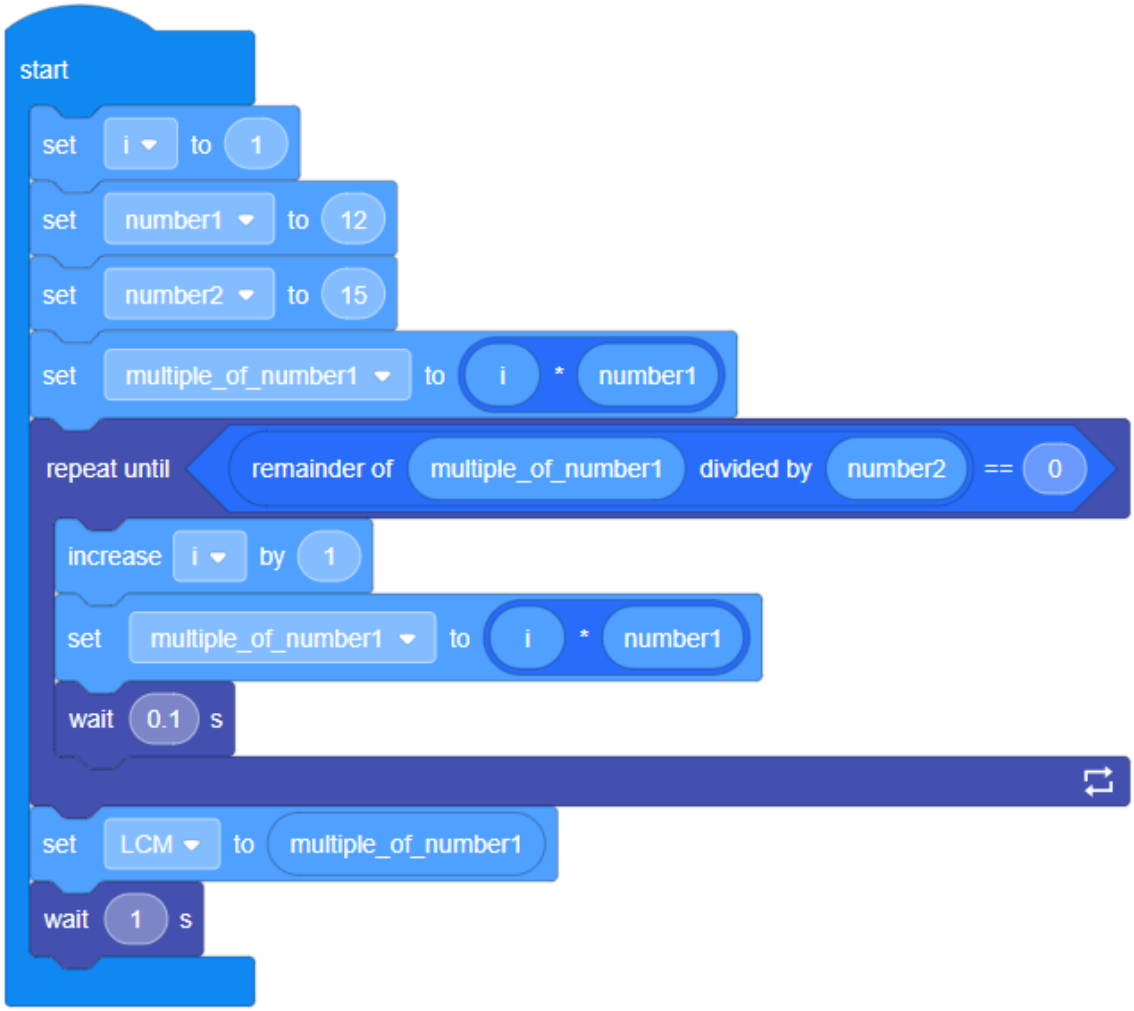


(1) Objective: Rounds up to obtain the nearest integer to a given value
(2) Type: Information block (variable-type)
(3) Example: Slow down chassis rotation
When the chassis rotates to the right, the rotation speed will decrease gradually from 600 degrees/sec to 60 degrees/sec.

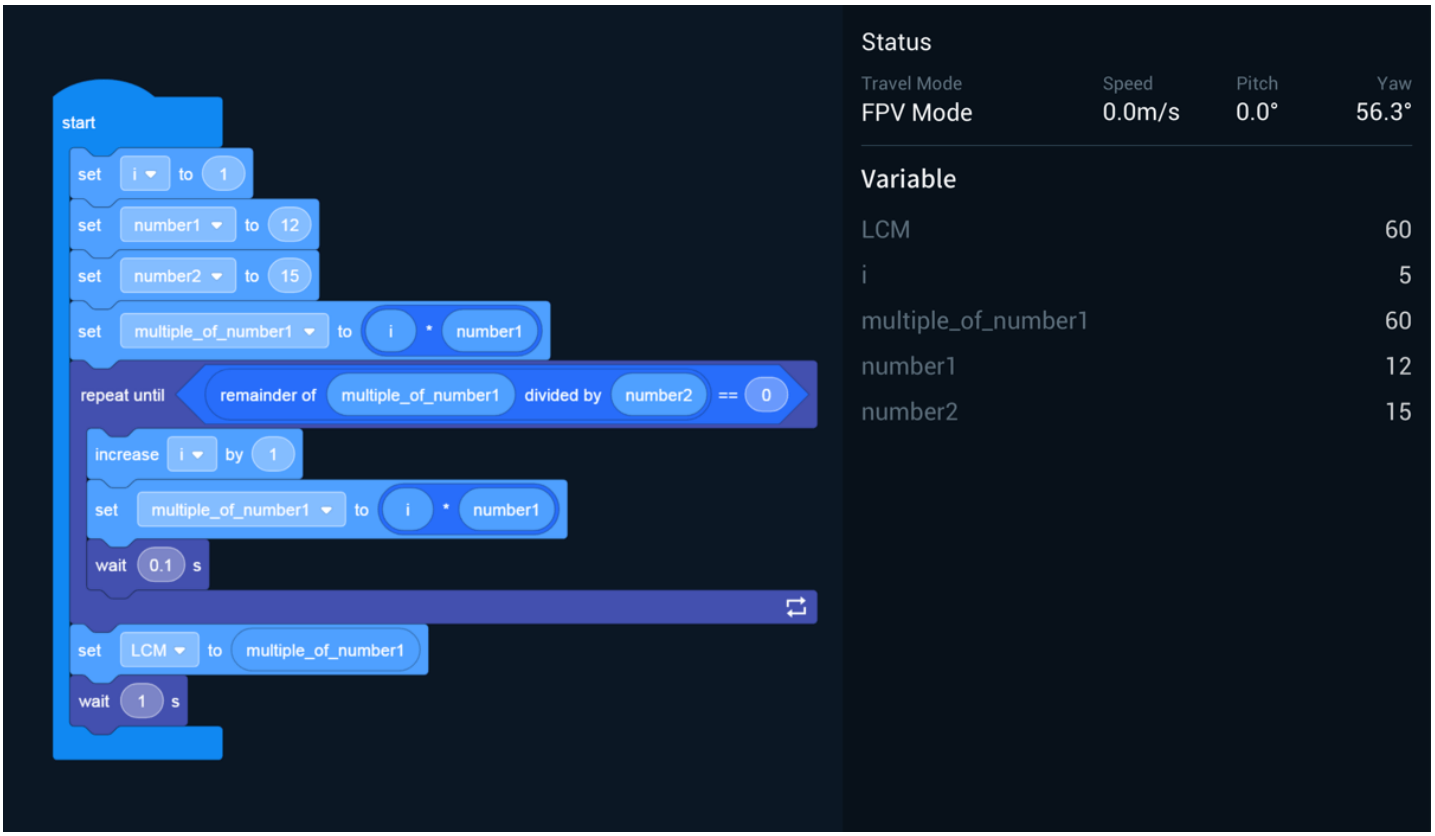


7. remainder of (0) divided by (1)

(1) Objective: Obtains the remainder of the first value divided by the second value (2) Type: Information block (variable-type data)
(3) Example: Find the lowest common multiple
The robot will find the lowest common multiple of 12 and 15.



You can check the result using the FPV window. (LCM=60)

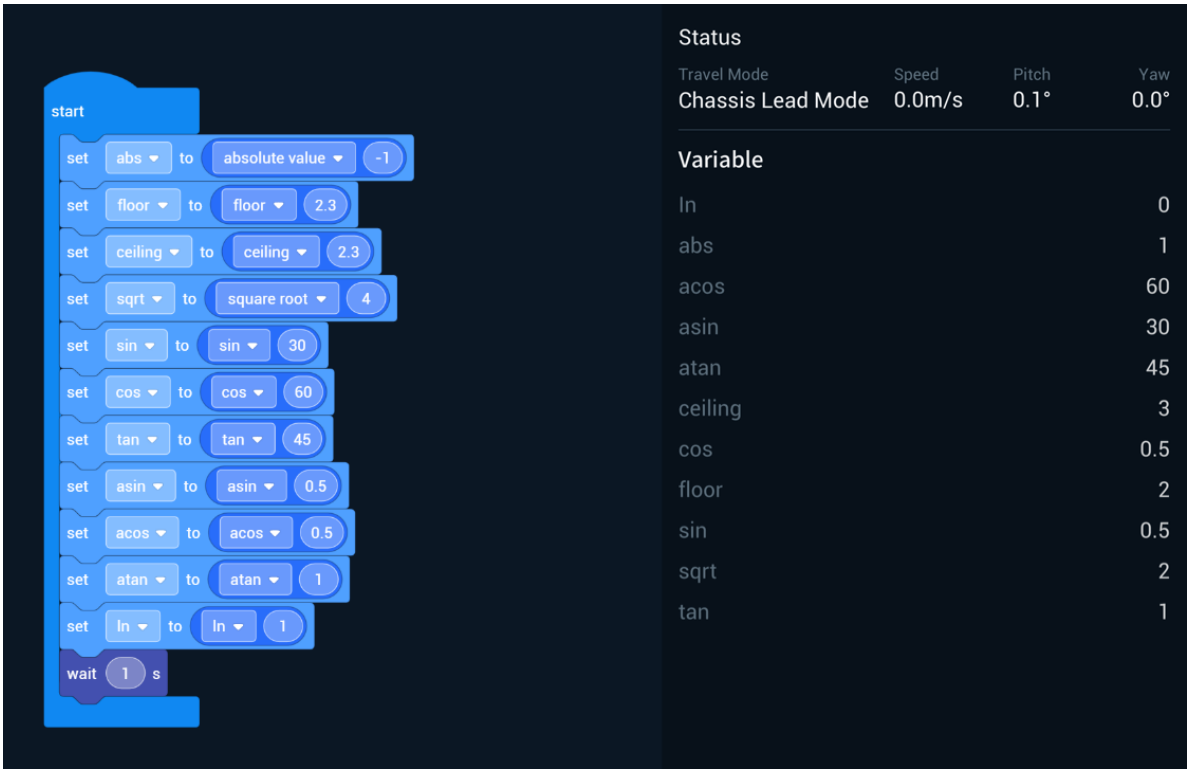


Note:
LCM is short for the lowest common multiple.

8. absolute value (0)

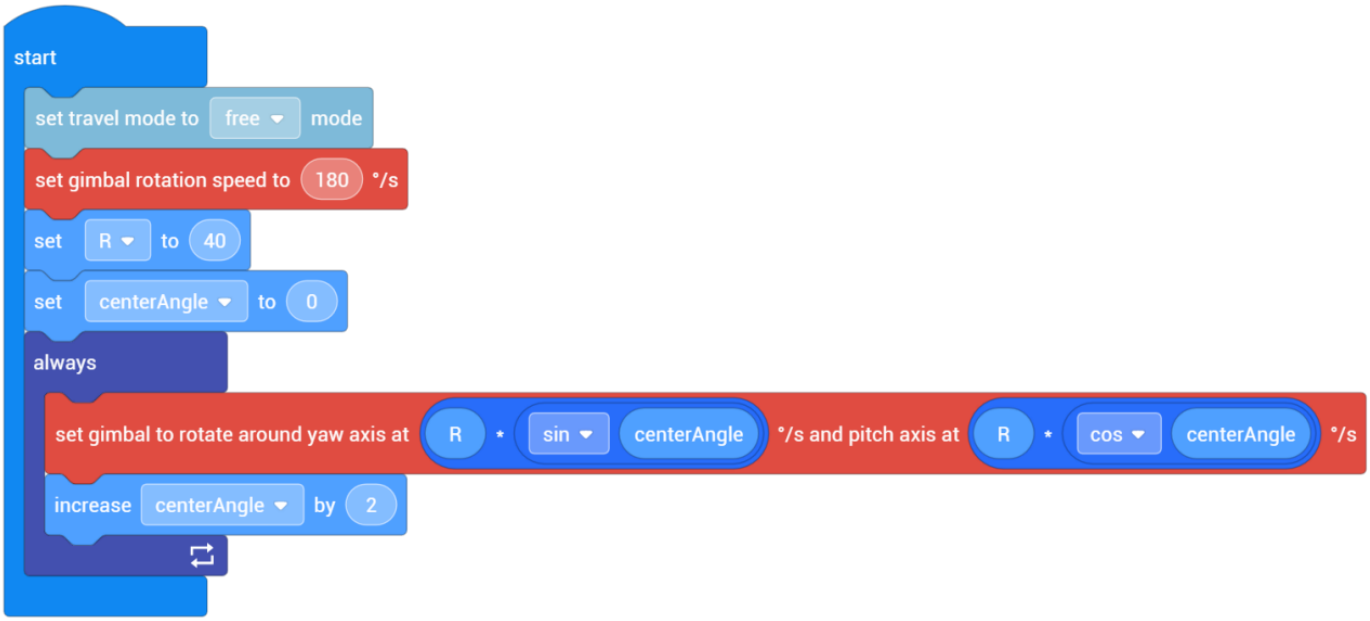
(1) Objective: Performs mathematical operations for a variety of functions, such as calculating absolute value, floor and ceiling functions, square root, and sine angle
(2) Type: Information block (variable-type)
(3) Examples: Perform calculation, Rotate gimbal
① Perform calculation

You can check results using the FPV window.



② Rotate gimbal

This will use the center angle and radius to calculate and control gimbal rotation.



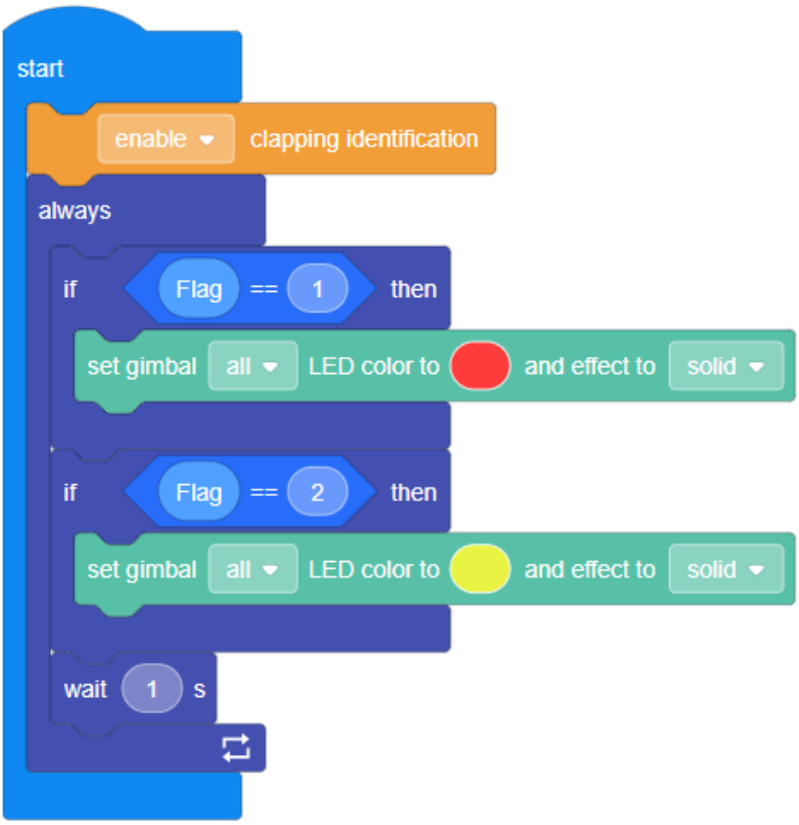
9. (0) == (0)

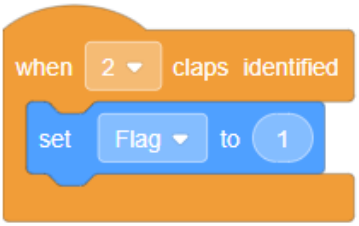
(1) Objective: Returns “True” when two values are equal; otherwise, “False” is returned

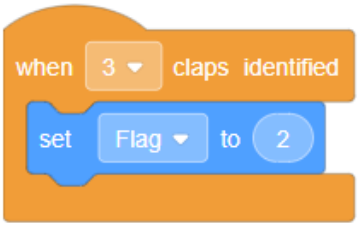
(2) Returned value: Boolean

(3) Example: Set flag bits

Initially, the gimbal displays a solid default color. All LEDs on the gimbal will turn solid red when the users clap two times, and solid yellow when the user claps three times.



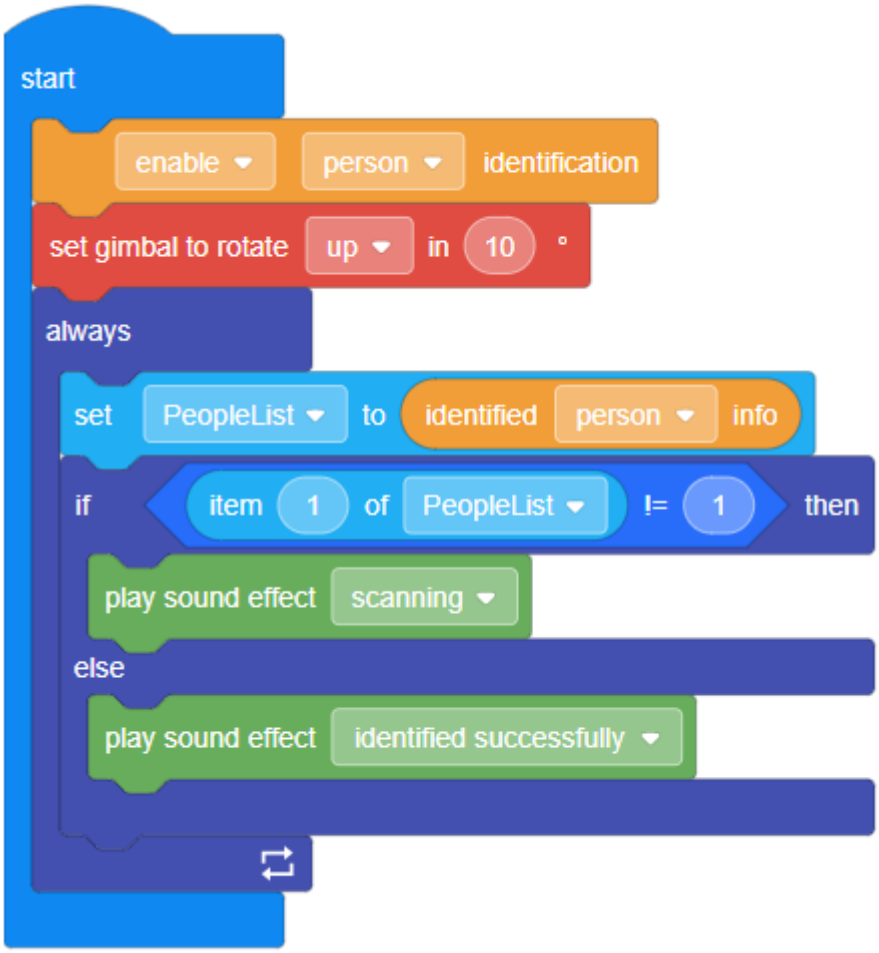




Note:
In programming, flag bits distinguish various states in order to make the robot execute different commands.

10. (0) != (0)

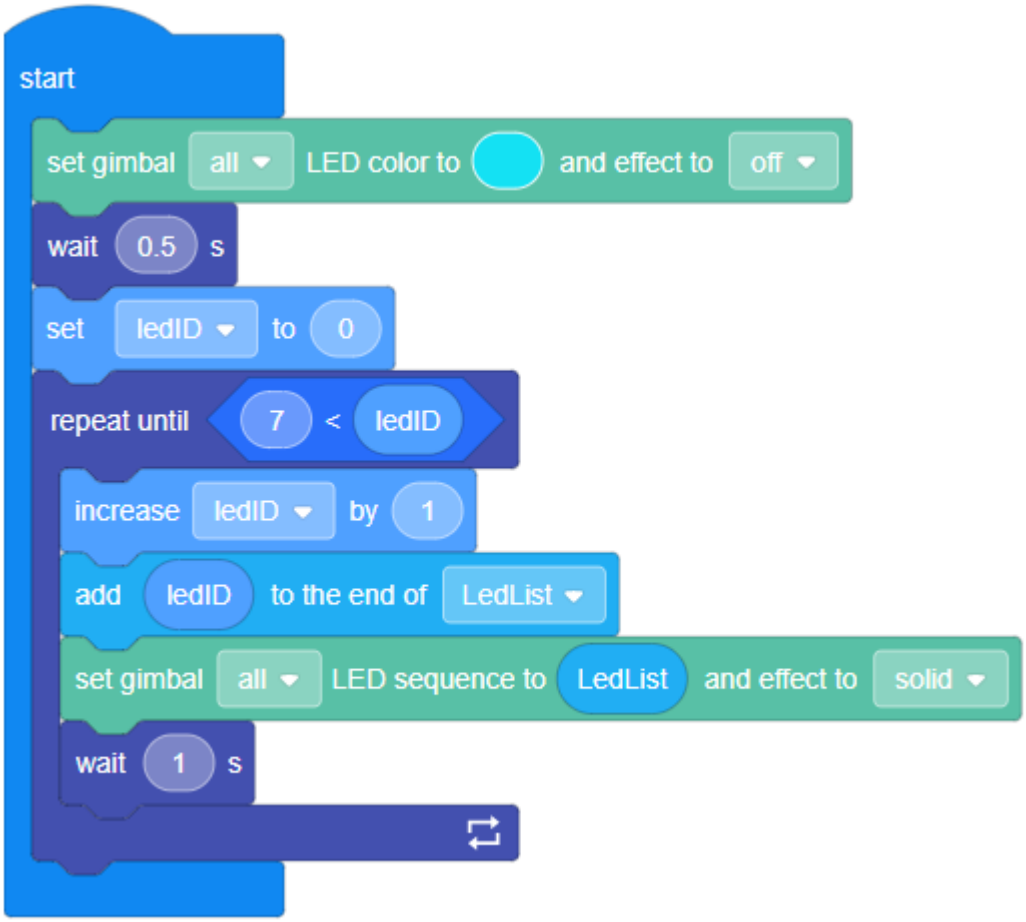
(1) Objective: Returns “True” when two values are not equal; otherwise, “False” is returned
(2) Returned Value: Boolean
(3) Example: Indicate successful identification of one person
If no person or multiple people appear in the robot's field of view, the “scanning” sound effect will play; if exactly one person is identified, however, the “identified successfully” sound will play.



Note:
"!=" is the symbol for "unequal to" in programming language and has the same meaning as "≠"in mathematical language.

11. (O) < (O)

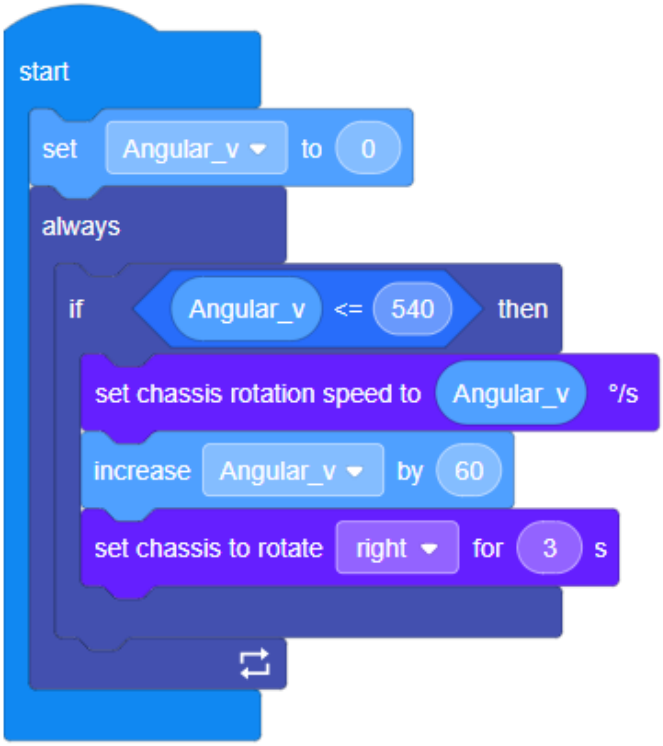
(1) Objective: Returns "True" if the first value is less than the second value; otherwise, "False" is returned.
(2) Returned value: Boolean
(3) Example: Light all platform LED lights in sequence
This will light up all gimbal LEDs in sequence.



Note:
This is often used with conditional statements.

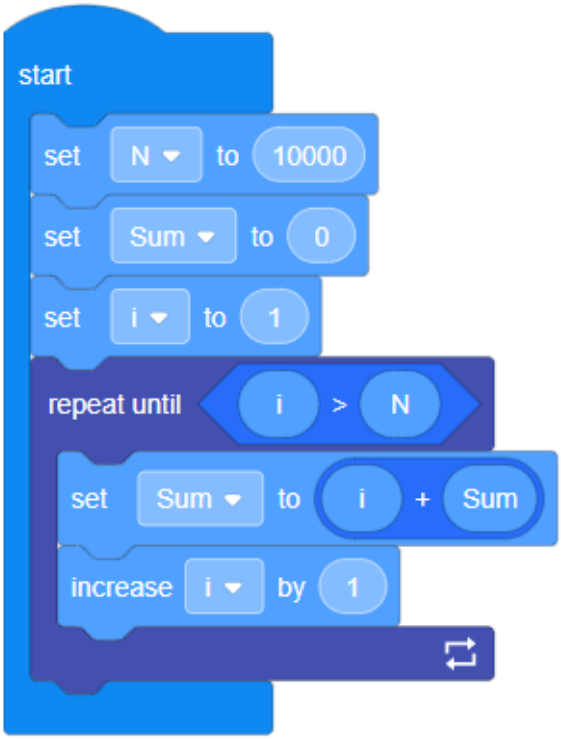
12. (0) <= (0)

(1) Objective: Returns the condition “True” if the first value is less than or equal to the second value; otherwise, “False” is returned.
(2) Returned value: Boolean
(3) Example: Accelerate rotation
If the chassis is rotating at a rate less than or equal to 540, it will increase by 60 degrees per second while rotating to the right for 3 seconds at a time until the maximum value of 600 degrees per second is achieved and acceleration stops.



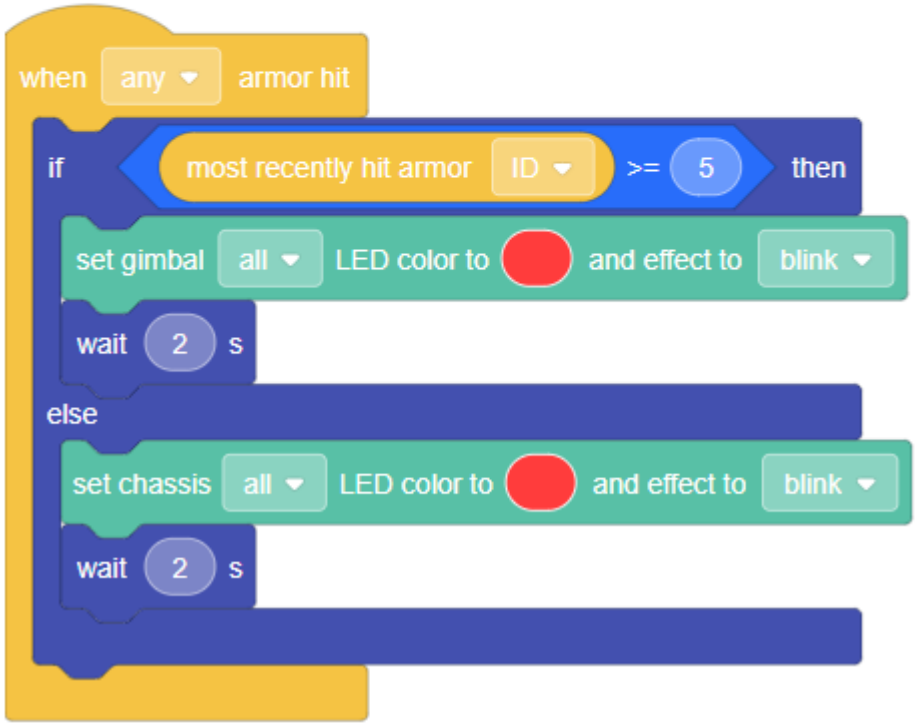
13. (0) < (0)

(1) Objective: Returns the condition “True” if the first value is larger than the second value; otherwise, “False” is returned.
(2) Returned value: Boolean
(3) Example: Calculate cumulative sum (1-10000)
This will calculate the accumulated sum of 1+2+3...+10000; you can check the result using the FPV window (displayed as sum=50005000).



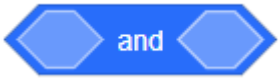
14. (0) >= (0)

(1) Objective: Returns the condition “True” if the first value is greater than or equal to the second value; otherwise, “False” is returned.
(2) Returned value: Boolean
(3) Example: Show where the armor was last hit
If the gimbal armor was hit last, all gimbal LEDs will blink red; if the chassis armor was hit last, all chassis LEDs will blink red.



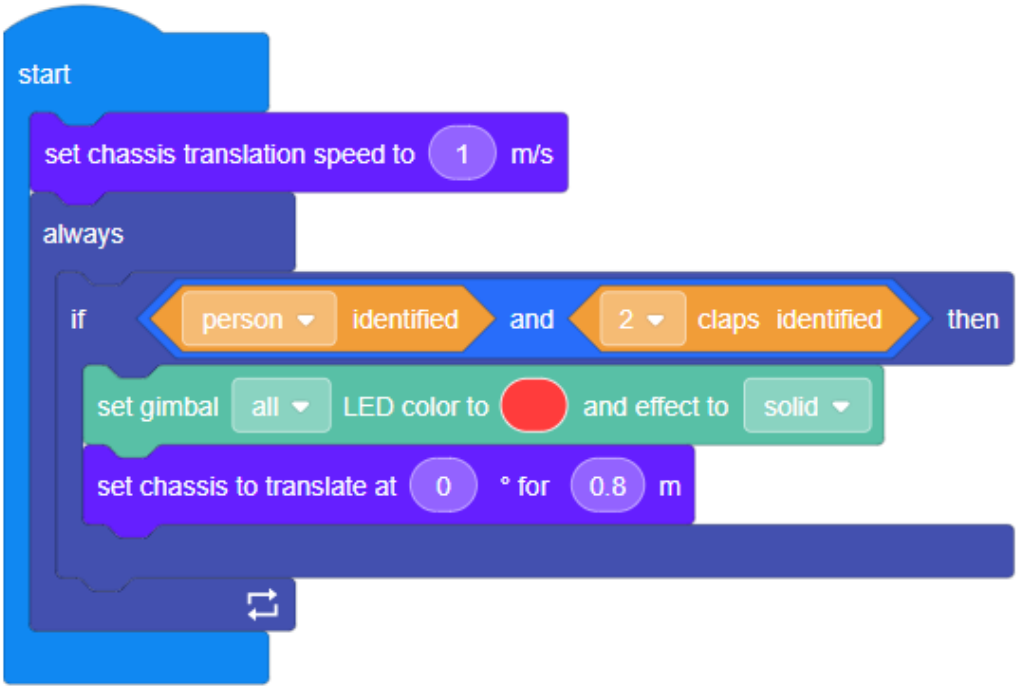
Note:
As in mathematical language, “>=” is the standard symbol for “greater than or equal to” in programming language.

15. () and ()



(1) Objective: Returns the condition “True” when two conditions are met; otherwise, the condition “False” is

returned.
(2) Returned value: Boolean
(3) Example: Clap to summon the robot
When a person stands one meter away from the robot and claps twice, the robot will approach. Both “Person” and “Clap two times” are essential conditions, and thus use “AND."

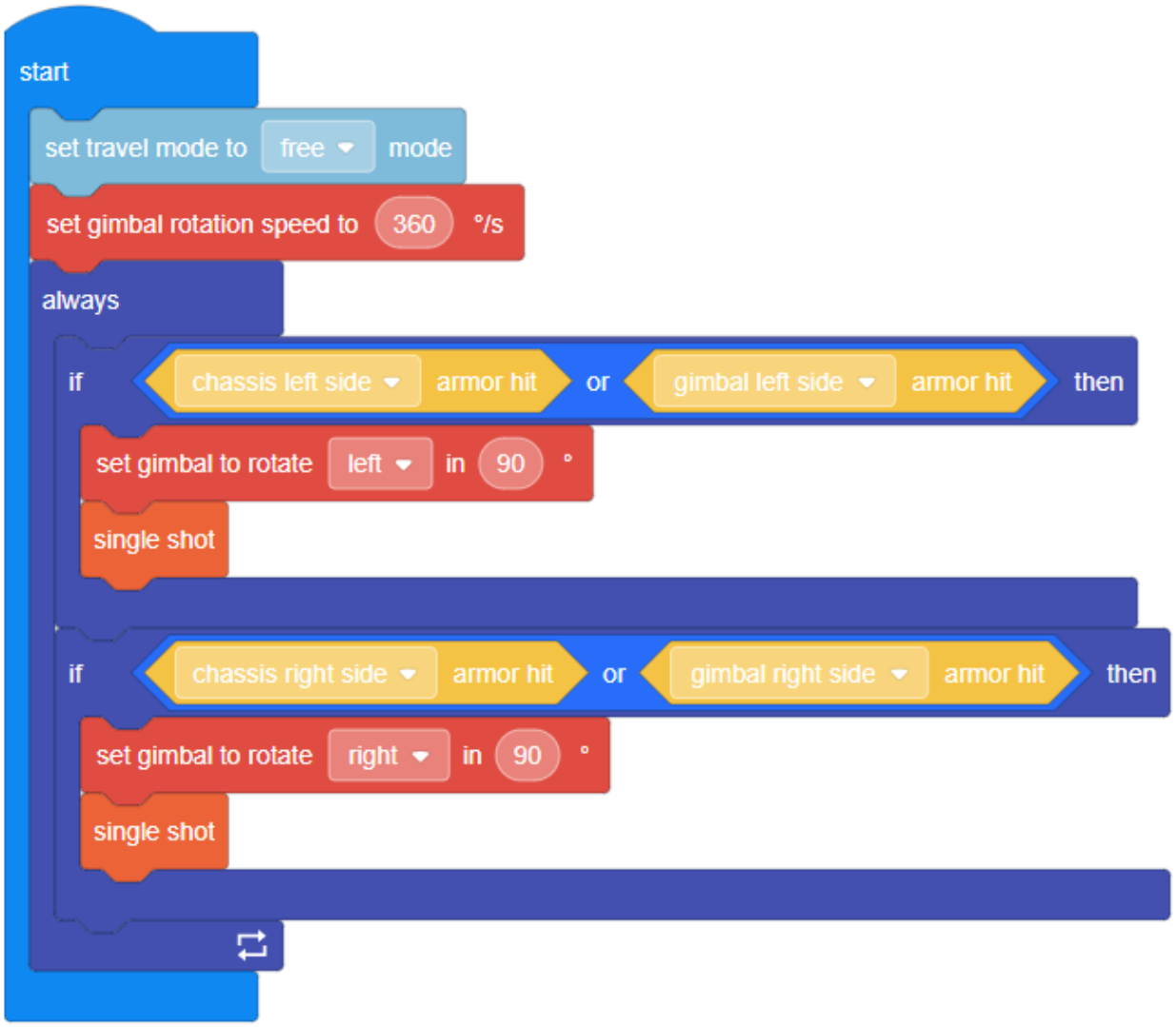


Note:
"and,” “or,” and “not" are logical operations, and the returned result will be a Boolean value: “True,” or “False.”
Boolean values:

A and B		
B \ A	Condition False	Condition True
Condition False	FALSE	FALSE
Condition True	FALSE	TRUE

16. () or ()

(1) Objective: Returns the condition “True” if either of a set of two conditions is met; “False” is returned if neither condition is met.
(2) Returned value: Boolean
(3) Example: Turn toward an enemy robot after being attacked
The gimbal will rotate to the left to fight back when the left side armor is struck. Because the “left side” here can be that of the gimbal or the chassis, the “OR” operation is used here.

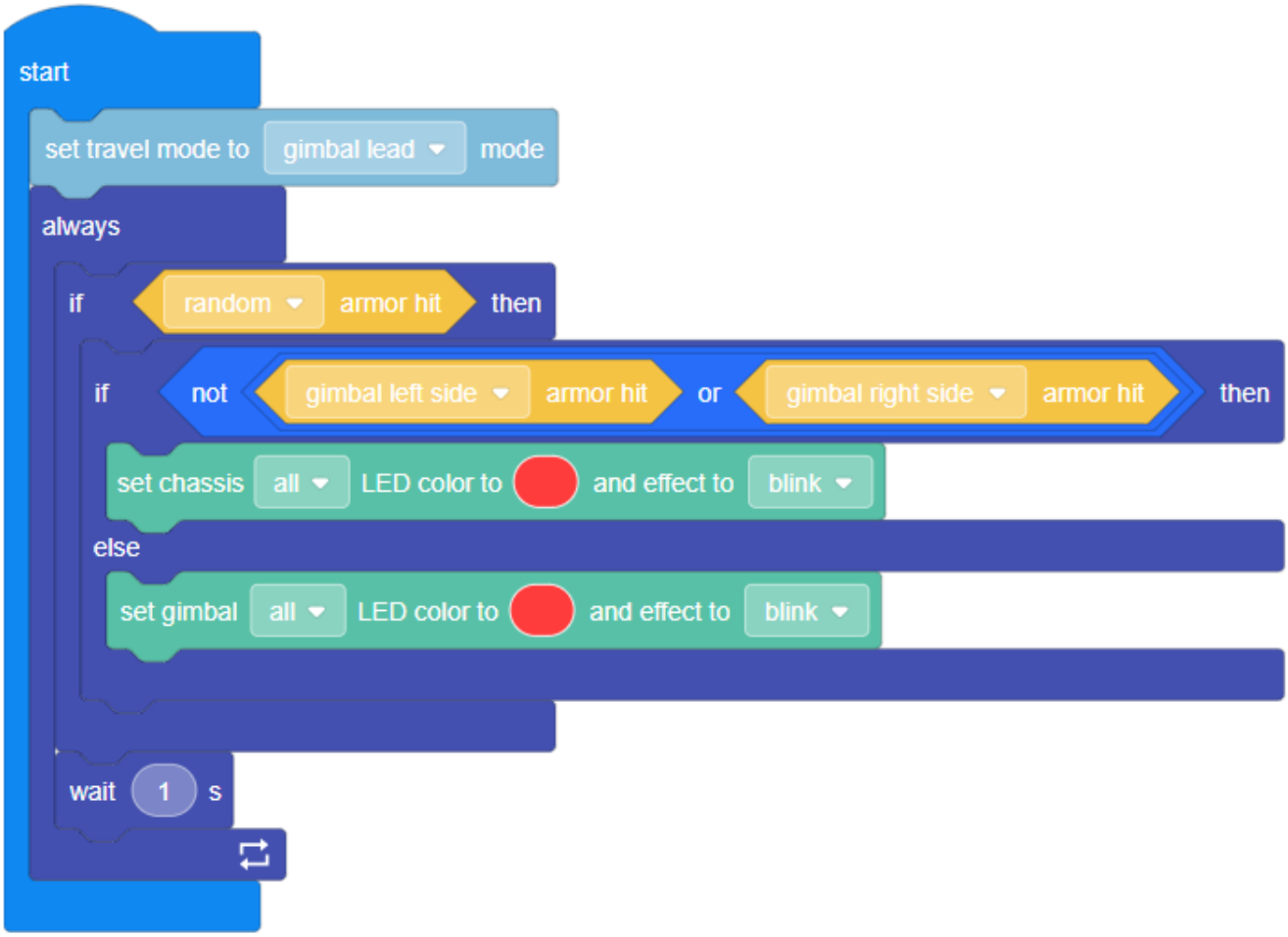


Note:
"and," "or," and "not" are logical operations, and the returned result will be a Boolean value: "True," or "False."
Boolean values:

A or B		
B \ A	Condition False	Condition True
Condition False	FALSE	TRUE
Condition True	TRUE	TRUE

17. not ()

(1) Objective: Returns "False" if a condition is met; otherwise, "True" is returned.
(2) Returned value: Boolean
(3) Example: Set either-or conditions
If any point of the gimbal armor is attacked, all gimbal LEDs will blink red; if any point of the chassis armor is attacked, all chassis LEDs will blink red. Here, when the attached armor is "not on gimbal," this indicates it must be "on the chassis."

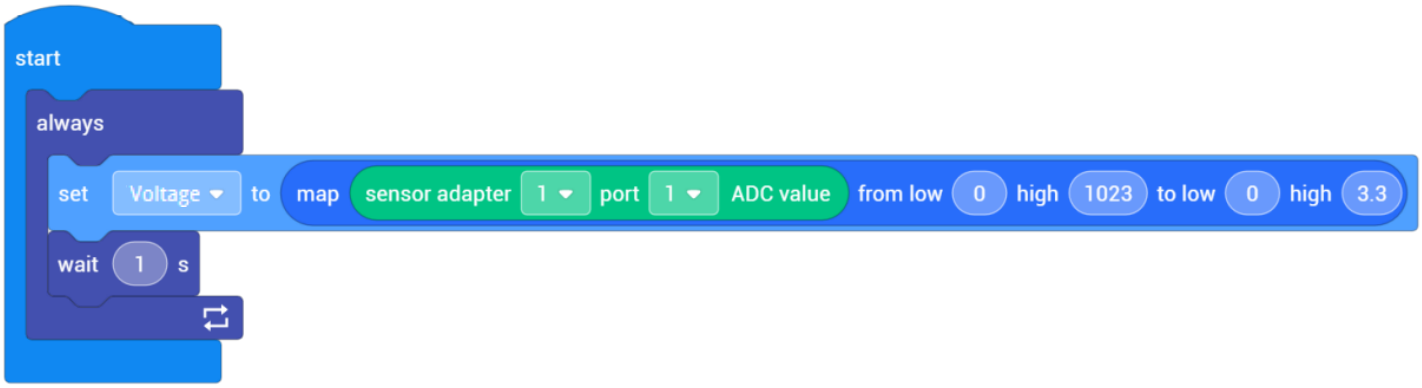


Note:
"and," "or," and "not" are logical operations, and the returned result will be a Boolean value: "True," or "False."
Boolean values:

Not A		
A	Condition False	Condition True
Not A	TRUE	FALSE

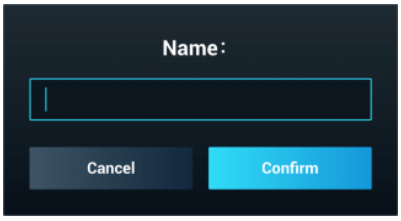
18. map (0) from the low (0) high (1023) to the low (0) high (4)

(1) Description: Scale the set value according to the proportional relationship.
(2) Type: Information
(3) Example: Ratio conversion
The data collected by the sensor is actually a 10-bit voltage value with 1024 discrete voltage levels with the value range of 0 to 1023. The value we see in the FPV window is the value after mapping, so only by conversion, scaling the value within the set range to another range according to the proportional relationship, can the true voltage value be known.

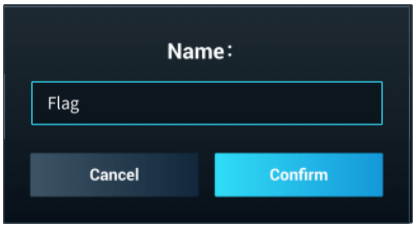

start
always
set
Voltage
to
map
sensor adapter
1
port
1
ADC value
from low
0
high
1023
to low
0
high
3.3
wait
1
s

Data Objects

1. create variable



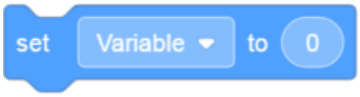
(1) Objective: Creates a name for a new variable
(2) Type: Settings block
(3) Example: Name a new variable
Each variable name must be unique and simple to understand.
For example, we often name the variable for a marker in the sequence as Flag, and the variable used to store value as Number.



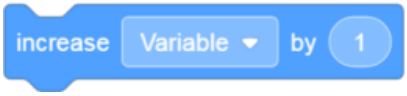
Variable names need to begin with an underscore or a letter, and can only contain numbers, uppercase and lowercase letters, and underscores.

Note:
After a variable name is created, you can assign and adjust the value using the three blocks available.

Use

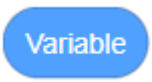


Assign Value

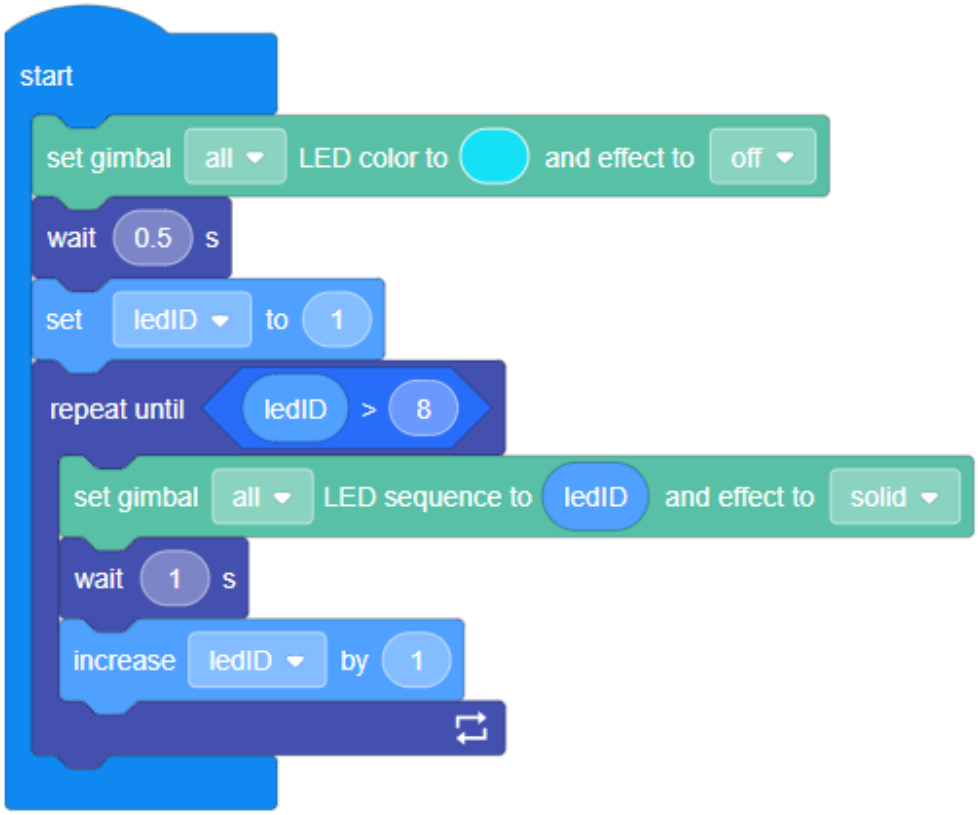


Increase/decrease

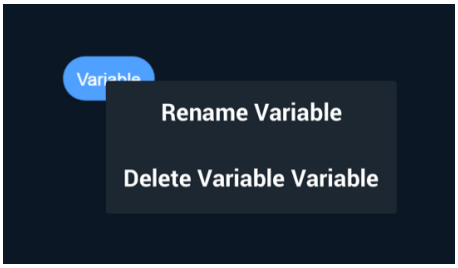
2. #Variable#



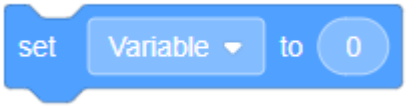
(1) Objective: Contains information about the variable
(2) Type: Information block (variable-type)
(3) Example: Switch on LEDs one after another
This will command the robot to switch off all gimbal LEDs, and then switch on LEDs 1 to 8 in sequence.



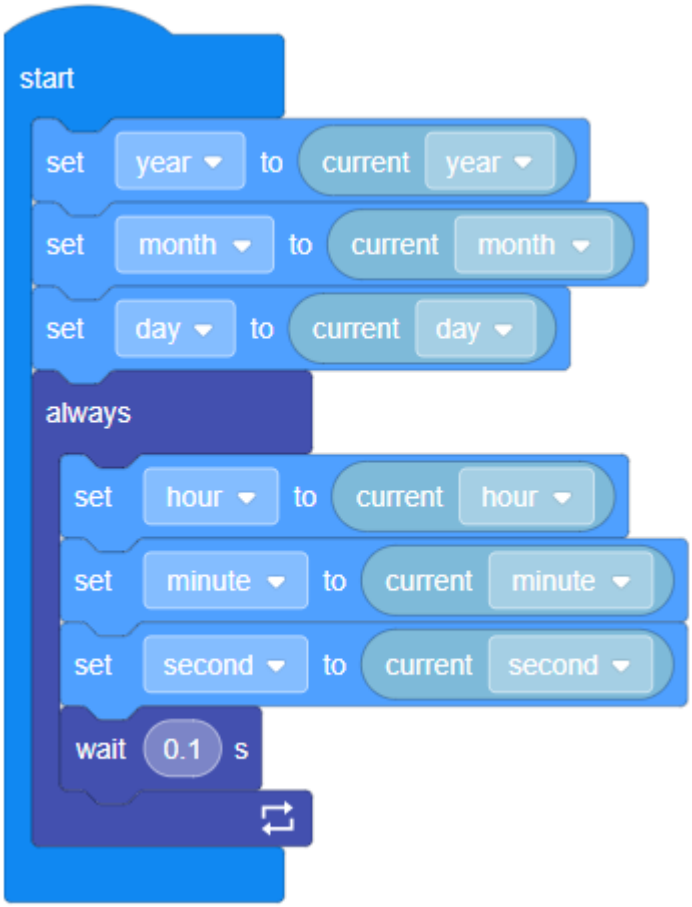
Note:
Right-click on a variable to rename or delete it.



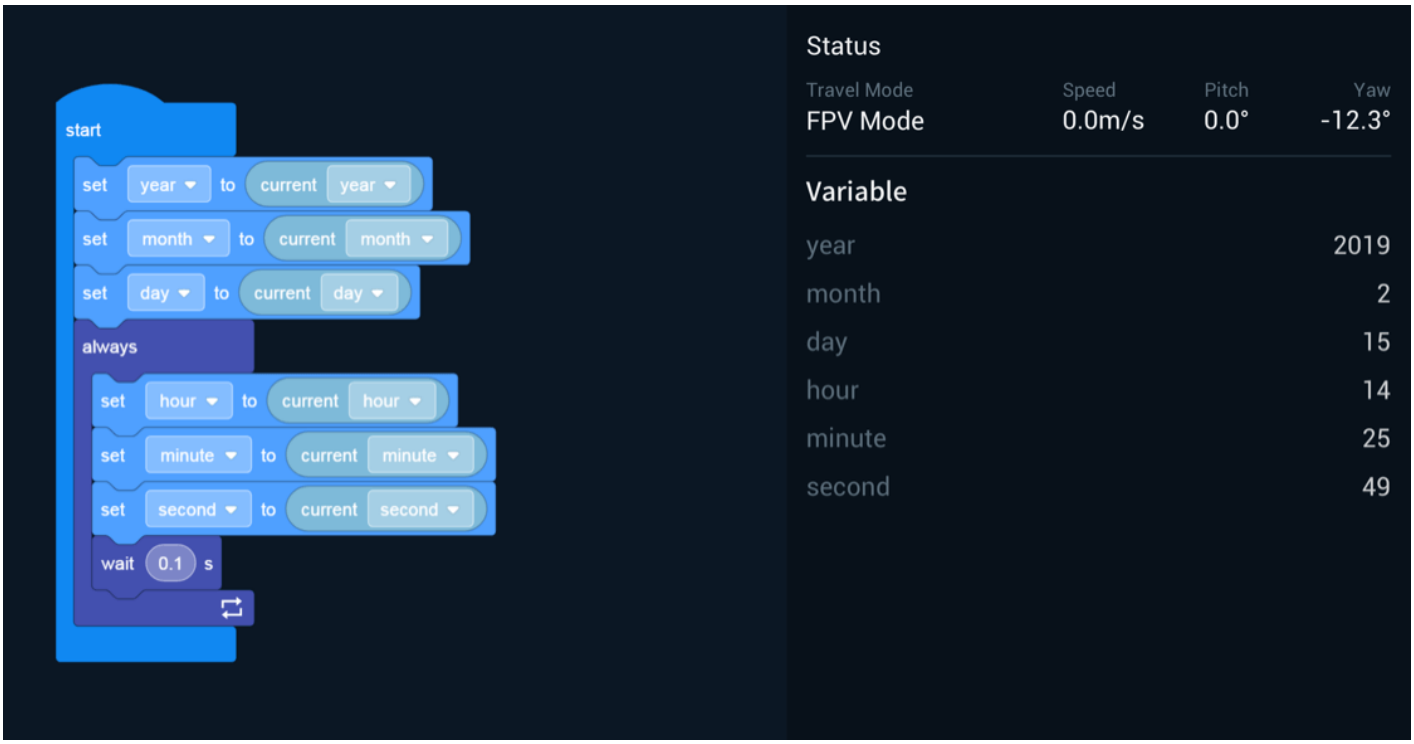
3. set (# Variable #) to (0)



(1) Objective: Assigns value to a variable
(2) Type: Execution block
(3) Example: Configure a digital clock



You can check details about the time using the FPV window. The minute and second values will change continuously.

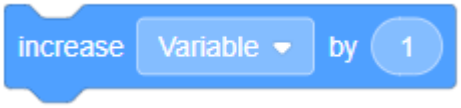


Note:

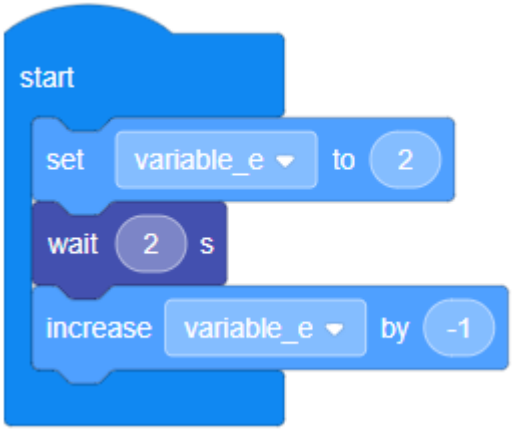
1) It is important to know the difference between a variable and a list; a variable stores one type of data, while a list stores a string containing a sequence of characters.

2) For a variable, the input value can be a number, a variable, or variable-type data, but cannot be a list or a list data type.

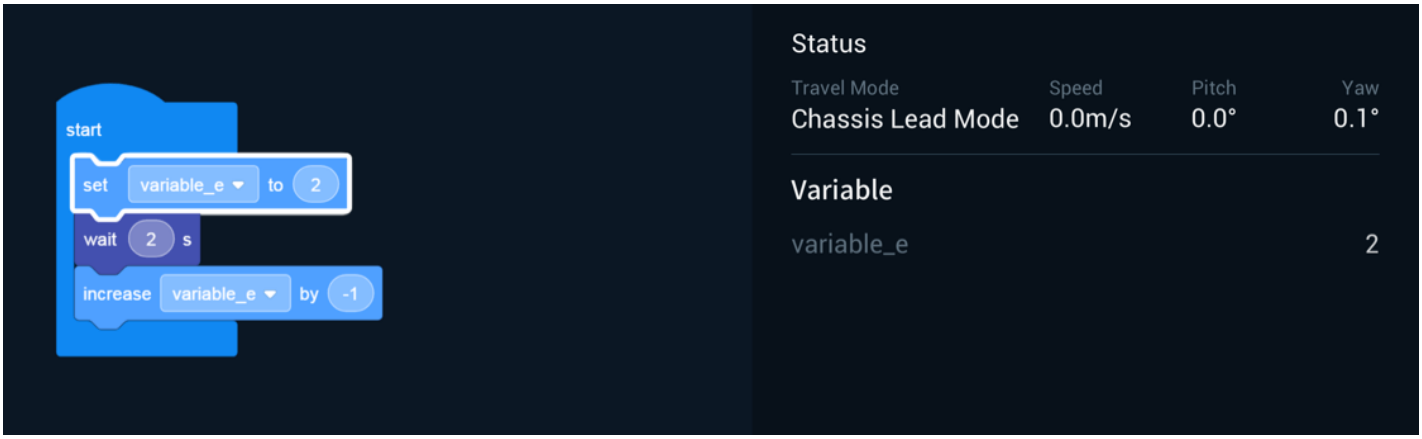
4. increase (# Variable #) by (1)



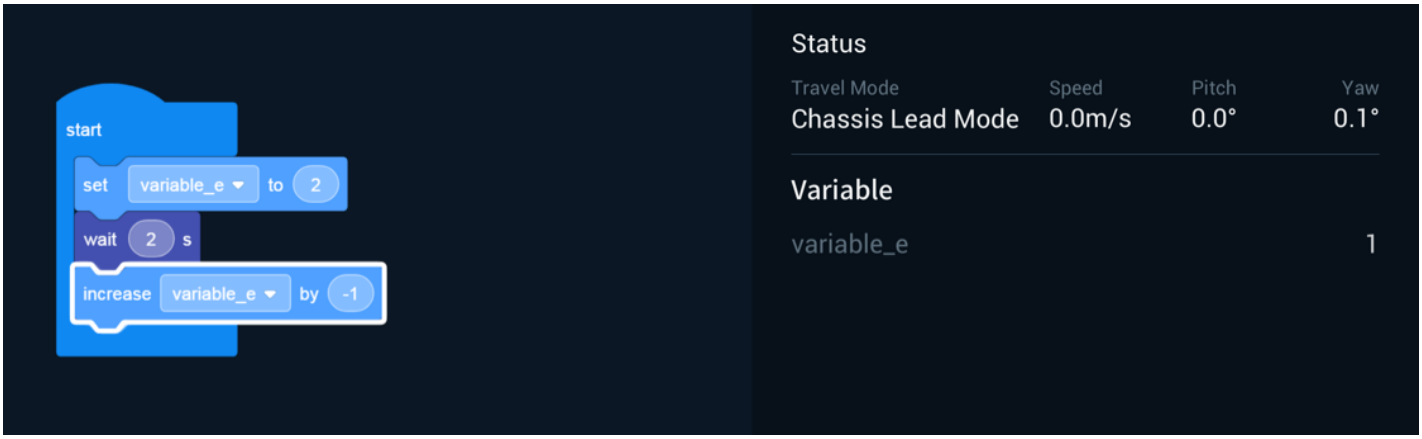
(1) Objective: Changes current variable value; positive values indicate an increase while negative values indicate a decrease.
(2) Type: Execution block
(3) Examples: Decrease variable value by 1, Translate in a figure-8 pattern
① Decrease variable value by 1
Assign a value to the variable, then configure the change in value.



In the FPV window, you can see that the initial variable value is 2.

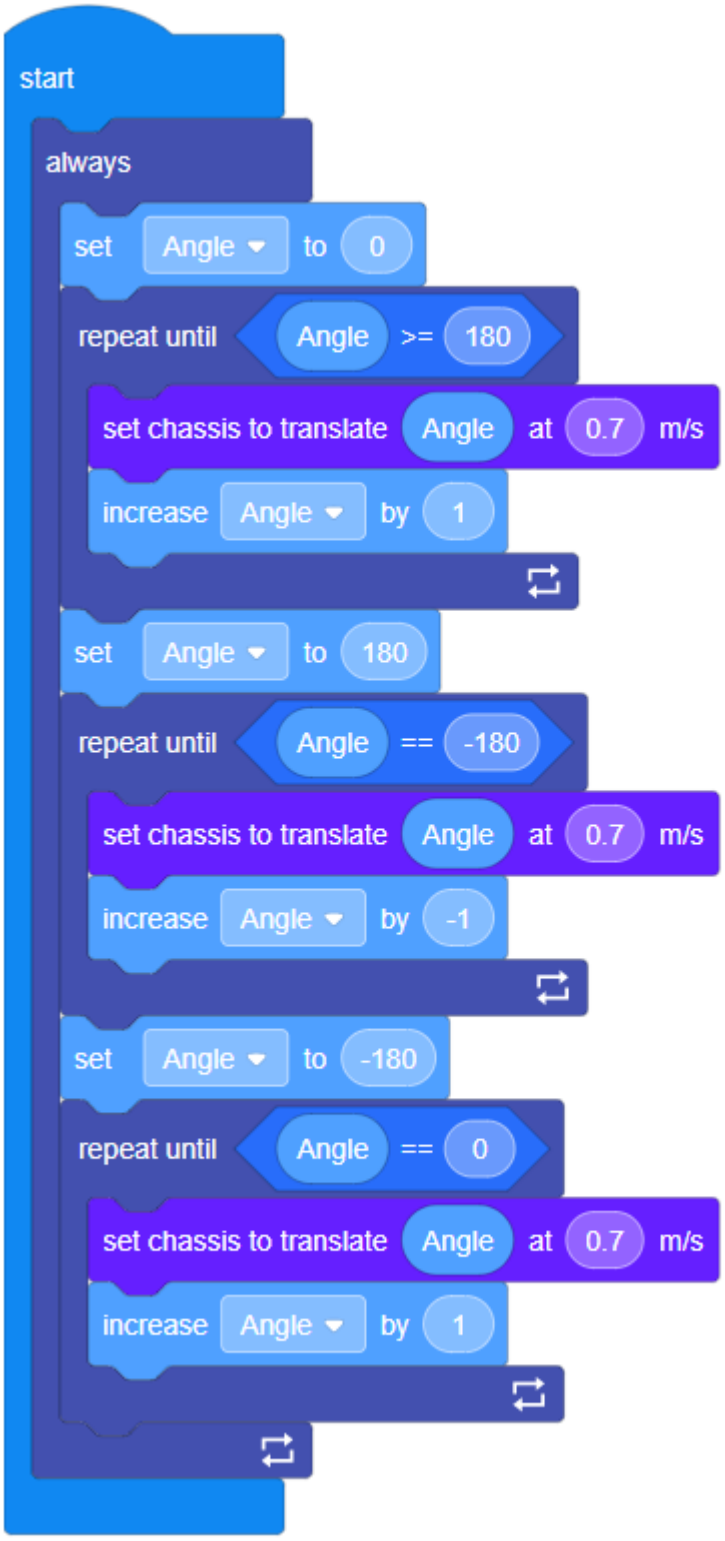


When you set the waiting time to 2 seconds and decrease the value by 1, the variable value becomes 1.



② Translate in a figure-8 pattern

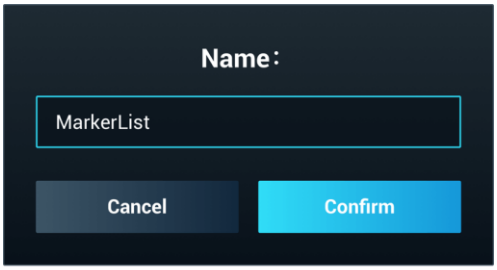
The screenshot below shows an alternative configuration for setting the robot to translate in figure-8 pattern.



5. create list

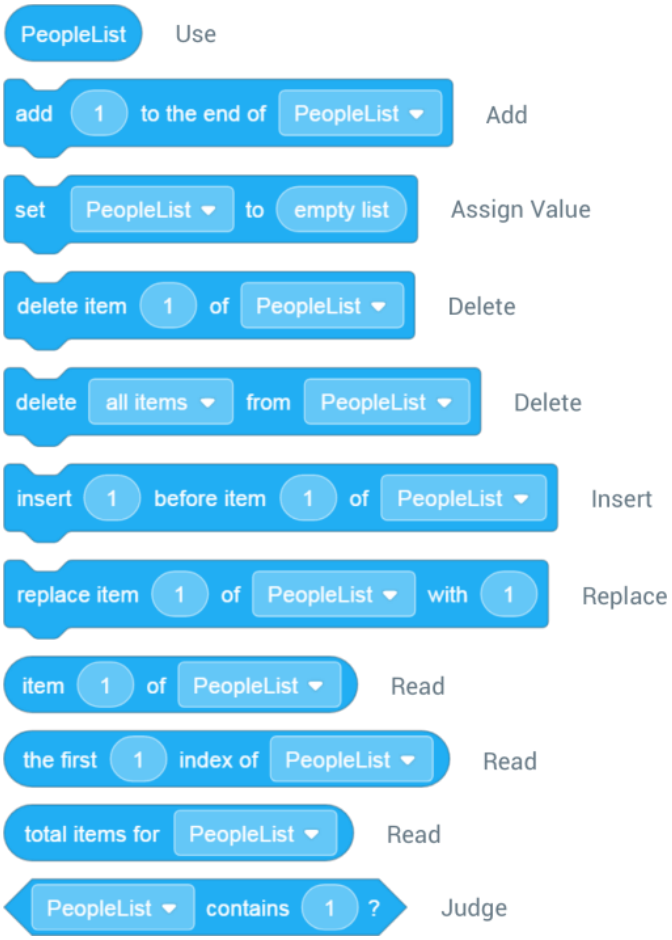
Create List

(1) Objective: Creates and names a list
(2) Type: Settings block
(3) Example: Name a new list



List names need to begin with an underscore or a letter and can only contain numbers, uppercase and lowercase letters, and underscores.

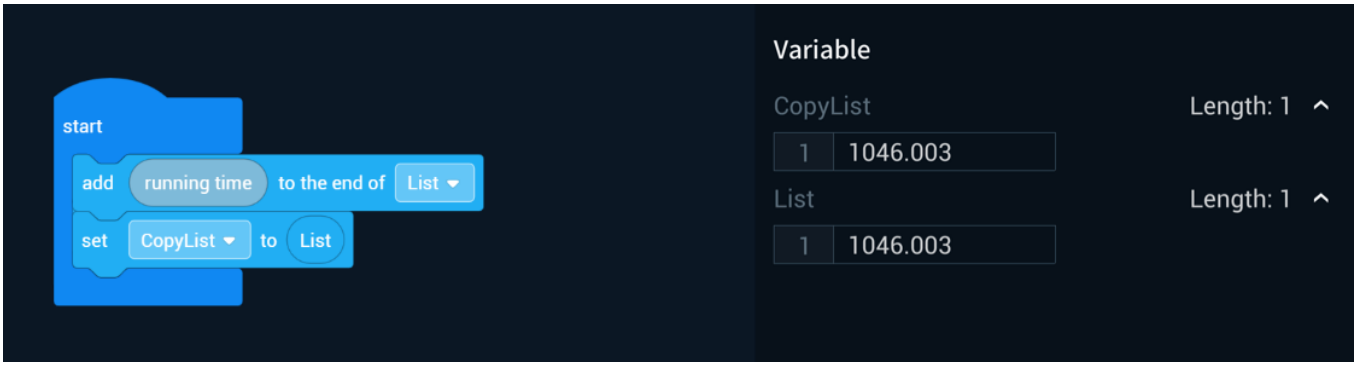
Note: After a list is created, assign and adjust the corresponding values using the blocks available.



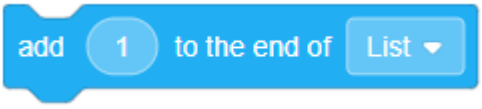
6. #List#

(1) Objective: Obtains all items in a list
(2) Type: Information block (list)
(3) Example: Duplicate a list
Ensure that the data in the new list "CopyList" and the current "List" are the same, and that the current running time is shown.

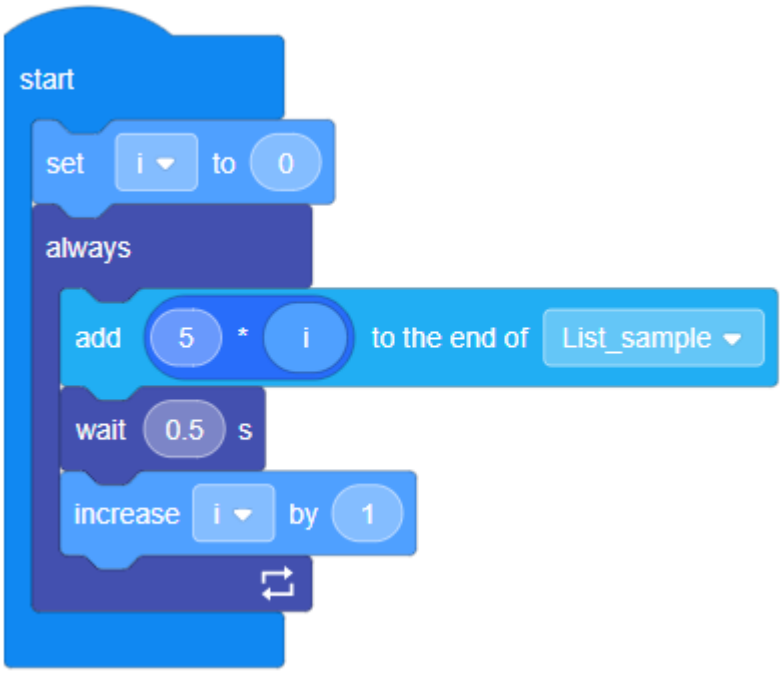
You can check specific details using the FPV window.



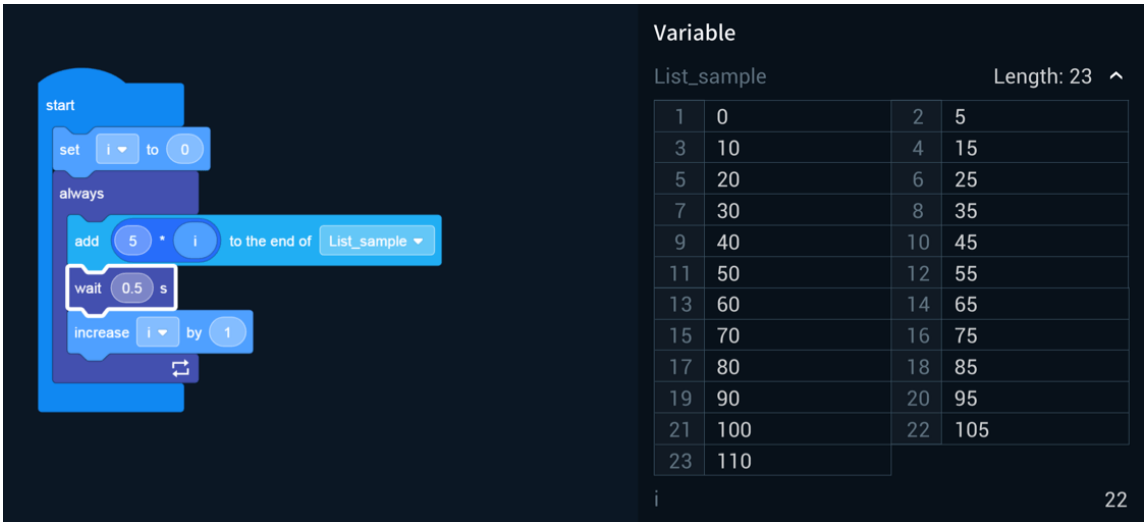
7. add (1) to the end of (#List#)



(1) Objective: Adds an item to the end of a list
(2) Type: Execution block
(3) Example: Display multiples of 5
The configuration below allows you to obtain and display all multiples (≥0) of 5.

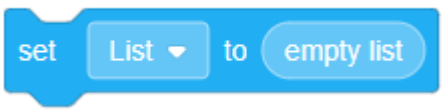


In the FPV window, a multiple of 5 is added to the list every 0.5 seconds.

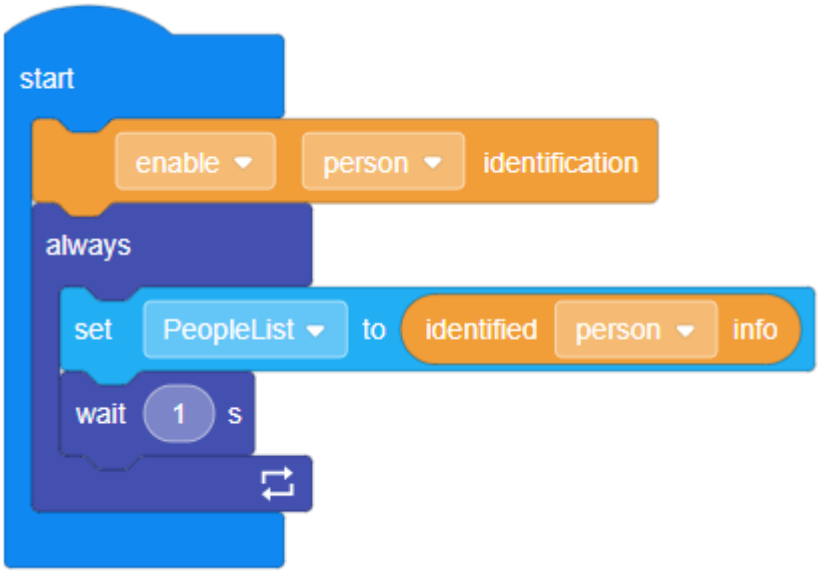


Note:
The input value can be a number, a variable, or variable-type data, but cannot be a list or a list-type data.

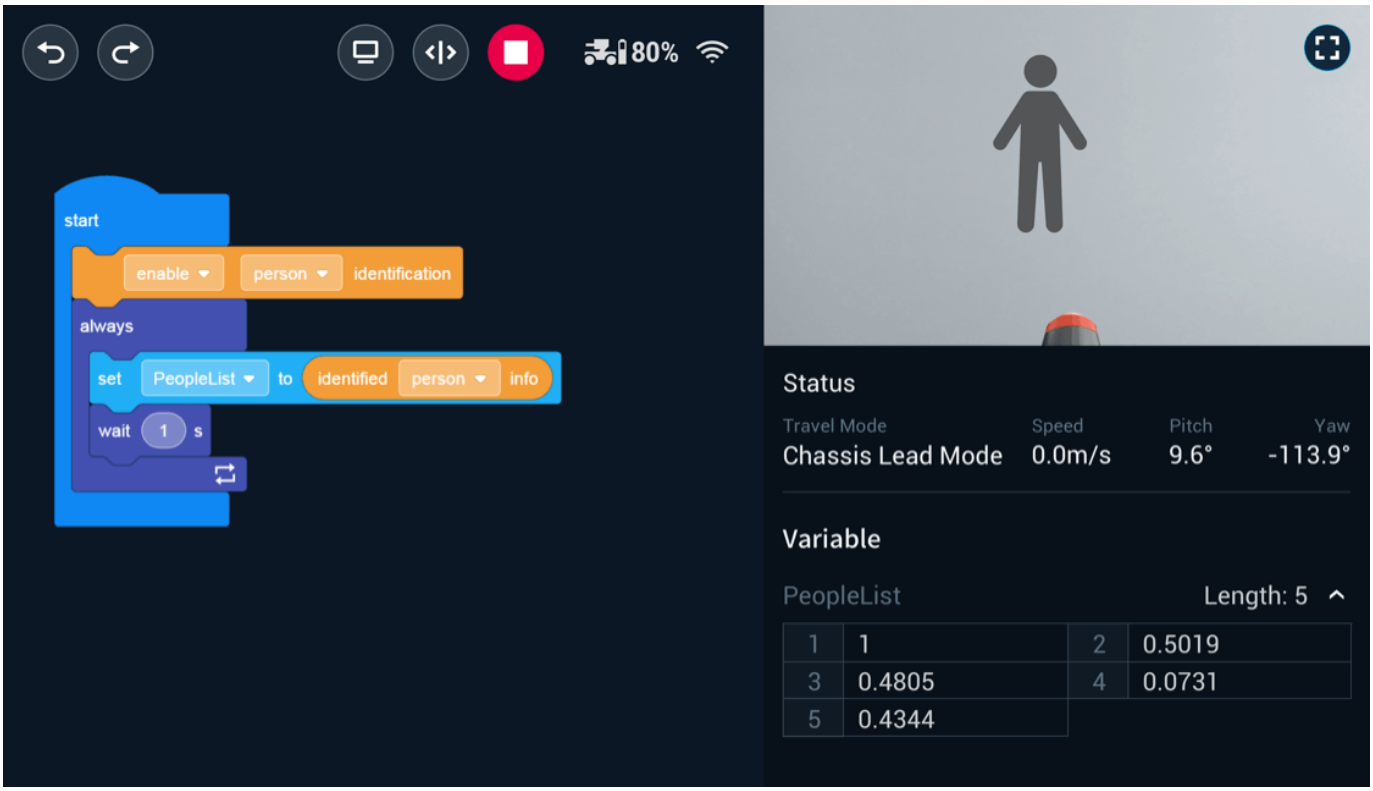
8. set (#List#) to (empty list)



(1) Objective: Assigns values to a list
(2) Type: Execution block
(3) Example: Recognize a person



In the FPV window, the list is configured to recognize a person and generate relevant information.



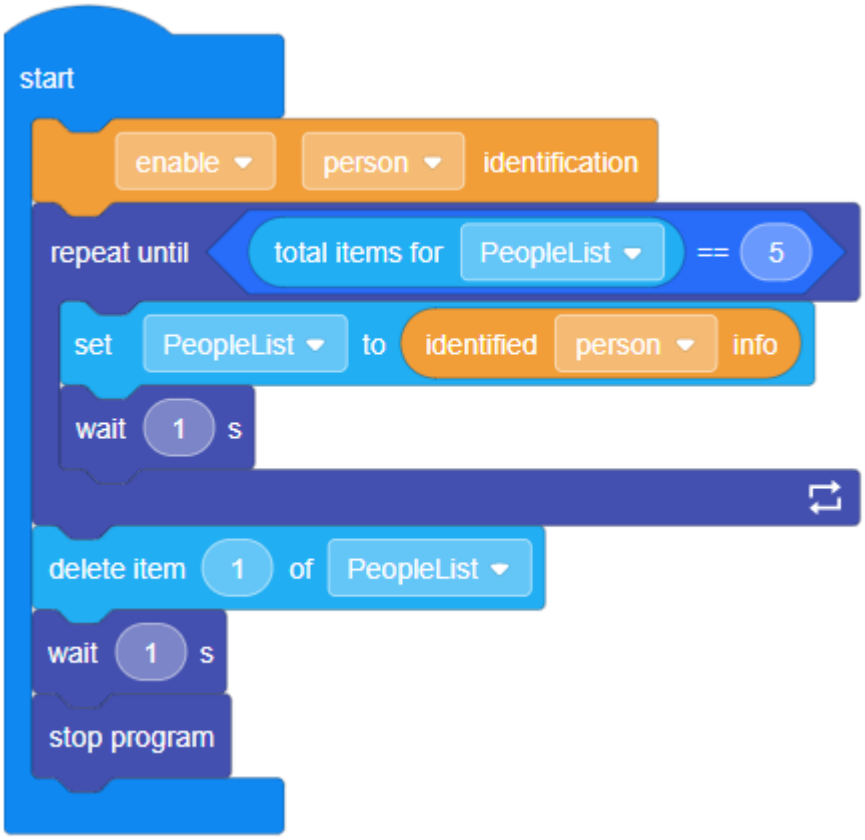
Note:

1) List values can be numbers, lists, or list-type data, but cannot be variables or variable-type data.

2) You cannot perform arithmetic operations directly on a list.

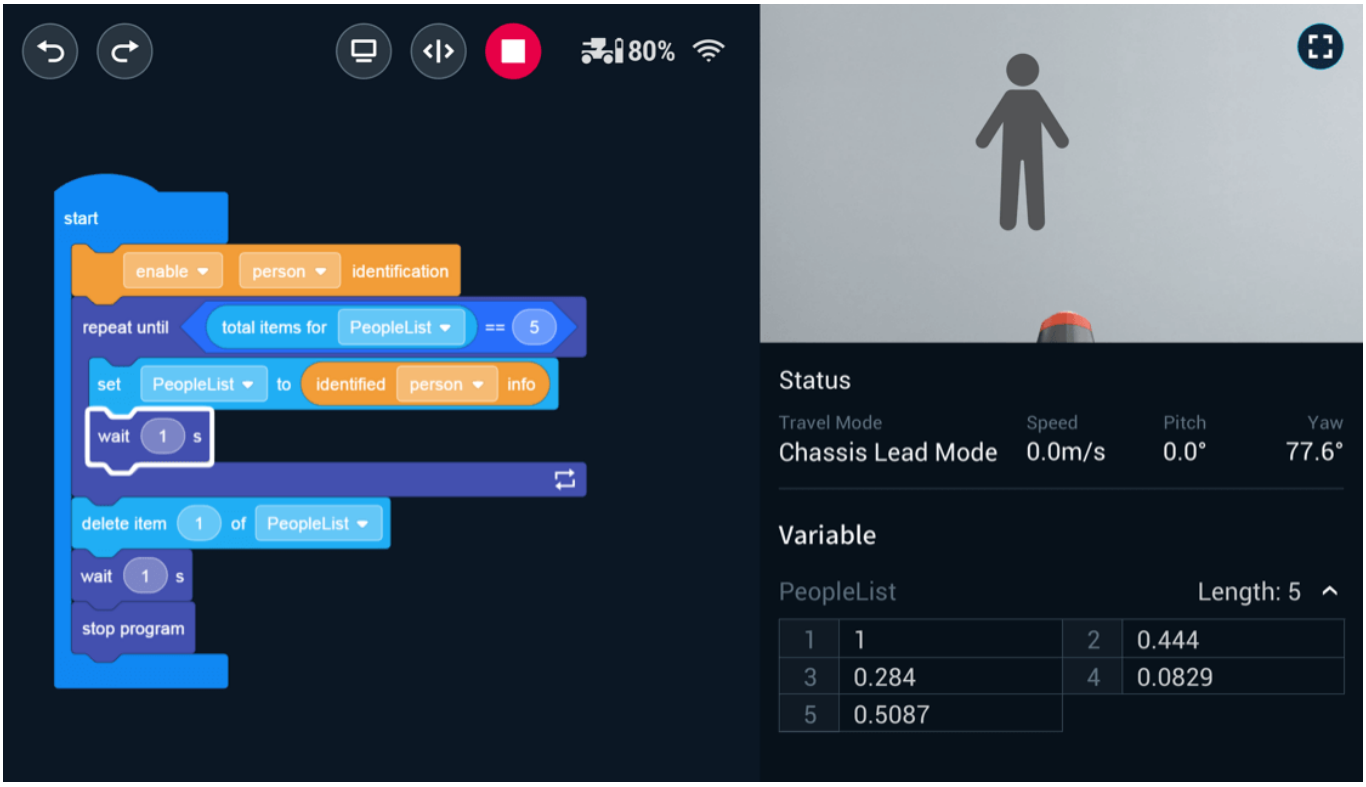
9. delete item (1) of (#List#)

(1) Objective: Deletes an item from a list
(2) Type: Execution block
(3) Example: Delete the number of people

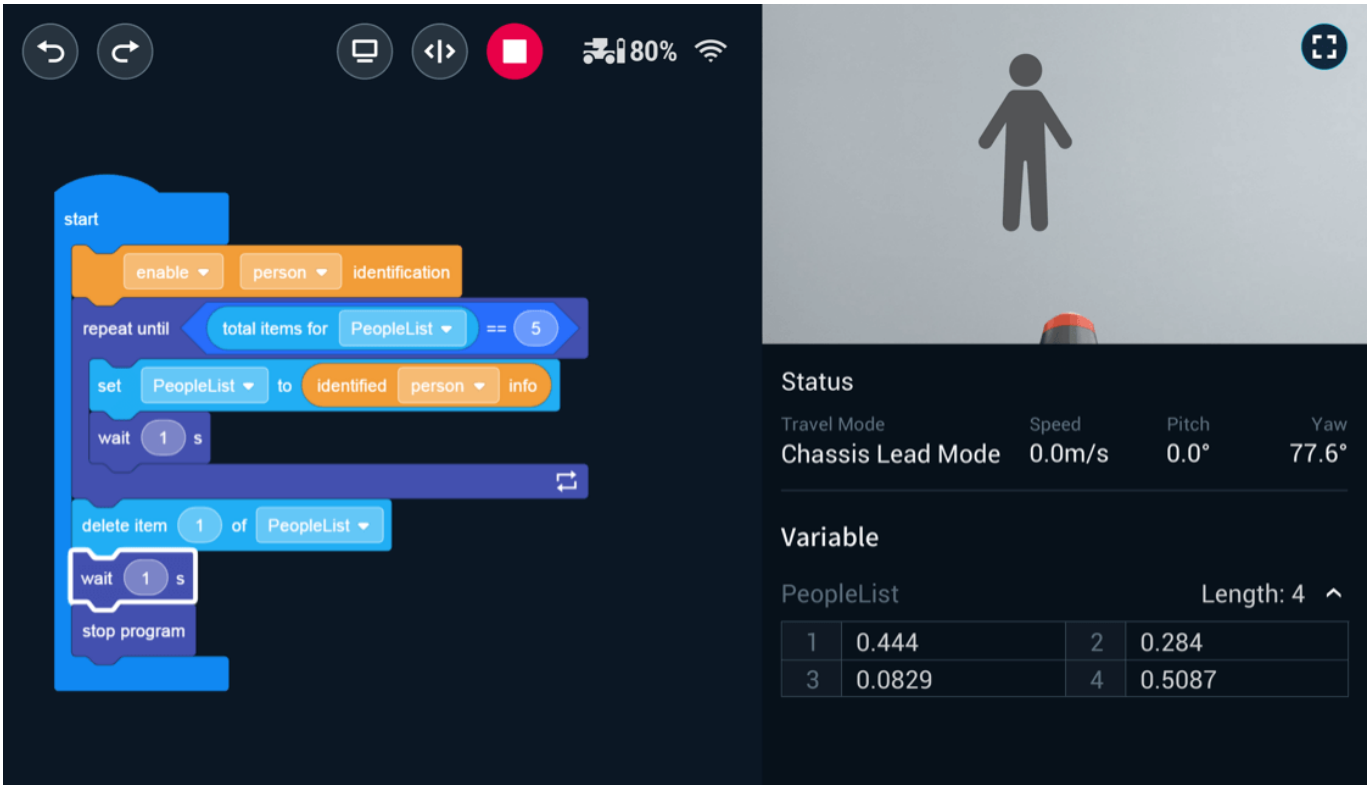


In the FPV window, the list length changes from 5 to 4 after deleting the first item.

Before:

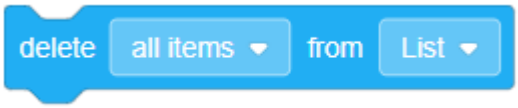


After:

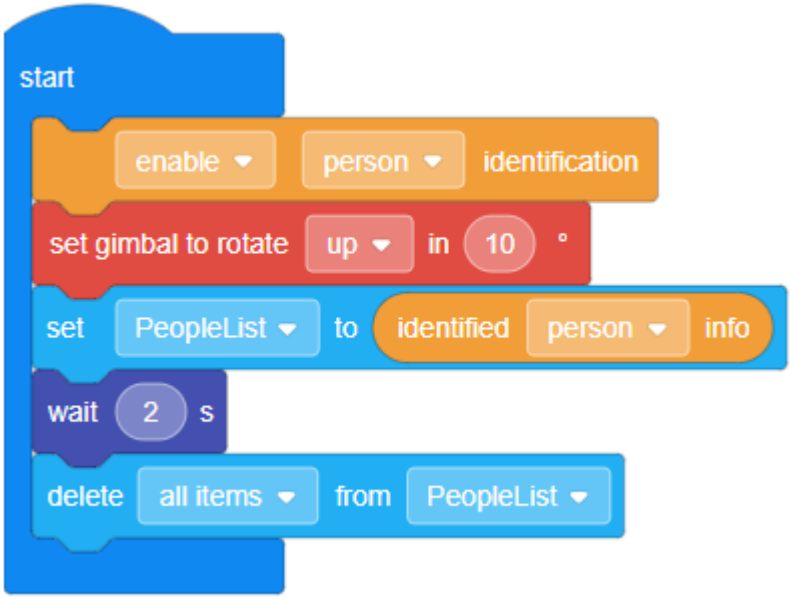


Note:
1) The user needs to know the serial number of the item before deleting it.
2) After deleting an item, the number of items decreases and their serial numbers change accordingly.

10. delete (all items) from (#List#)

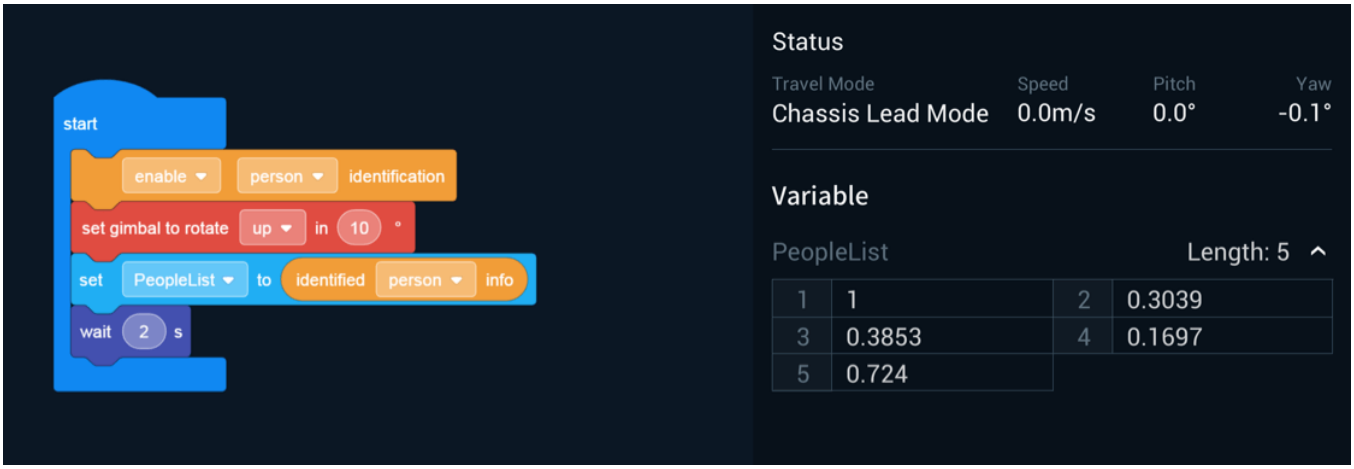


(1) Objective: Deletes all items or the last item from a list
(2) Type: Execution block
(3) Example: Clear list(All Items)

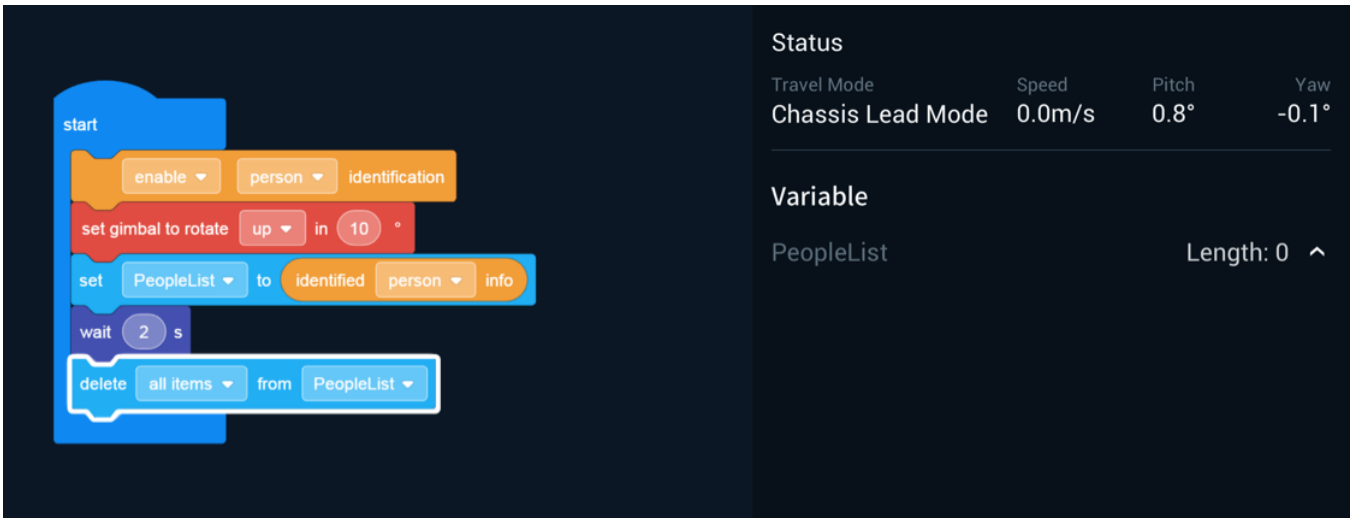


In the FPV window, the list length changes from 5 to 0.

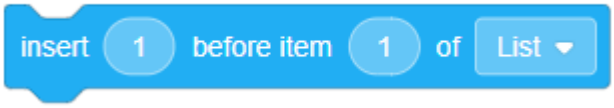
Before:



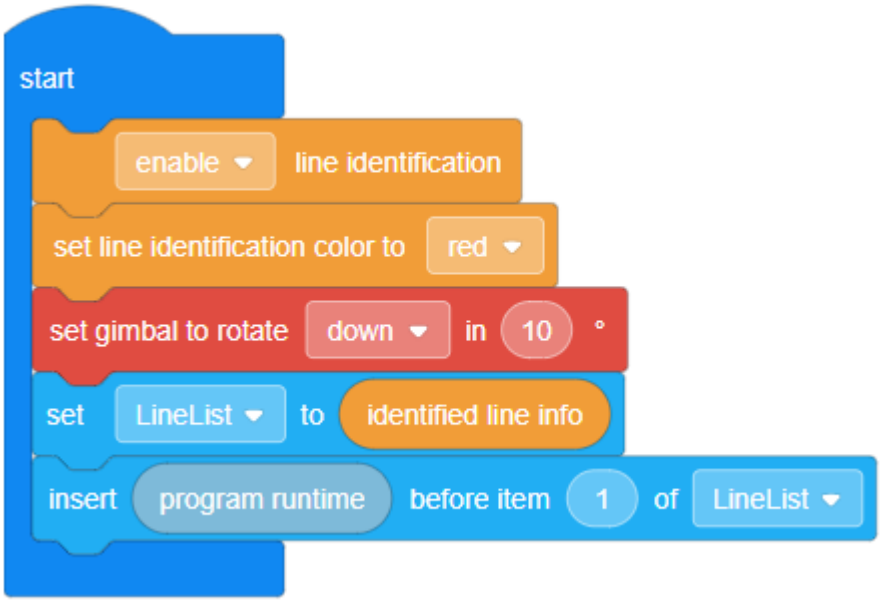
After:



11. insert (1) before item (1) of (#List#)

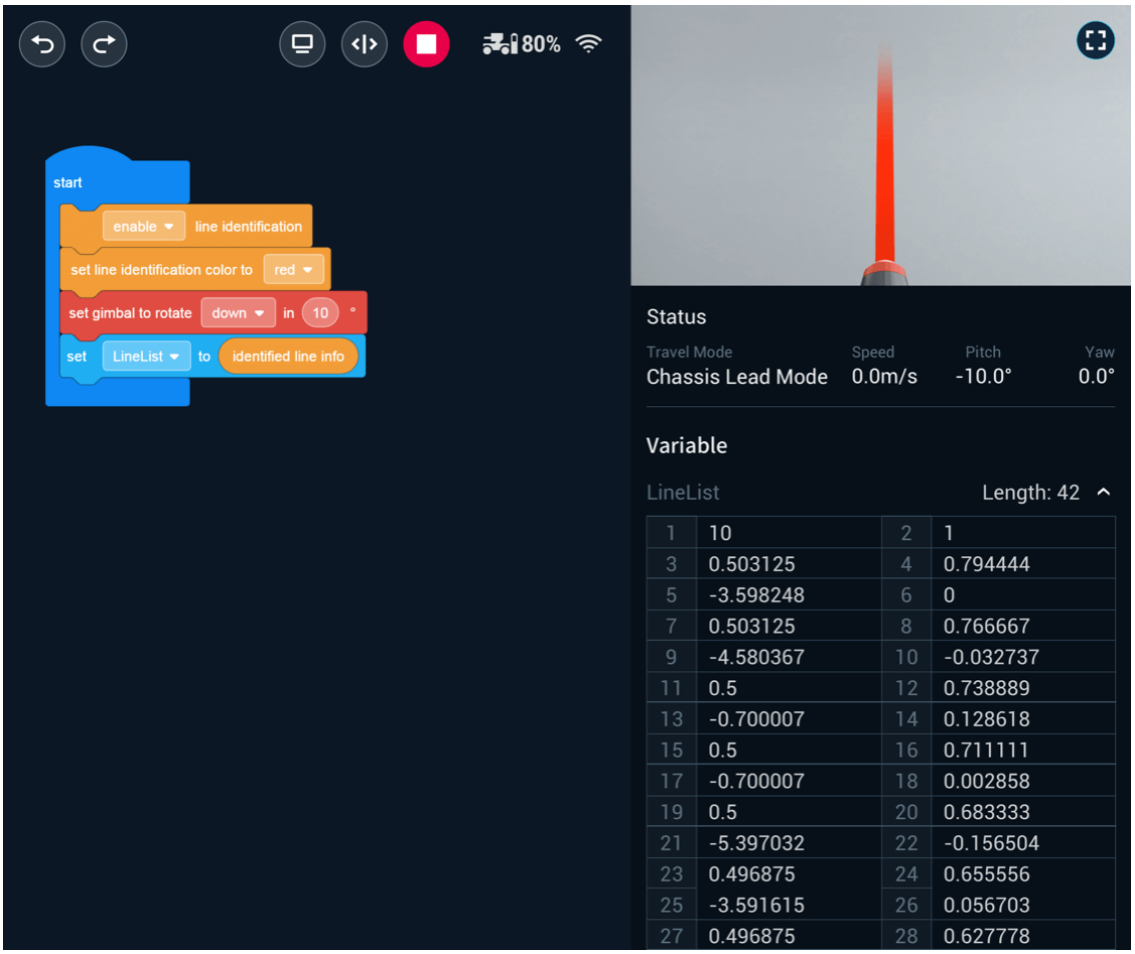


(1) Objective: Inserts an item at a specific location in a list and shifts subsequent items down the list
(2) Type: Execution block
(3) Example: Insert program runtime timer

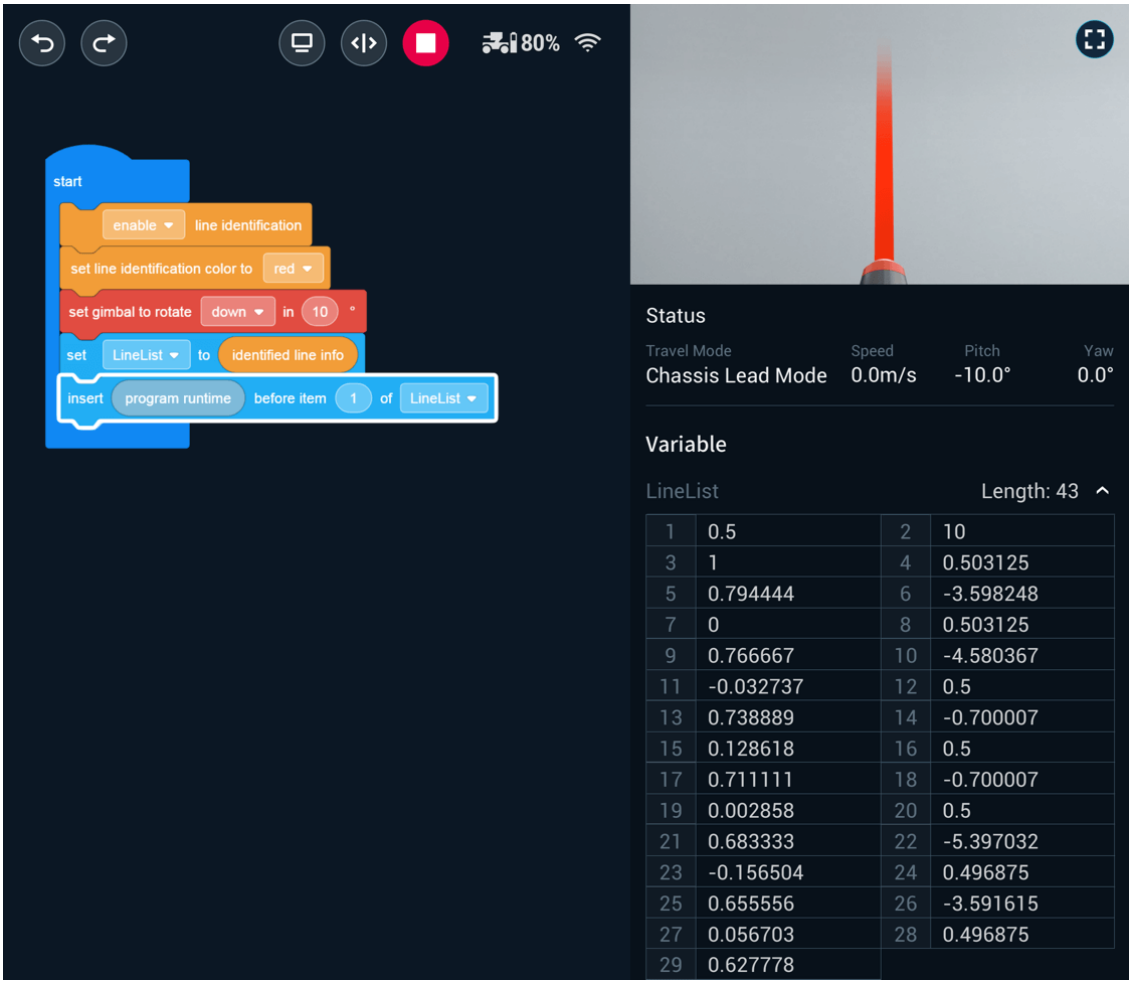


Apply the red tape and observe the FPV window. You can see that the program runtime is inserted as the first item of the list, the original items are shifted down the list, and the total number of items increases by 1.

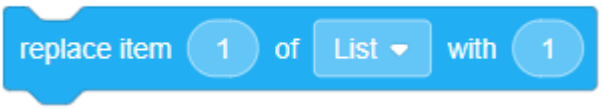
Before:



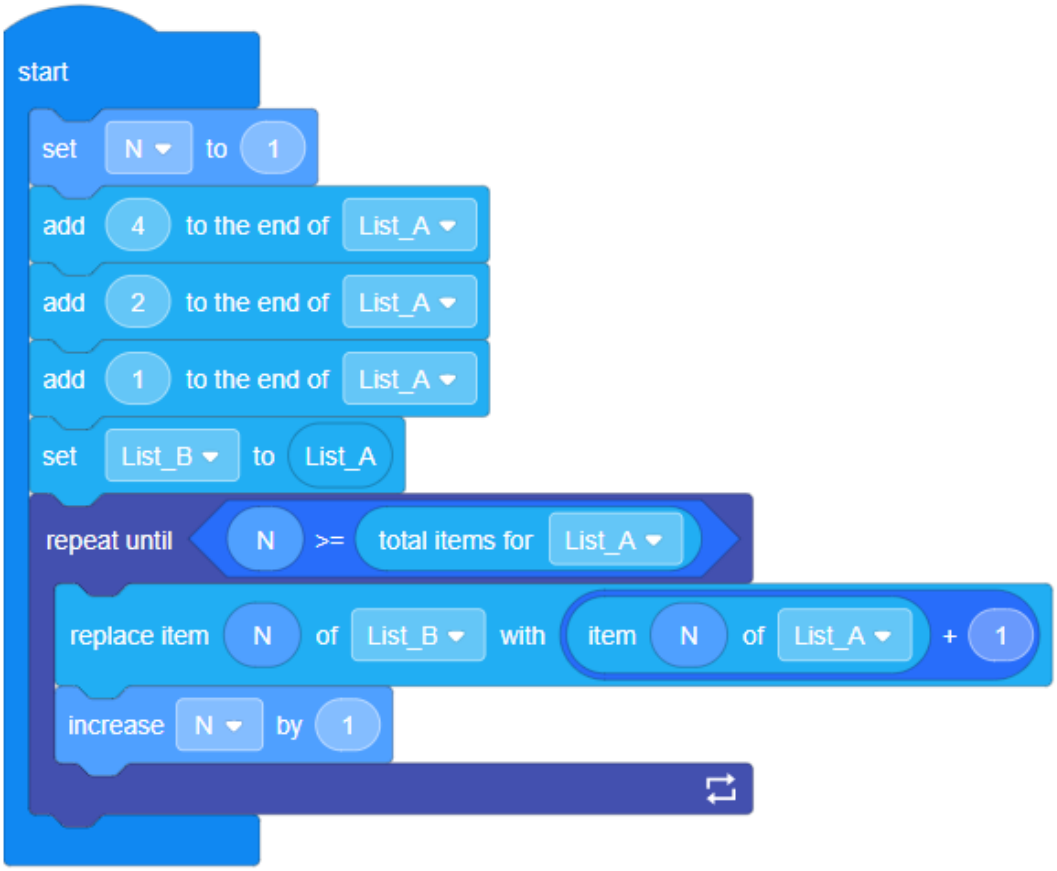
After:



12. replace item (1) of (#List#) with (1)



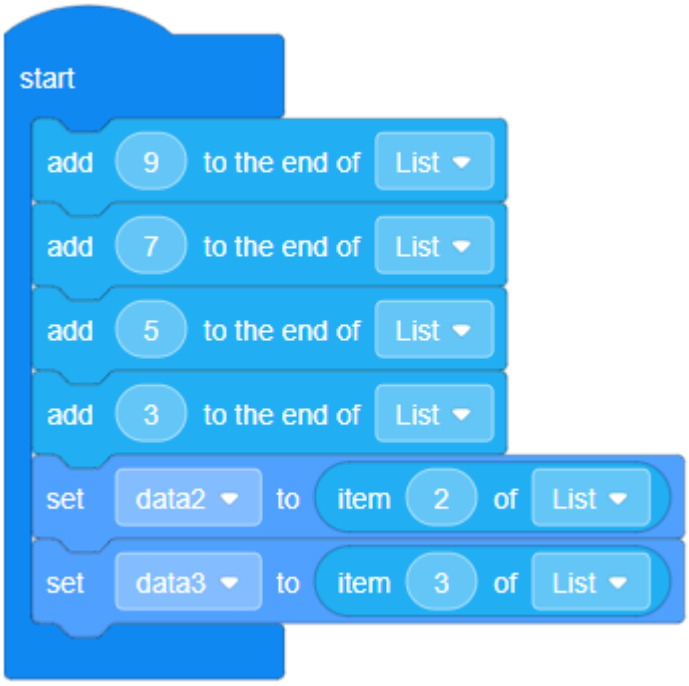
(1) Objective: Replaces an item on the list
(2) Type: Execution block
(3) Example: Replace a value
This will make every value in List_B be greater than the corresponding value in List_A by 1.



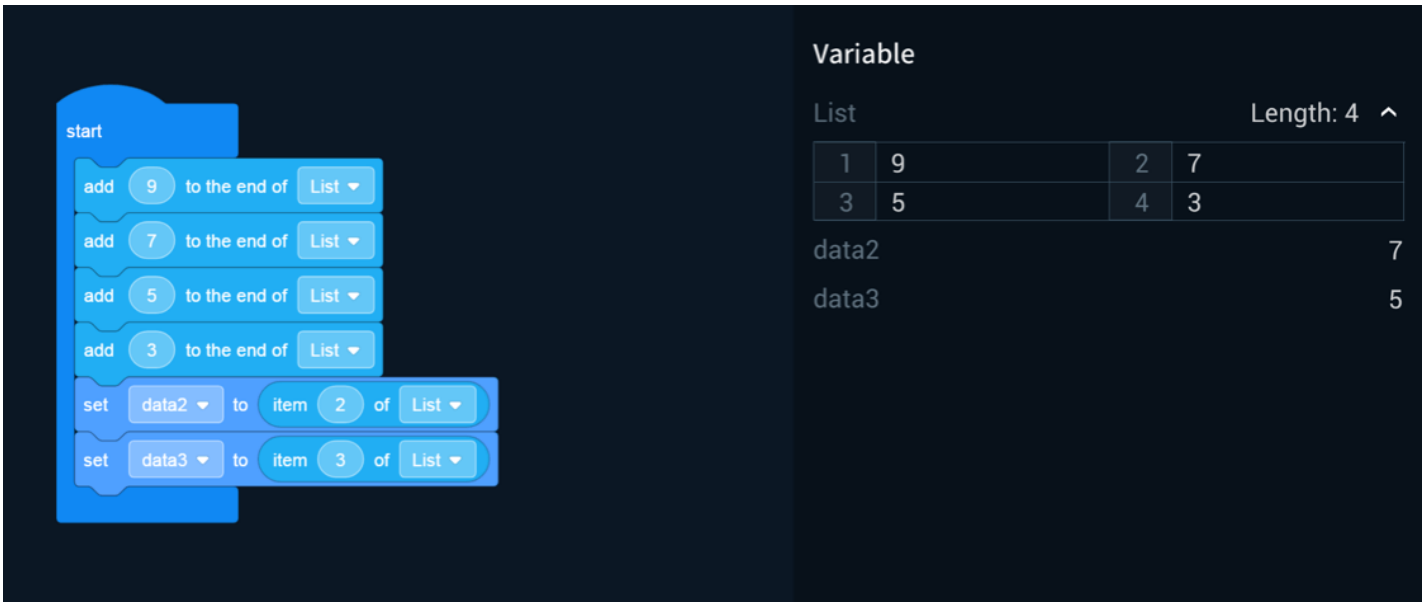
Note:
The item must contain a value before it can be replaced. For example, if List_B is empty, you will not have any items to replace it with.

13. item (1) of (#List#)

(1) Objective: Returns a specific item on a list
(2) Type: Information block (variable-type)
(3) Example: Specify an item on a list

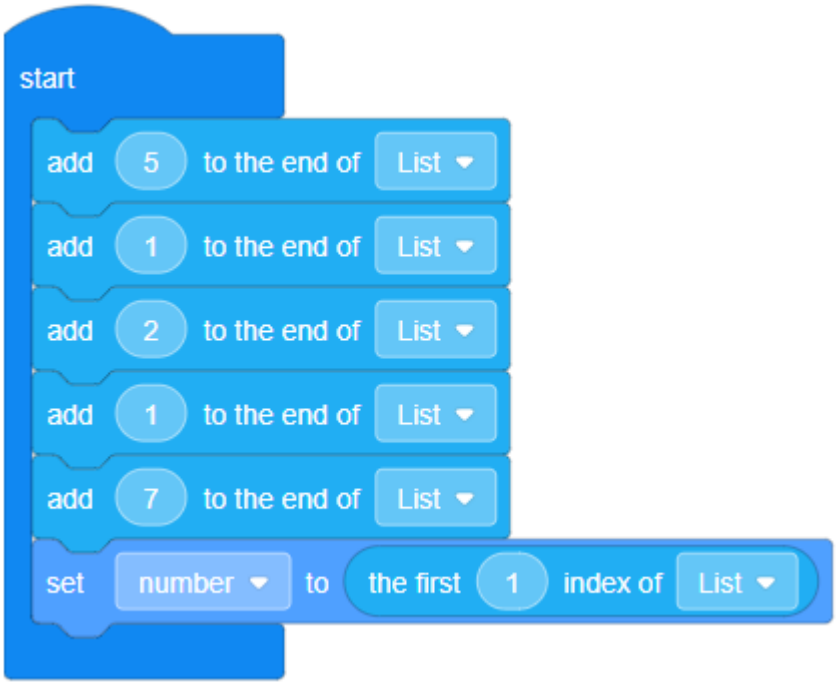


In the FPV window below, data2 = 7 and data3 = 5.

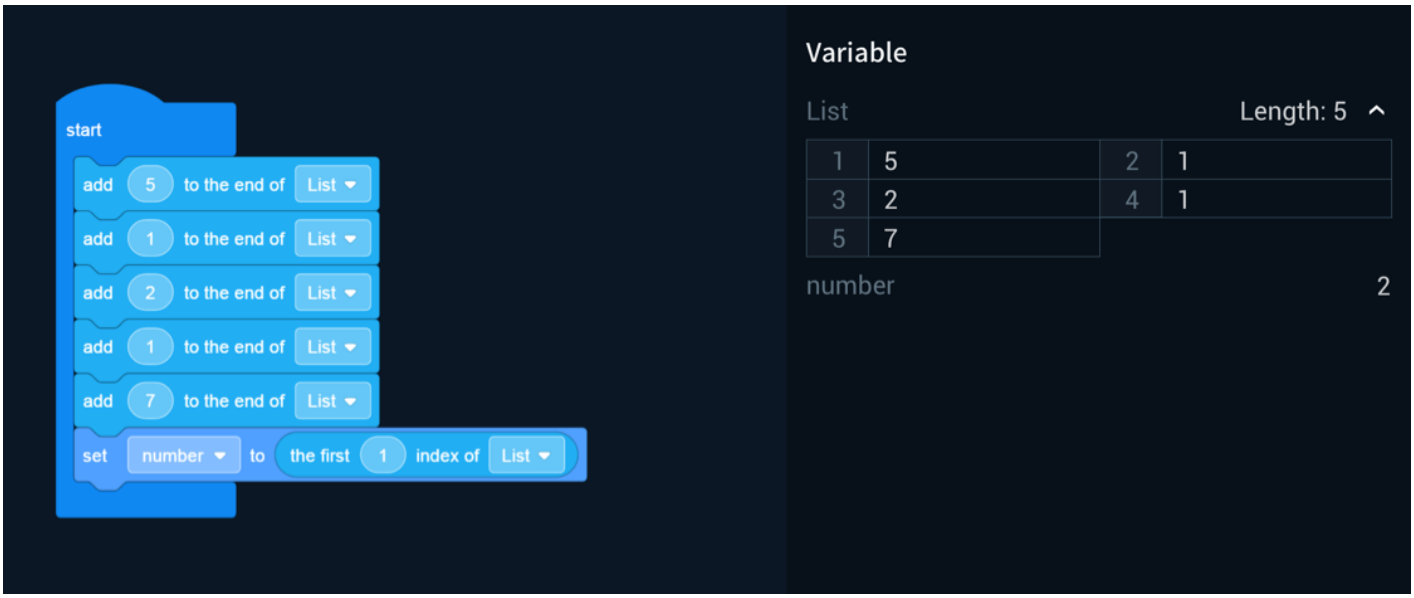


14. the first (1) index of (#List#)

(1) Objective: Obtains the location of the data's first occurrence in the current item
(2) Type: Information block (variable-type)
(3) Example: Read the index value
Create a list of {5, 1, 2, 1, 7}, and set the index number for the first "1" to "2."

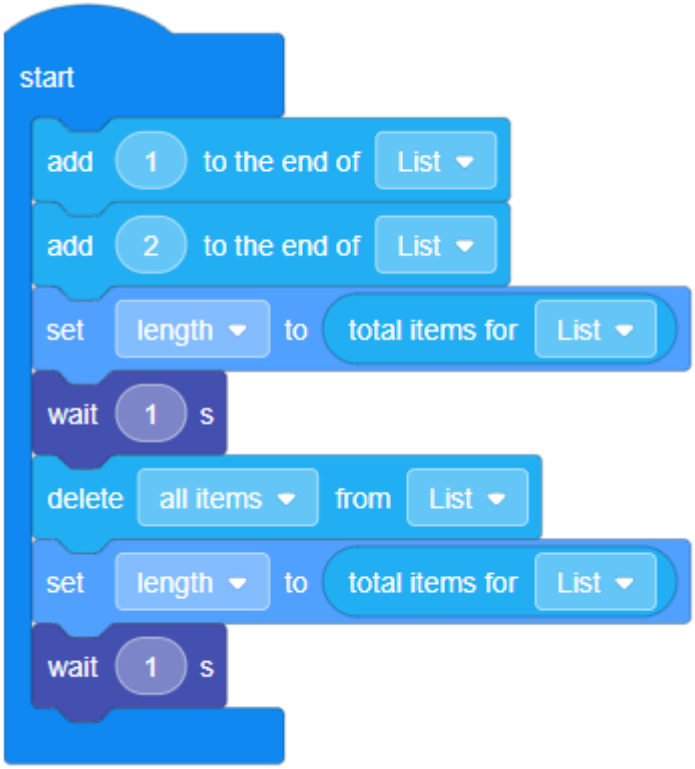


You can check the result using the FPV window:



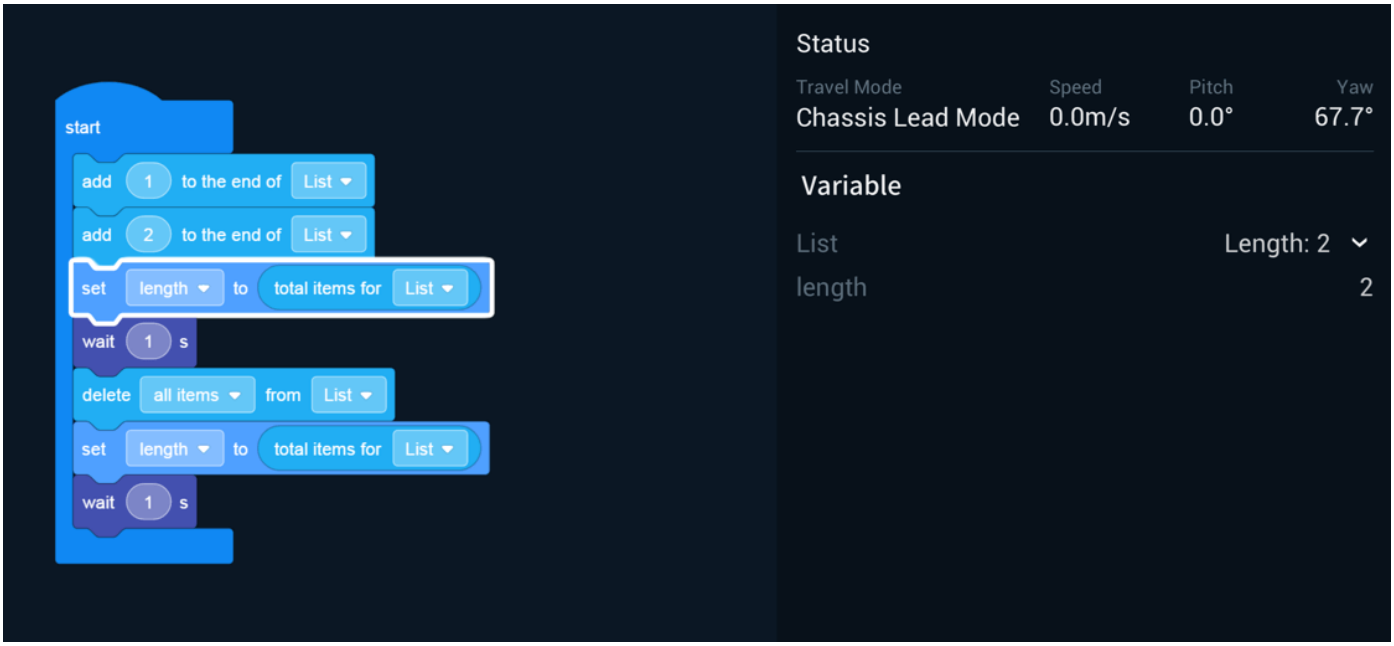
15. total items for (#List#)

(1) Objective: Obtains the total number of items for a list
(2) Type: Information block (variable-type)
(3) Example: Calculate the number of items

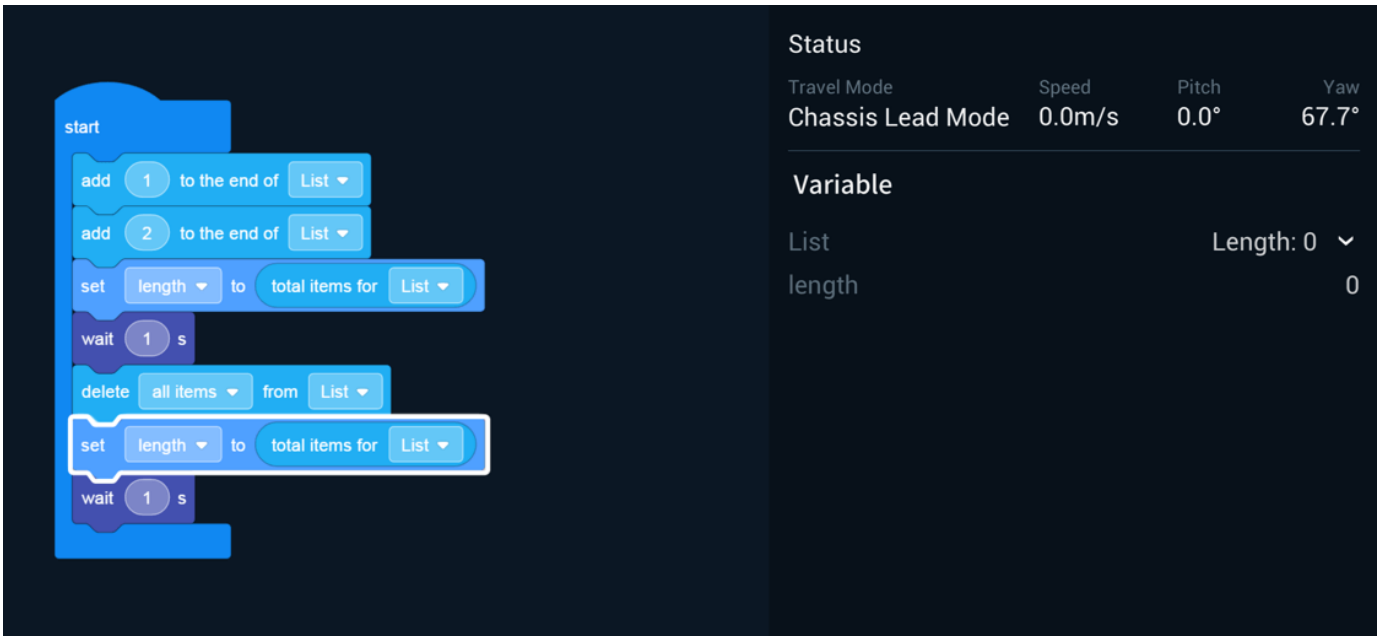


You can check value changes using the FPV window. The length value starts at 2 and becomes 0 after deleting all items.

Before:

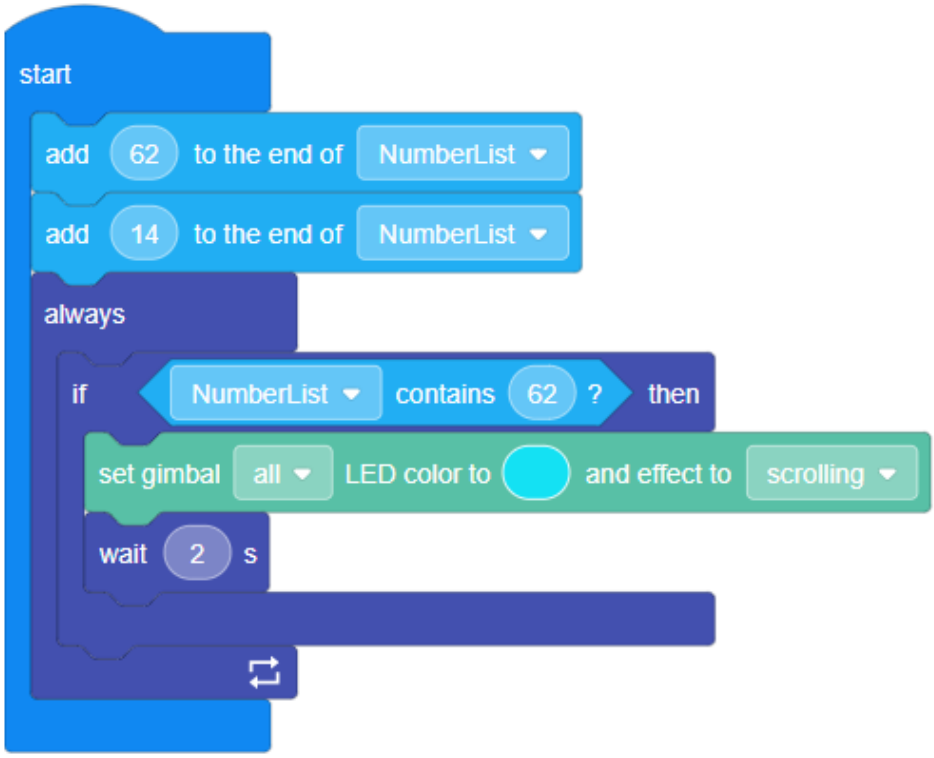


After:



16. (#List#) contains (1) ?

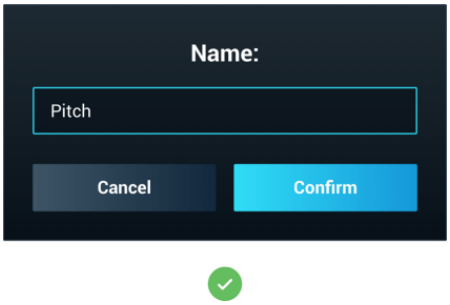
(1) Objective: Returns “True” when a list contains a specific value; otherwise, “False” is returned
(2) Returned value: Boolean
(3) Example: List containing a specified value
If there is a specific value set for a list, the gimbal LEDs will start to scroll.



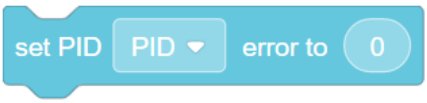
Note:
The condition “True” will only return when the determining item meets the conditional statement.

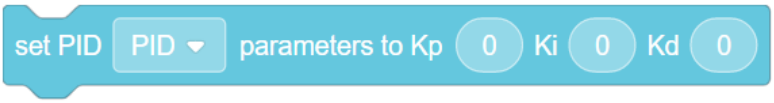
17. create PID controller

(1) Objective: Creates and names a PID controller
(2) Type: Settings block
(3) Example: Name controller



Note:
After the controller is created, there will be three modules available to configure error settings, adjust parameters, and obtain output information.

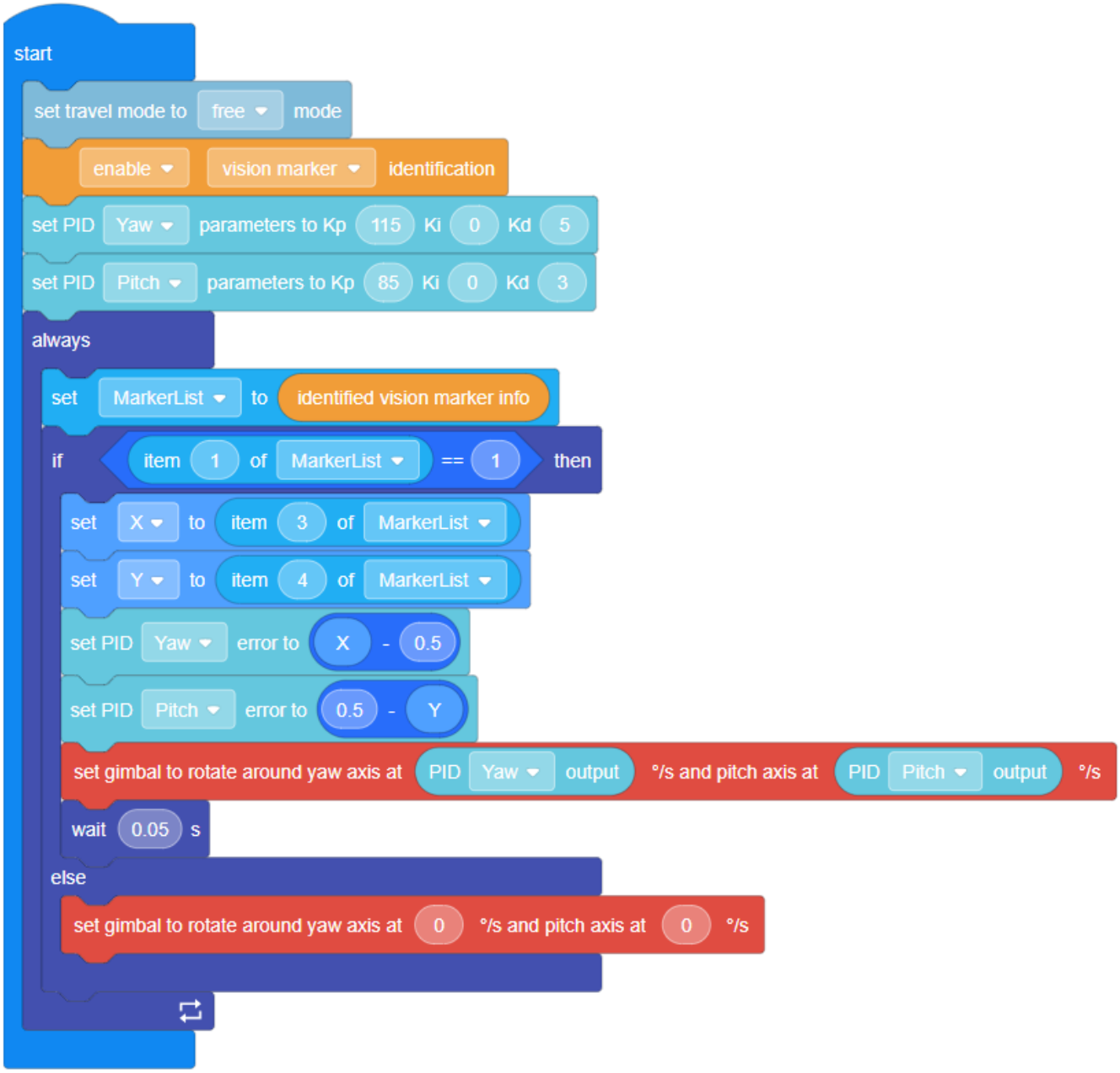




Python API:
Class: rm_ctrl.PIDCtrl()

18. set PID (#PID#) error to (0)

(1) Objective: Sets the PID controller error, which is the difference between the target and returned values
(2) Type: Settings block
(3) Example: Follow a Vision Marker
Manually hold and move a Vision Marker to control the gimbal's movement to follow.
This will set the PID controller error, which is the difference between the center of the Vision Marker and the center of the robot's field of view.



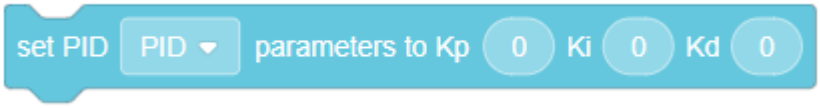
	Leaning	Upside Down	Skewed	Moving Too Fast	Dim	Obstructed
Case						
Impact	No Impact	No Impact	Impacted	Impacted	Impacted	Fail to identify

Note:
Before running the program, make sure that the Vision Marker is aligned to the robot's field of view.

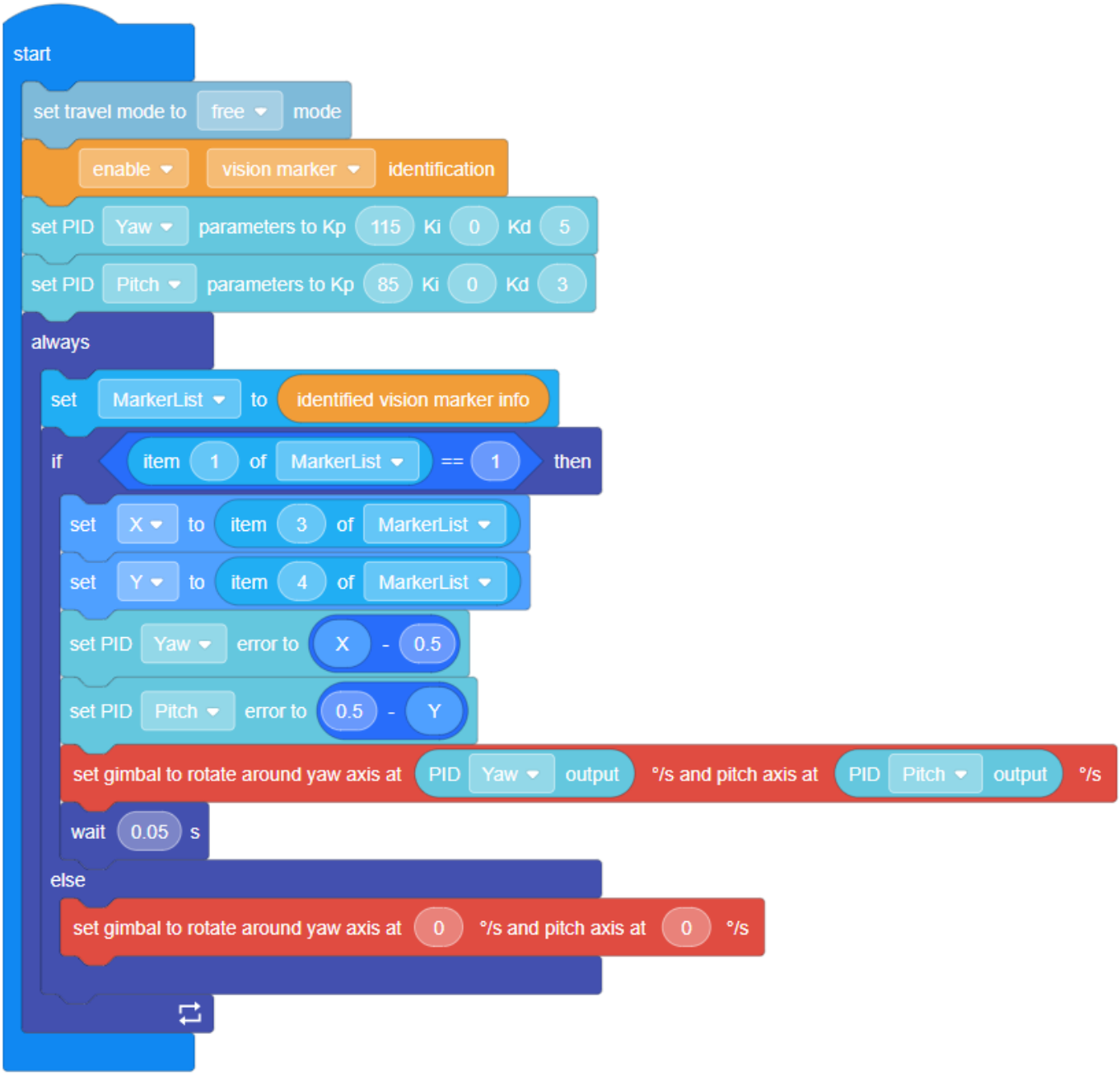
Python API:
Class: rm_ctrl.PIDCtrl()
Function:

- set_error(error)

19. set PID (#PID#) parameters to Kp (0) Ki (0) Kd (0)



(1) Objective: Adjusts the PID parameters; Kp is the proportional coefficient, Ki is the integral coefficient, and Kd is the differential coefficient.
(2) Type: Settings block
(3) Example: Follow Vision Marker
Modify parameters of Kp, Ki, and Kd to optimize the closed-loop control system.



Note:

	Function	Features	Weakness
Proportional Control(P)	Amplifty or weaken the error signal. Proportional coefficient determines the strength of control.	The larger the proportional coefficient, the more responsive the system becomes. But may oscillate and destabilize the system if the coefficient is too large.	Cannot eliminate the steady-state error of the system, lowering the relative stability of the system.
Integral Control(I)	Affect controller output with error accumulation, and reduce deviation with negative feedback of the system.	It is related to the existence period of the error signal. As long as there is enough time, the integral control can eliminate the steady-state error.	Fail to overcome the effects of interference in a timely manner.
Differential Control(D)	It can reflect change of speed of the error signal, and have immediate control when the error just occurs.	Helps reduce adjustment time and improve system quality.	Cannot eliminate the steady-state error of the system.

Find out more about PID in the RoboMaster app by searching for the "Seek & Destroy" project in the Road to Mastery section.

Python API:

Class: rm_ctrl.PIDCtrl()

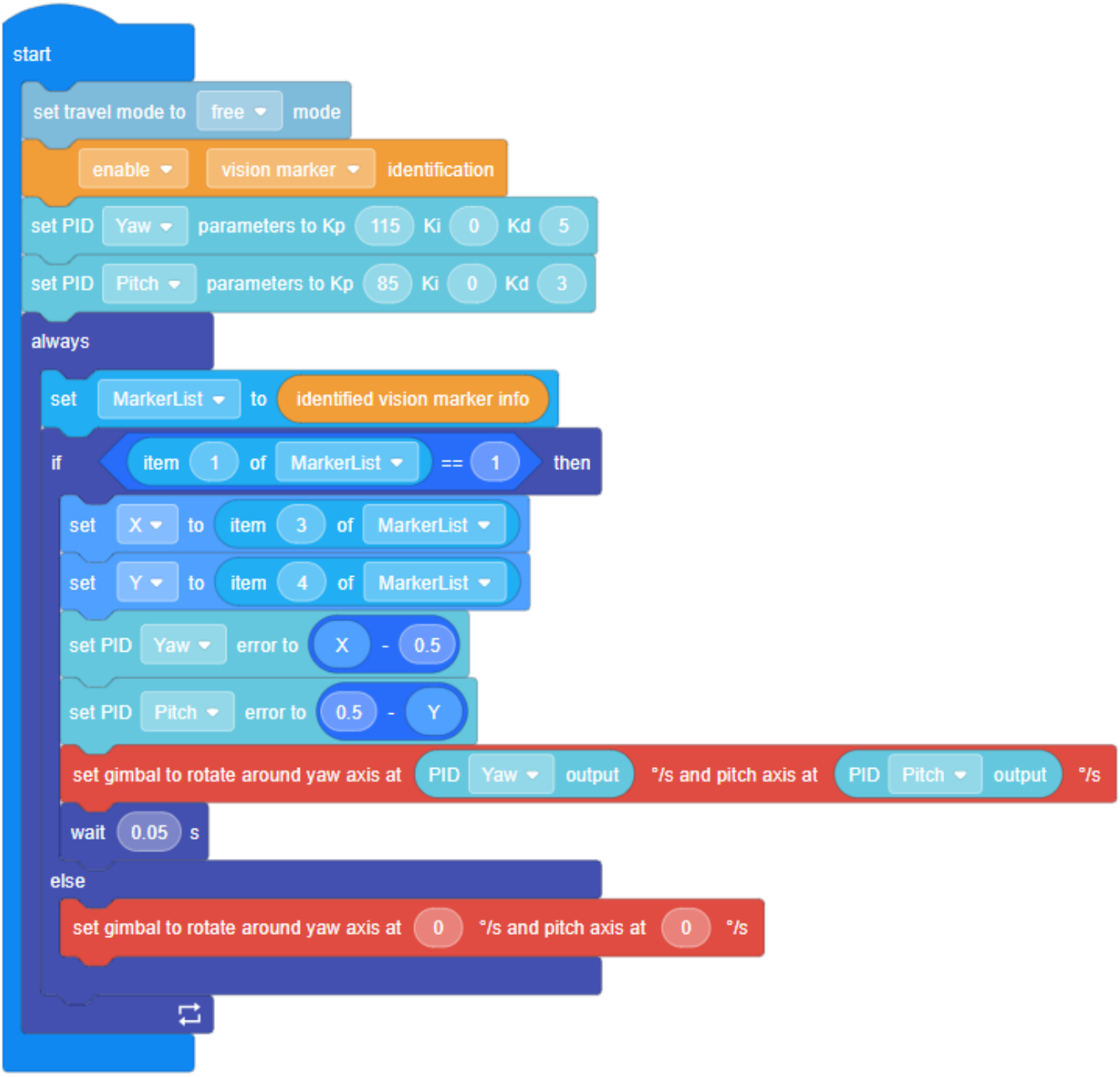
- Function:
  - set_ctrl_params(kp, ki, kd)
- Parameters:
  - kp(float)
  - ki(float)
  - kd(float)

20. PID (# PID#) output

(1) Objective: Obtains the output value for a PID

(2) Type: Information block (variable-type)

(3) Example: Follow Vision Marker



Python API:

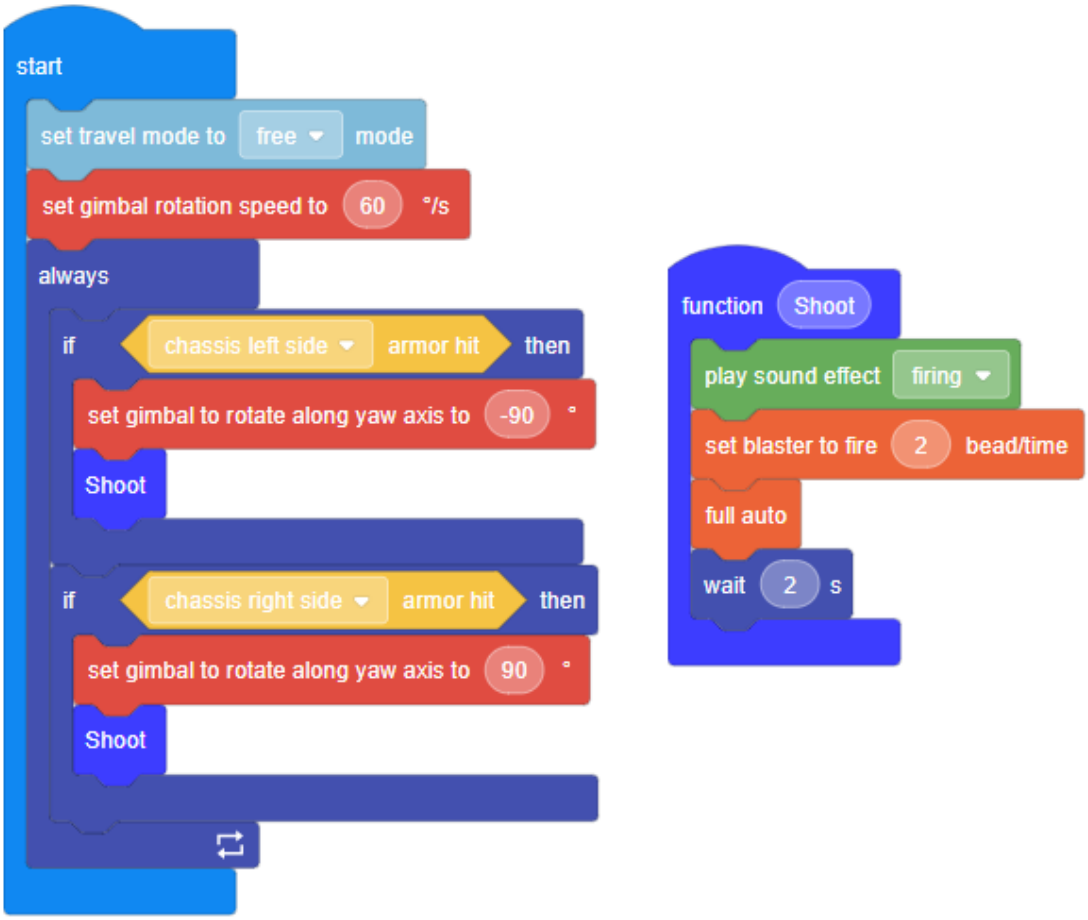
Class: rm_ctrl.PIDCtrl()

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

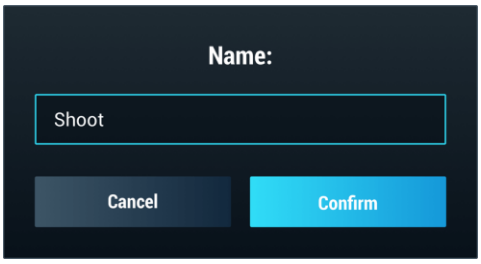
Functions

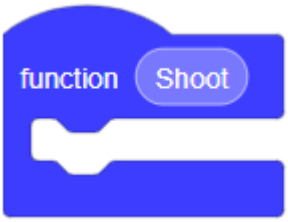
1.#function#

(1) Objective: Packages a program that needs to appear multiple times into a function, making it convenient to use
(2) Type: Function block
(3) Example: Launch counterattack
If the armor pieces on either side of the chassis are attacked, the gimbal will turn to the side where it was attacked to aim a counterattack.



Note:
1) Function names must begin with an underscore or a letter and can only contain numbers, uppercase and lowercase letters, and underscores.





2) After a function has been created, the packaged block will appear for use:

3) Using functions helps make the whole program more concise and clear.