dji EDUCATION

ROBOMASTER EP CORE

SCRATCH PROGRAMMING GUIDE

Catalog

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

Brief Introduction

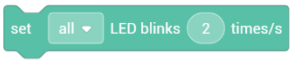
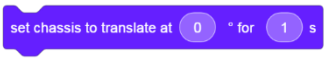
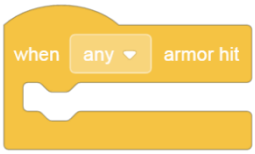
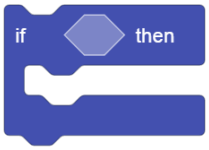
The RoboMaster EP CORE programming guide is designed to help new users quickly learn programming techniques for controlling the EP CORE.

The RoboMaster lab offers hundreds of graphical programming blocks that allow you to access specific features like EP CORE's PID control, intelligent identification 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 build new users' skills from nothing.

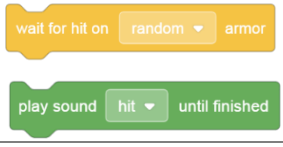
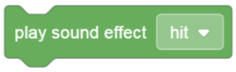
It is suggested to first read through the guide to gain a basic understanding of robotic programming. Afterward, you can refer to it for help with any questions or challenges you encounter while programming. We hope you find this resource helpful to improving your programming skills, making the most of each block, and learning new ways to win.

1. Block types

The RoboMaster EP CORE programming lab offers five block types:

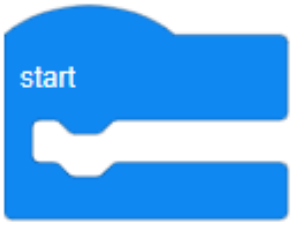
Block Type	Description	Block Example
Settings	Set parameters, such as speed, frequency, quantity and more.	
Execution	Control EP CORE to execute specified commands	
Event	Main function will pop out and begin to run programs included in event blocks when certain conditional statements are met	
Information	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

Block Type	Description	Block Example
Blocking	Follow-up commands will not be executed before blocking blocks stop running	
Non-blocking	Follow-up commands can be executed when non- blocking blocks are running	

System

1. Start

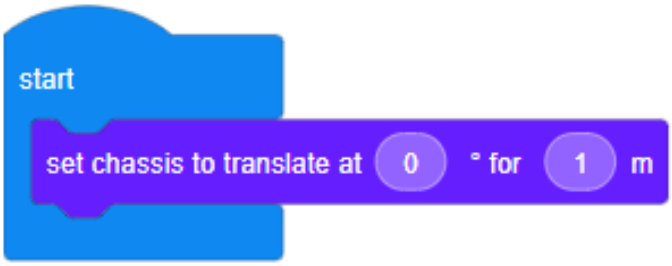


(1) Description: Set the first program executed when the robot powers on

(2) Type: Embedded block

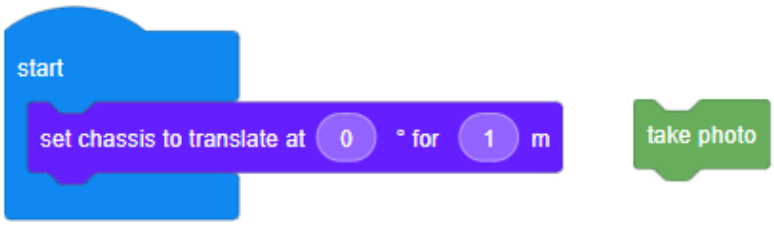
(3) Example: Move forward 1 meter

This will control the RoboMaster EP CORE to move forward by 1 meter.



Note:

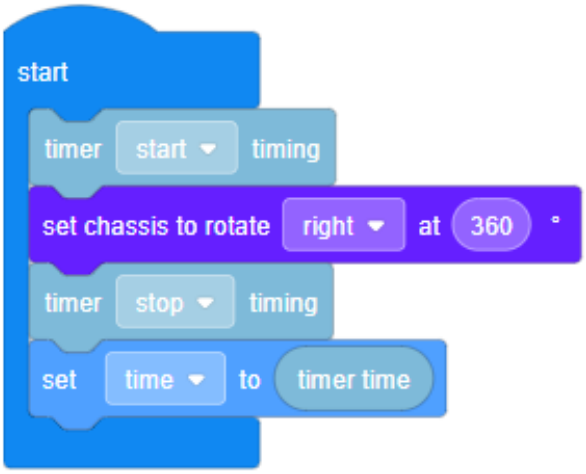
If blocks (other than event triggering blocks and function blocks) are not placed within the “Start” area, they will not be executed. For example, in the program shown below, the RoboMaster EP CORE will not be able to take a photo.



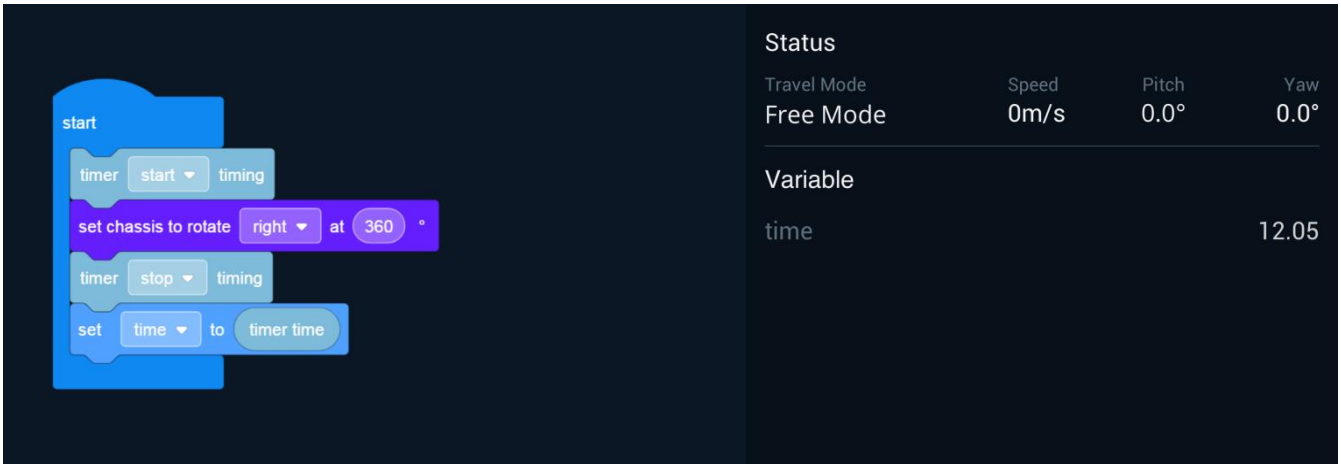
Python API:
Function: start( )
Type: Main function

2. Timer (start) timing

(1) Description: Start, pause, or stop the timer
(2) Type: Execution block
(3) Example: Time a rotation
This measures the time it takes the chassis to complete one full rotation.

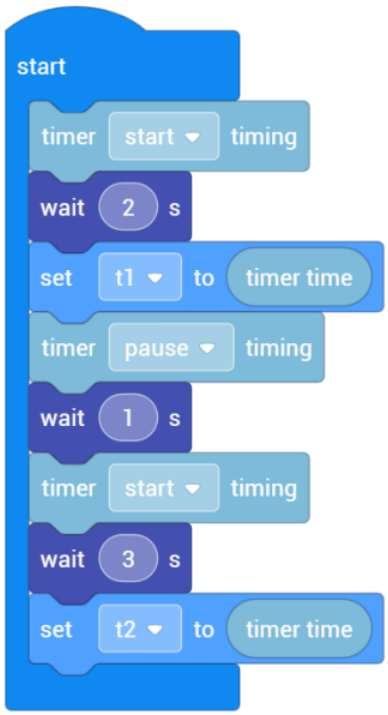


You can check details using the FPV window:

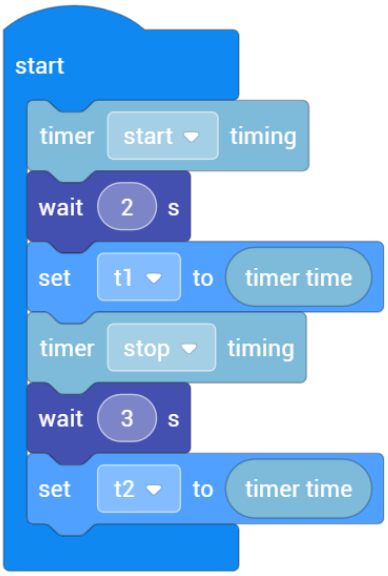


Notes:

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. 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

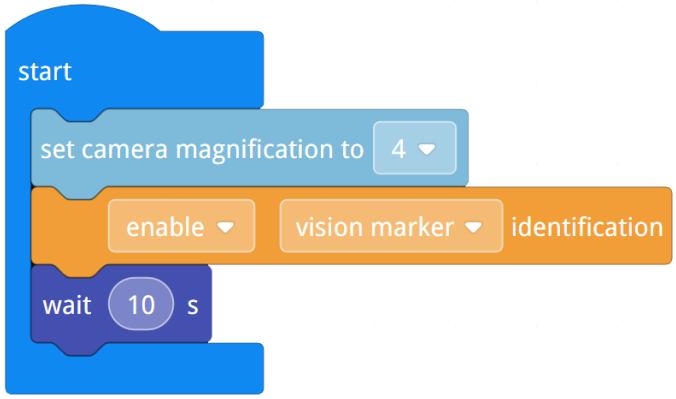
3. Set camera magnification to (1)

(1) Description: Apply higher magnifications to support visual recognition over longer distances, enabling the robot to focus on unclear objects more accurately

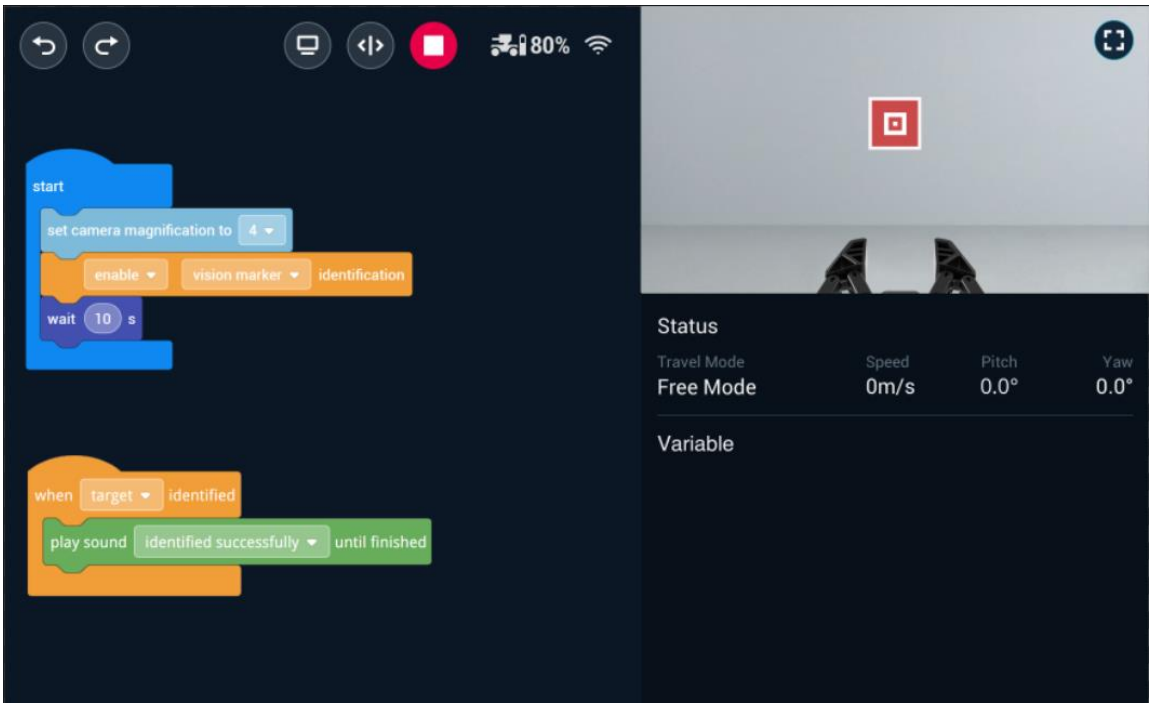
(2) Type: Execution block

(3) Example: Enlarge camera frame

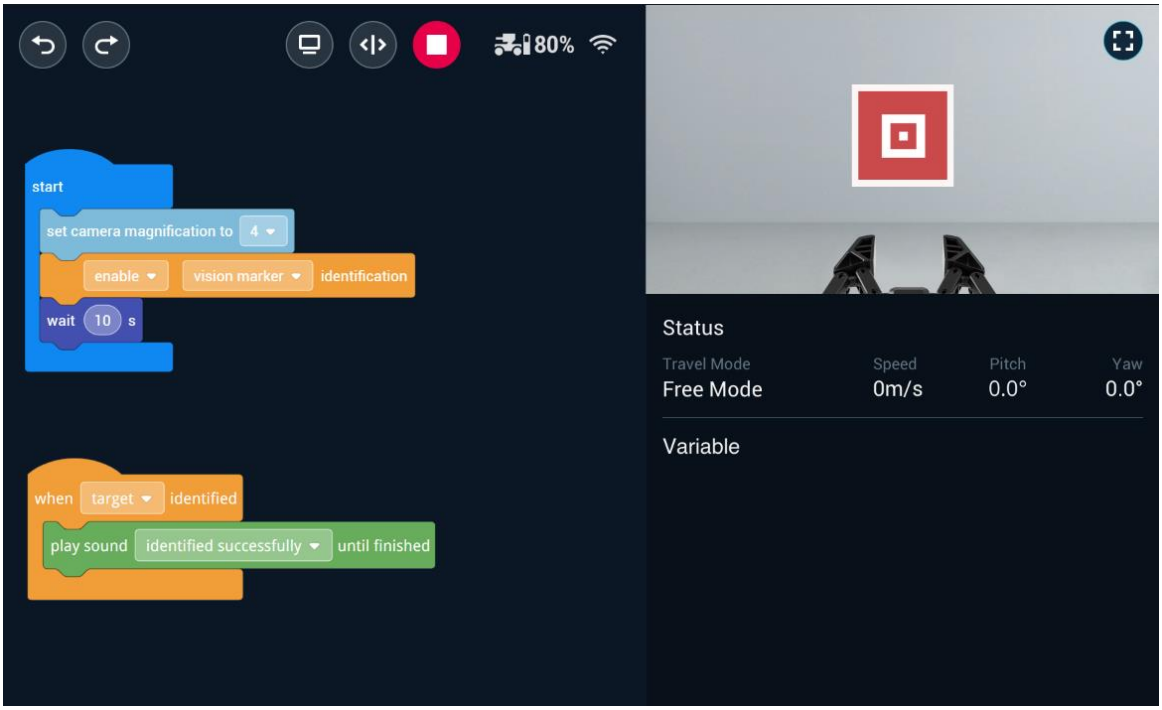
Place a Vision Marker 10 meters in front of the robot's gimbal and set the camera magnification to 4; the robot will now be able to accurately recognize the Vision Marker at this distance.



Before running the enlarge camera frame program:



After enlarging the camera frame:



Python API:

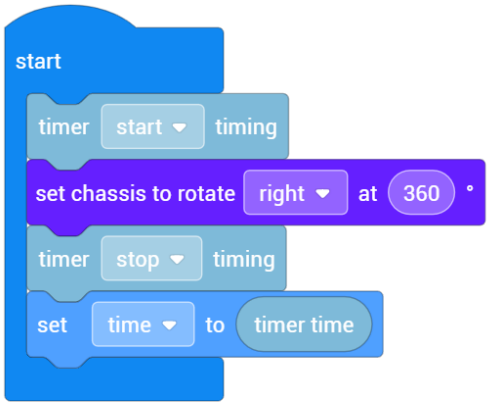
Function: media_ctrl.zoom_value_update(value)

Parameters:

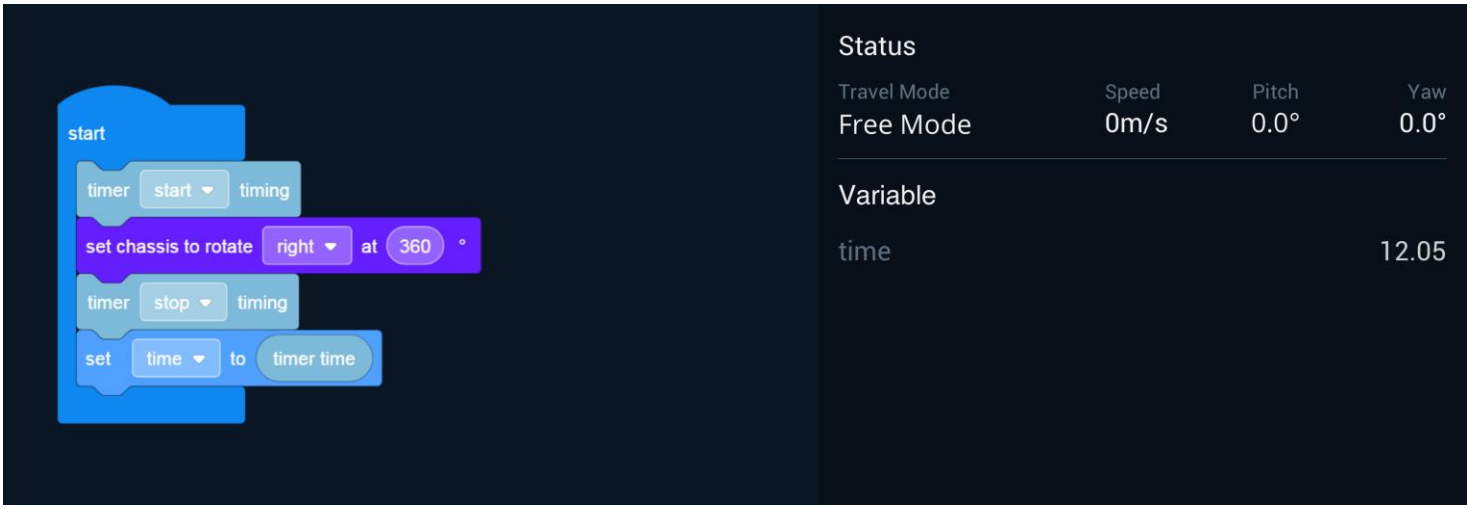
- value (int): [1, 4]

4. Timer time

(1) Description: Obtain the total time elapsed from when the timer first started to the current time (in seconds)
(2) Type: Information block (variable-type data)
(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:

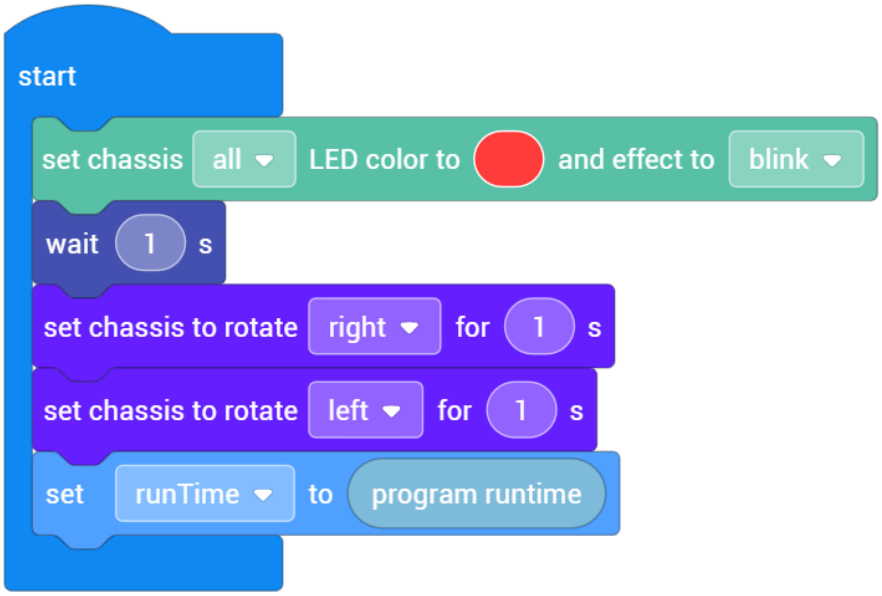
- time_stamp(float)

5. Program runtime

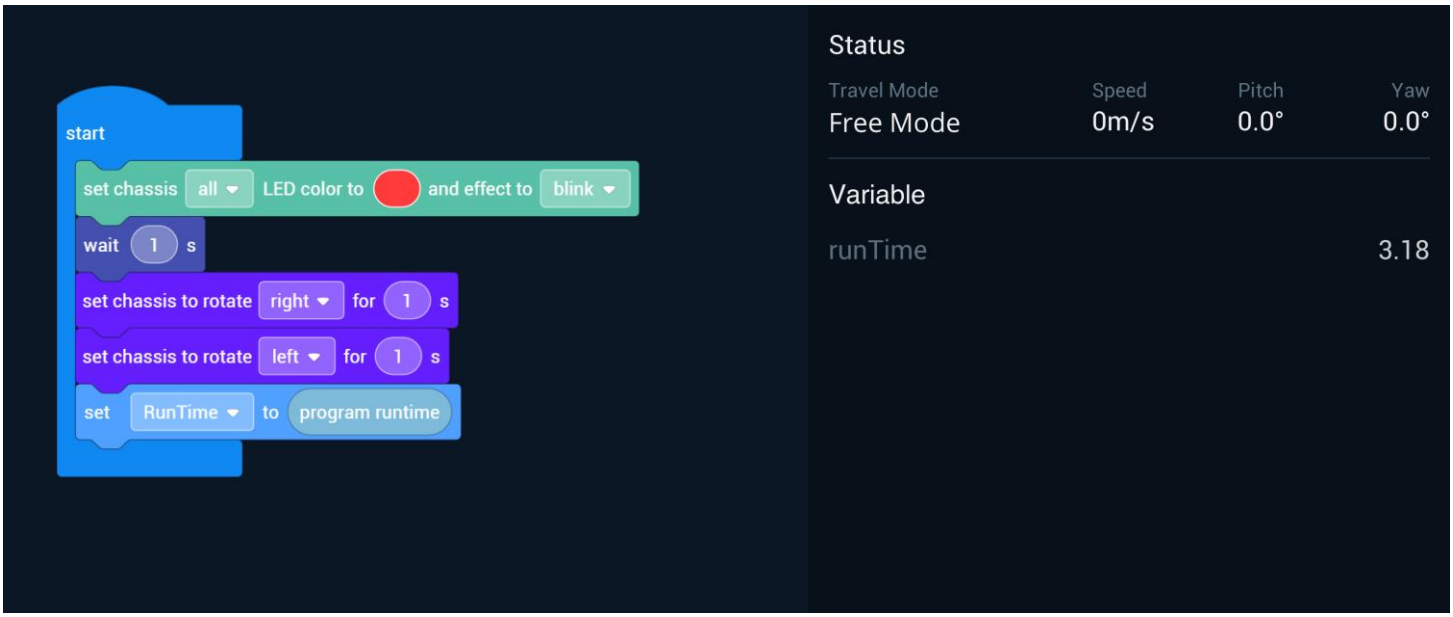
(1) Description: Obtain the program running time (in seconds)

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

(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)

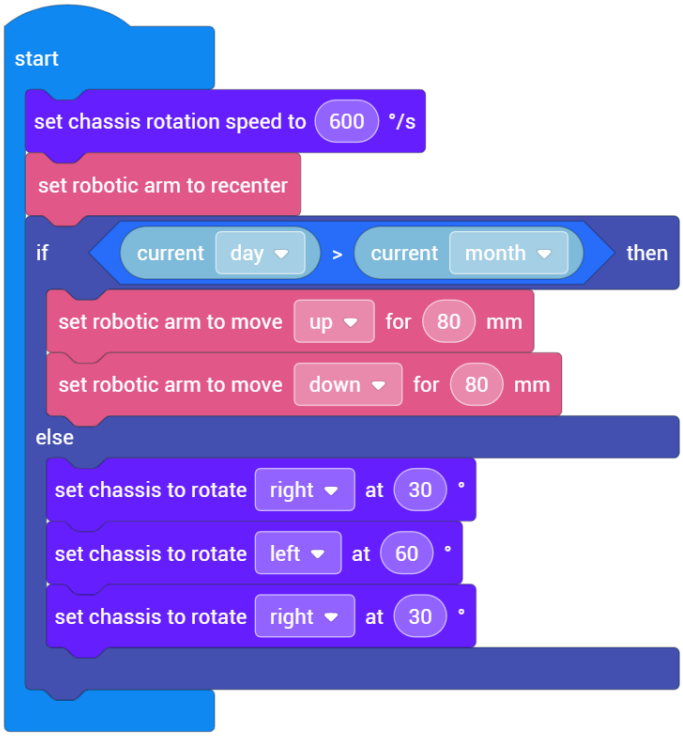
6. Current (year)

(1) Description: Acquire current time information including the year, month, day, hour, minute, second, etc.

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

(3) Example: Compare time values

If the number of the current month is larger than the number of the current day, the Robomaster EP CORE will move its robotic arm up and down; if the number of the current month is less than or equal to the number of the current day, the EP CORE will rotate its chassis from left to right.



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)

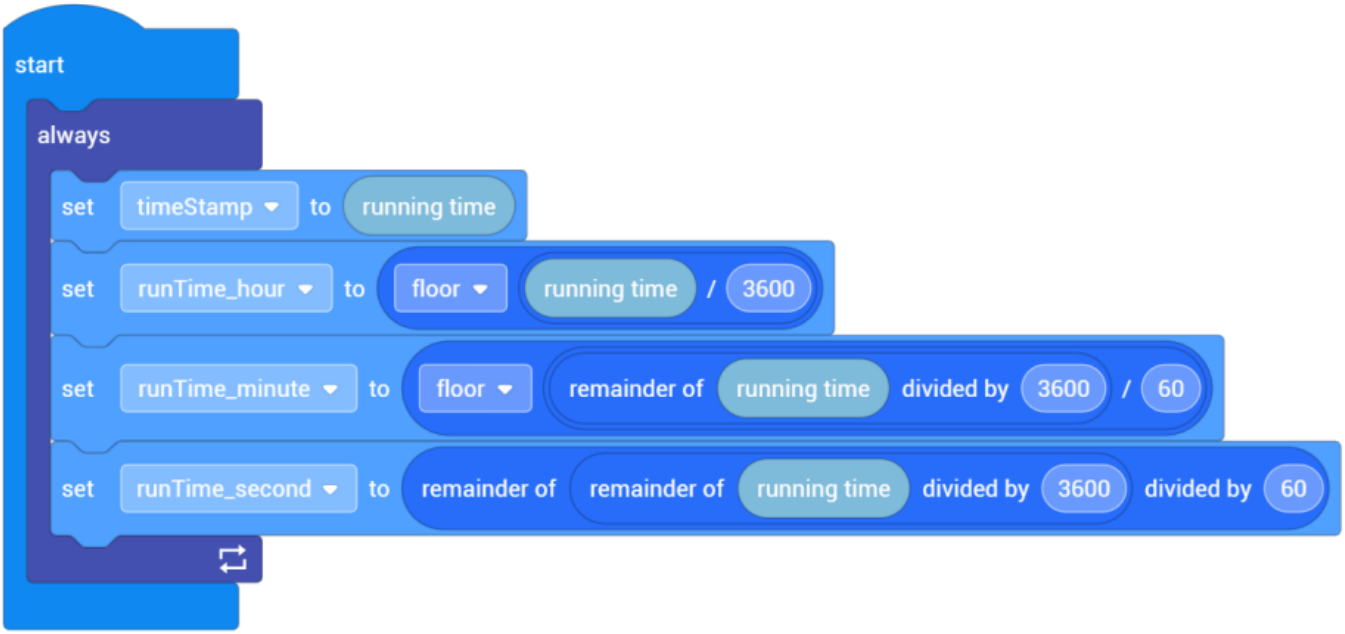
7. Running time

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

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

(3) Example: Calculate running time

This measures the total time elapsed from when the robot most recently started running up to the current time (in 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 over one hour. (i.e., when runTime_hour>1).

Notes:

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:

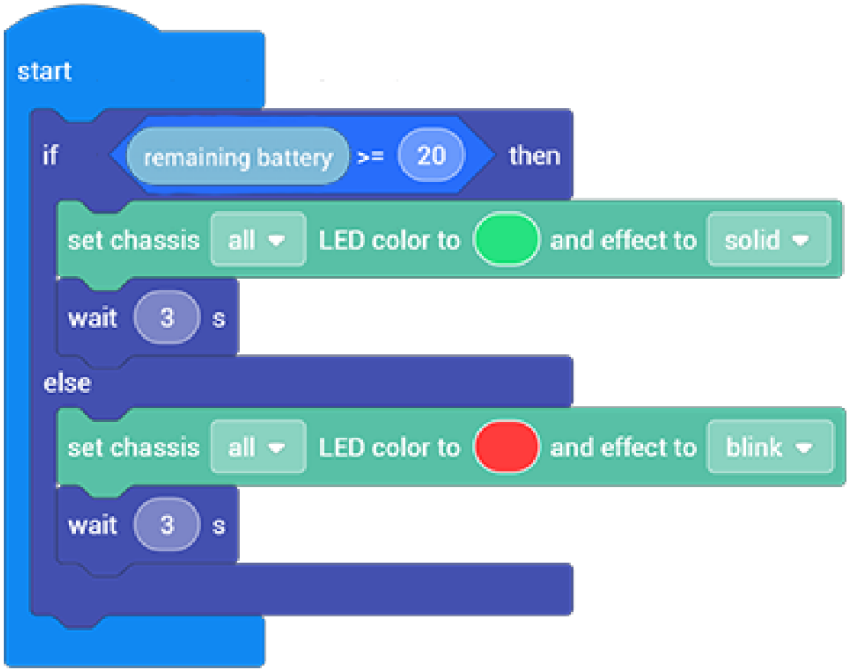
● time (float)

8. Remaining power

(1) Description: Obtain the remaining power of the robot, and return an integer between 0 and 100 to indicate the remaining power percent

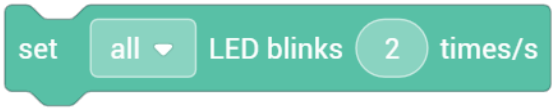
(2) Type: Information block

(3) Example: Power percent on Chassis LED
If the current remaining power of the robot is 20% or more, the chassis LED will be solid green. Otherwise, the chassis LED blinks red.

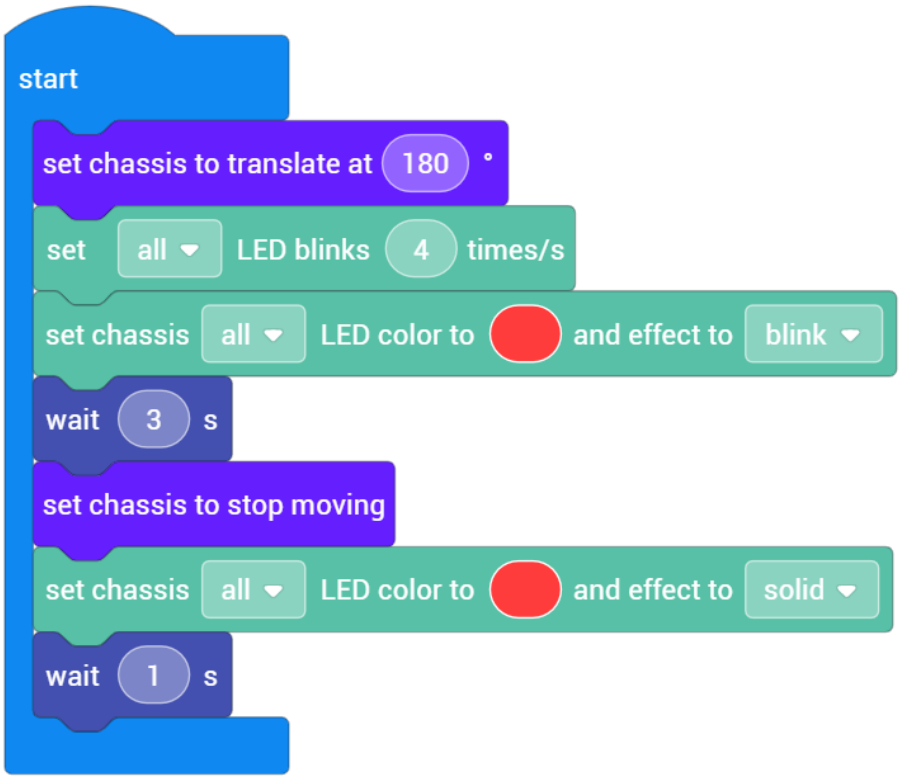


LED Effects

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



(1) Description: Set 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 RoboMaster EP CORE reversing.



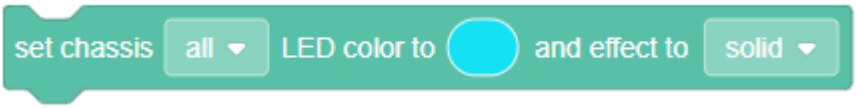
Python API:
Function: led_ctrl.set_flash(armor_enum, frequency)
Parameters:

- 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)



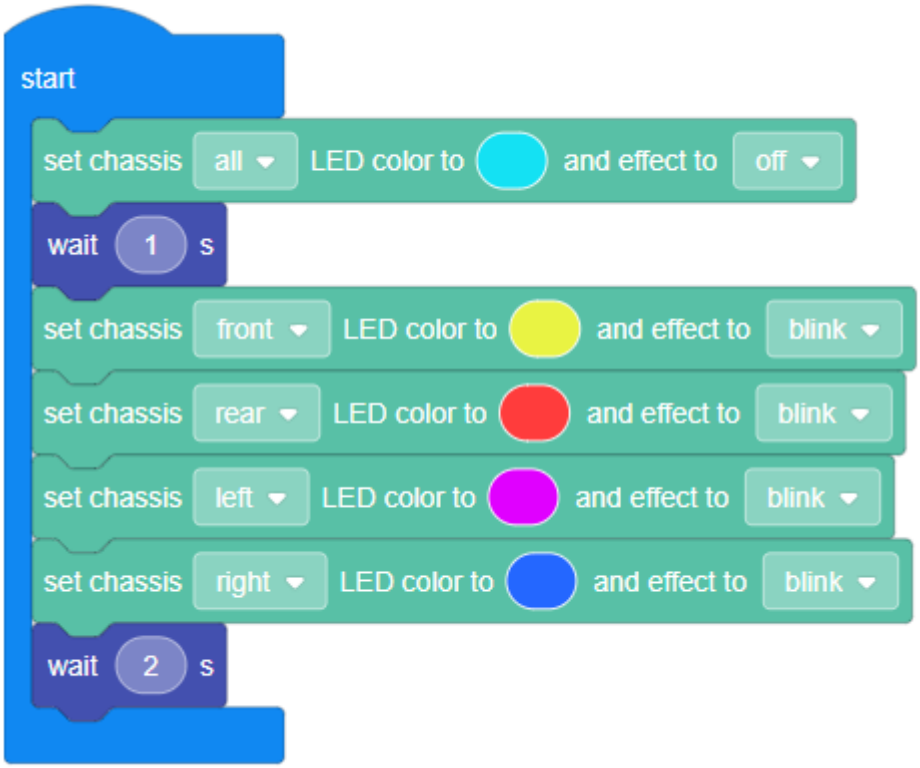
(1) Description: Set chassis LED colors and effects:

- Solid: LED will remain steady
- Off: LED will switch off
- Pulse: LED will flicker (from dark to bright and then back to dark)
- 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

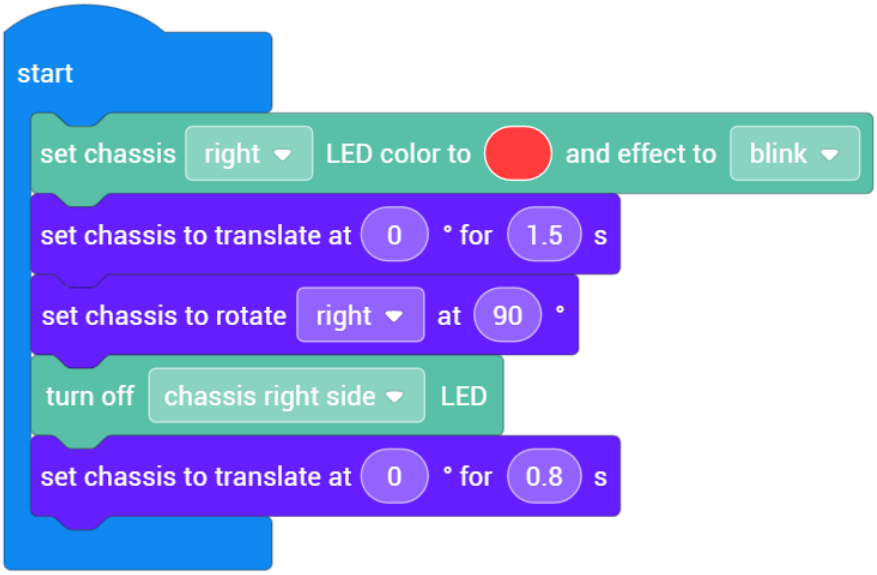
3. Turn off (all) LEDs

(1) Description: Switch off designated LEDs

(2) Type: Execution block

(3) Example: Configure signal light

This will set the LEDs of RoboMaster EP CORE to flash on the right side of the chassis 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

Chassis

1. Set (PWM_all) output to (7.5)

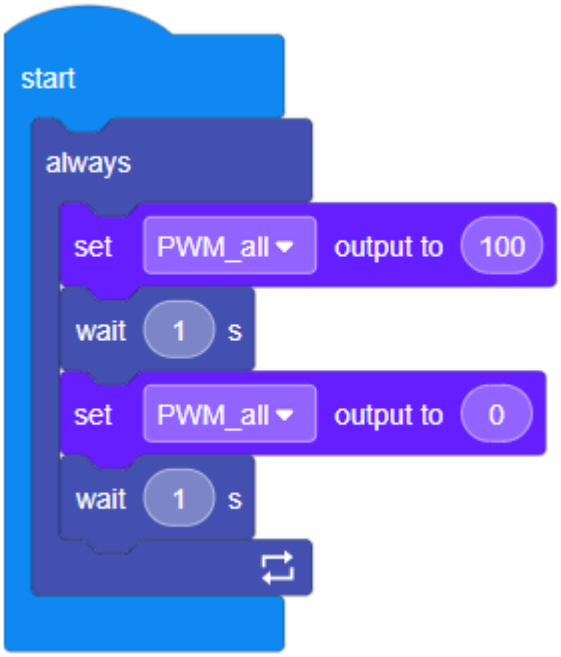
(1) Description: Set the output percentage for the PWM port; the larger the value used, the longer the port will maintain a high level of output over the specified time period. The basic frequency for the PWM port is 50Hz.

(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.

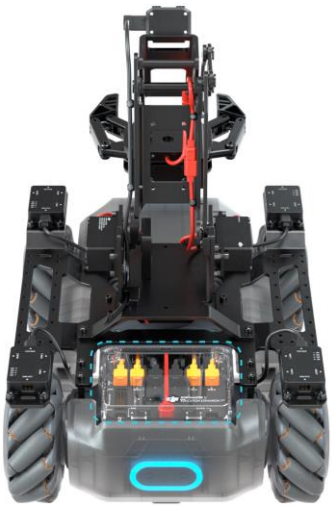


② Rotate navigation gear

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

Notes:

1) PWM port is located on the chassis motion control module and can be seen by removing the transparent cover on the rear side of the chassis.



There are six PWM ports in total.

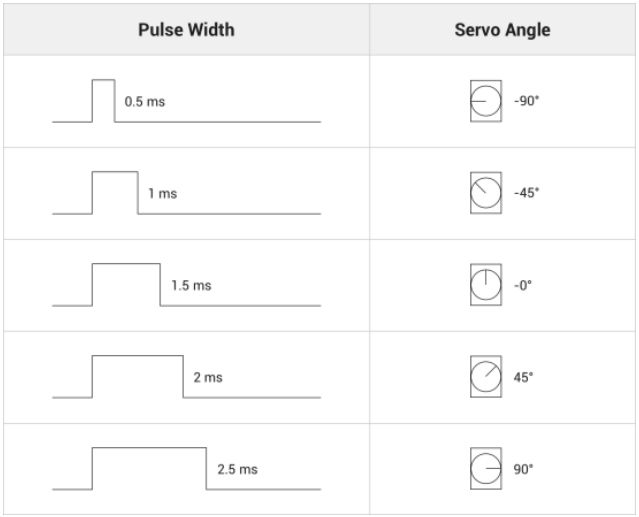
2) PWM (pulse width modulation) controls the duration of a high level of output during a certain period, and is broadly used to control LEDs, navigation gears, and more.

3) When the robot is powered on, PWM port outputs a signal of 7.5% PWM for default. When every program is running to the end, the output signal will return to its default setting.

4) 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.

5) 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 20ms, 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.



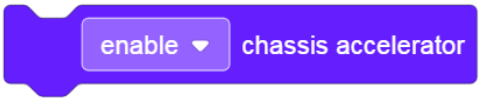
Python API:

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

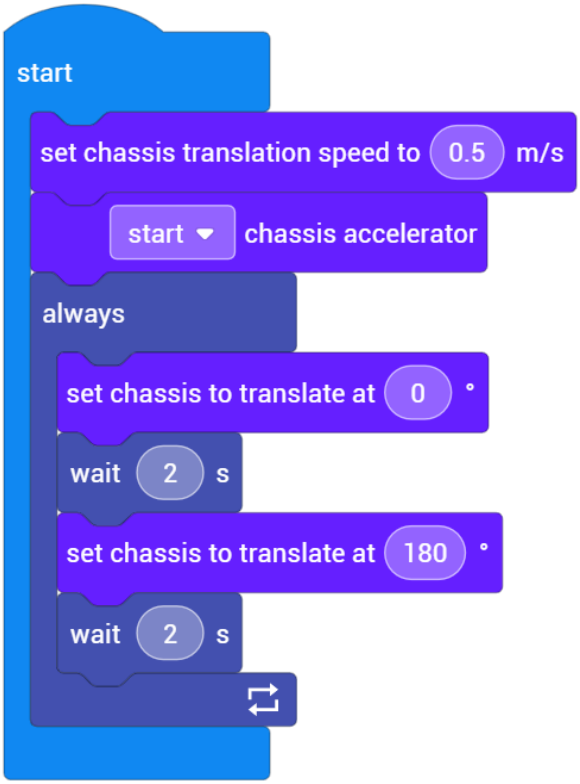


(1) Description: Enable/disable the chassis accelerator

(2) Type: Settings block

(3) Example: Operate chassis accelerator

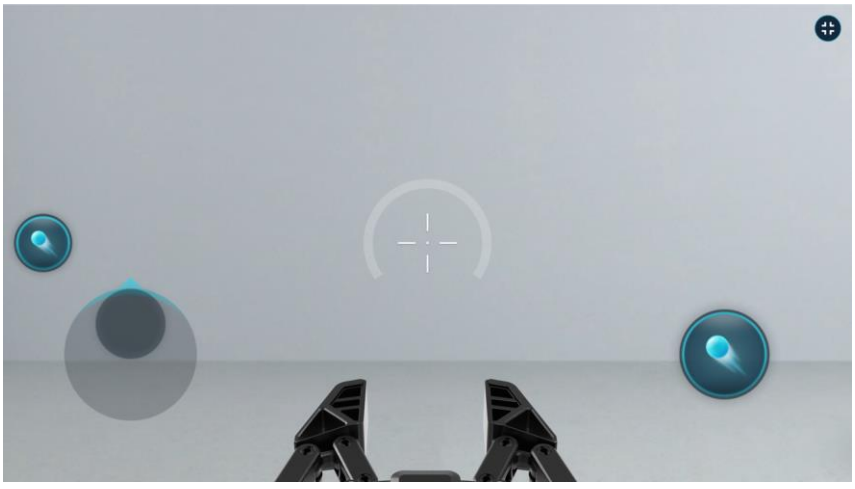
When the chassis is moving automatically, this will enable you to use the joystick to manually rotate or translate the chassis and increase its translation speed.



Notes:

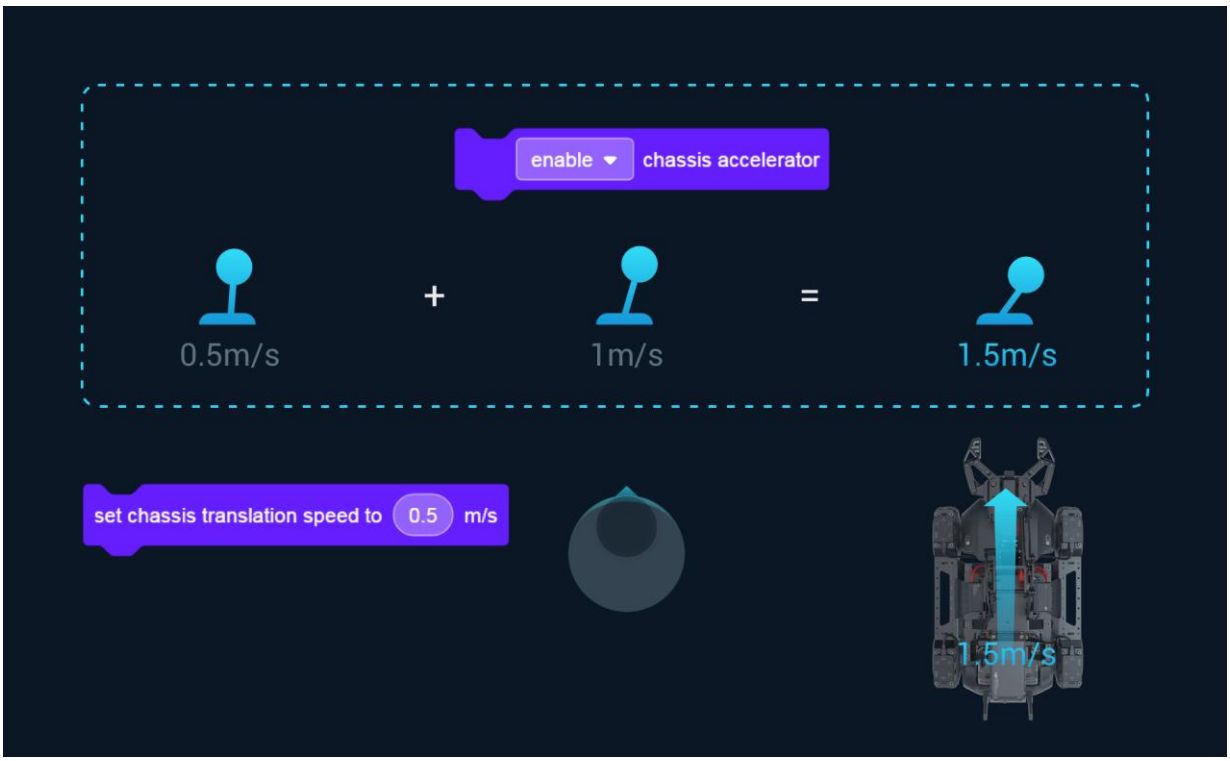
1) If the “Enable chassis accelerator” block has not been added, you will not be able to manually control the chassis while running the program. After adding this block, you will be able to manually control and accelerate the robot’s movements.
2) Joystick sensitivity refers to the push range of the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

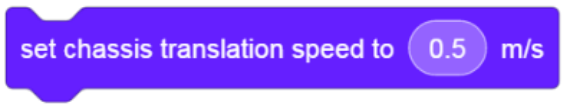


Python API:

Function: chassis_ctrl.enable_stick_overlay( )

chassis_ctrl.disable_stick_overlay( )

3. Set chassis translation speed to (0.5)m/s

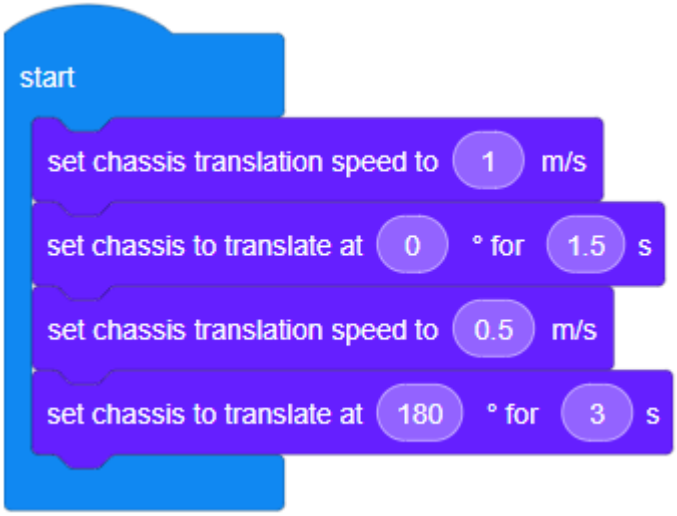


(1) Description: Set the default translation speed of the chassis to 0.5 m/s; the chassis will move faster when set to a higher speed value.

(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 3 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.

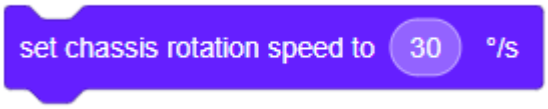
Python API:

Function: chassis_ctrl.set_trans_speed(speed)

Parameters:

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

4. Set chassis rotation speed to (30)°/s

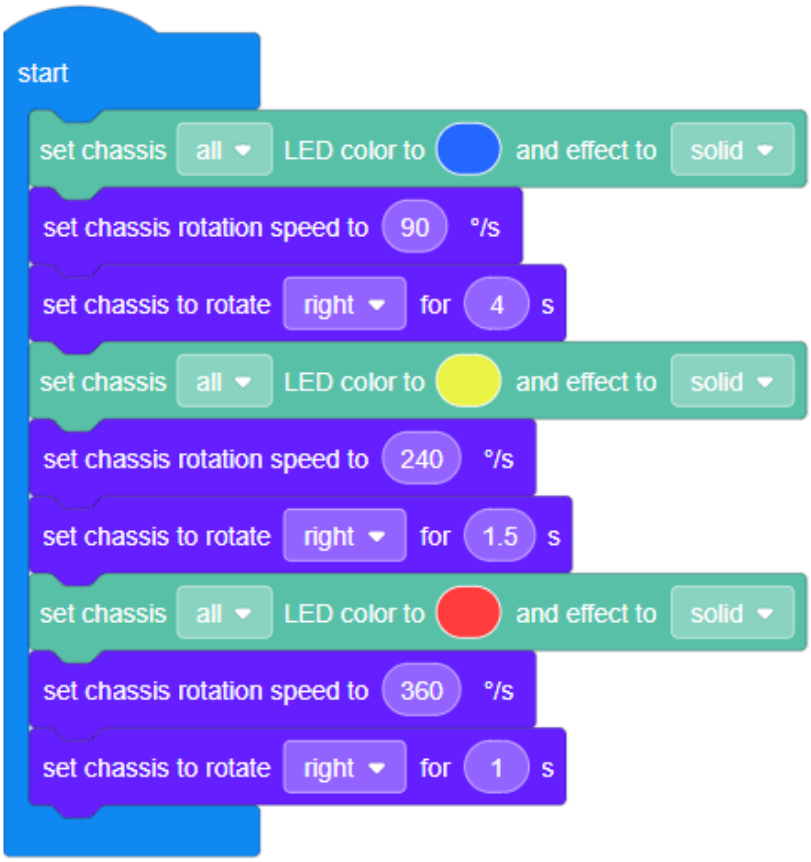


(1) Description: Set the default rotation speed of the chassis to 30°/s; the chassis will rotate faster when set to a higher rotation speed value.

(2) Type: Settings block

(3) Example: Set acceleration warning

As the warning LED color changes, the rotation of the chassis will accelerate.



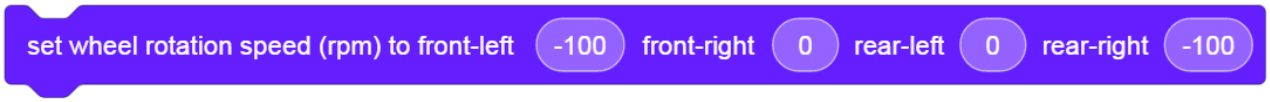
Python API:

Function: chassis_ctrl.set_rotate_speed(speed)

Parameters:

- speed(int): [0, 600] °/s

5. Set wheel rotation speed (rpm) to front-left (100) front-right (100) rear-left (100) rear-right (100)



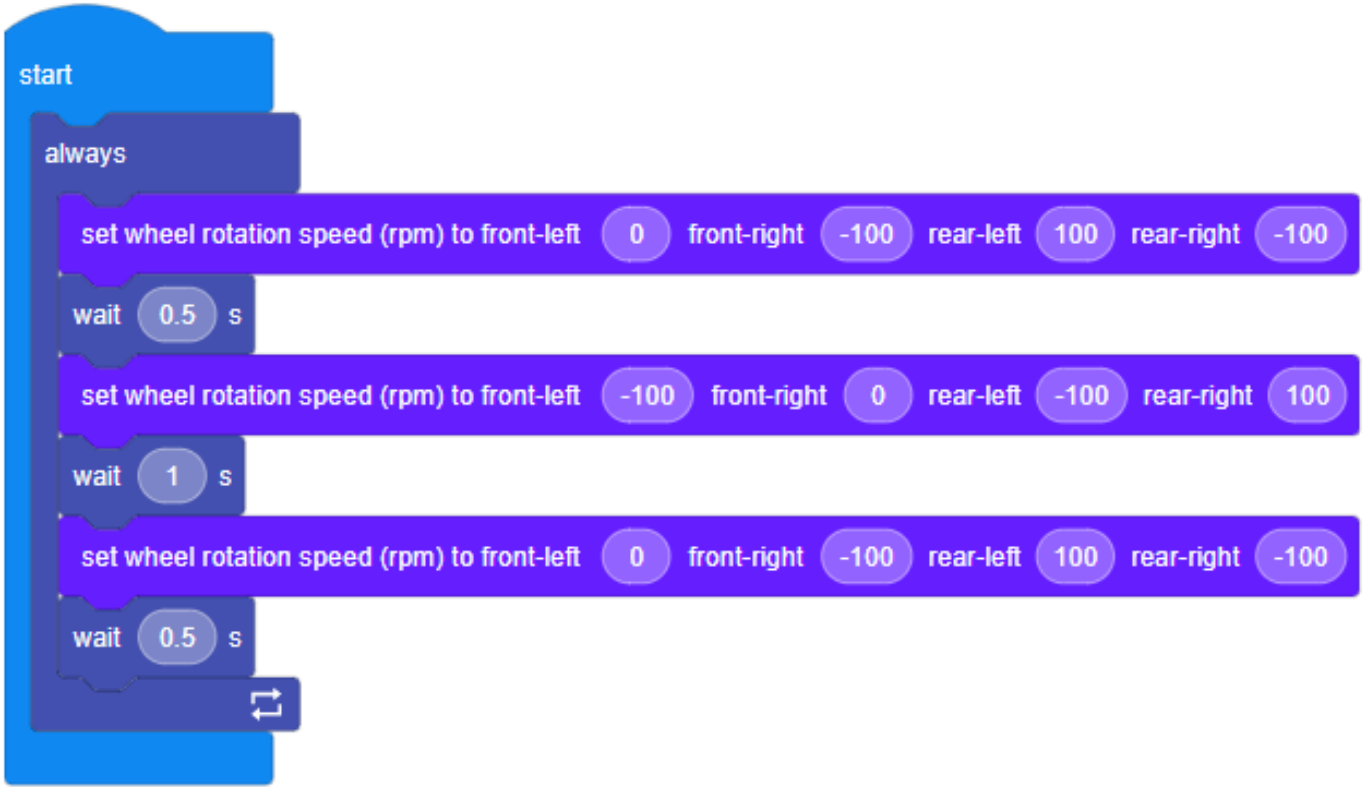
(1) Description: Allow each wheel's rotation speed to be set independently; only valid combinations of rotation direction and speed will take effect.

(2) Type: Execution block

(3) Examples: Reversing in an "S" pattern, Translating in a circle

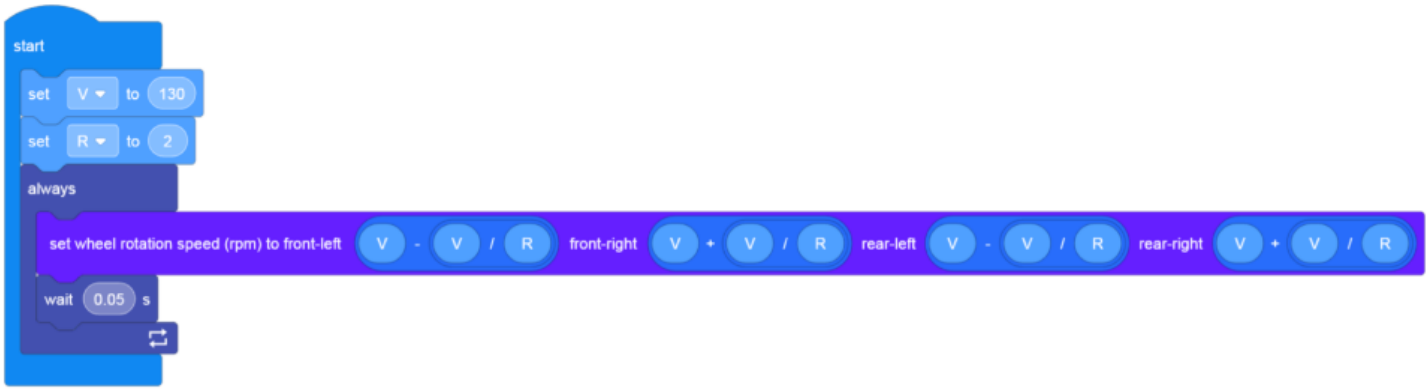
① Reversing in an "S" pattern

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.



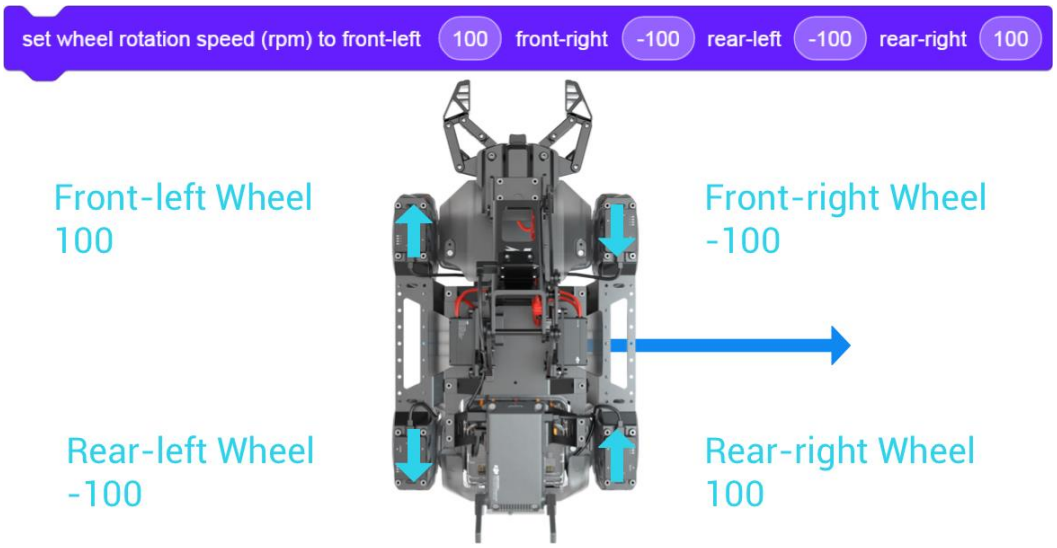
Notes:

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,

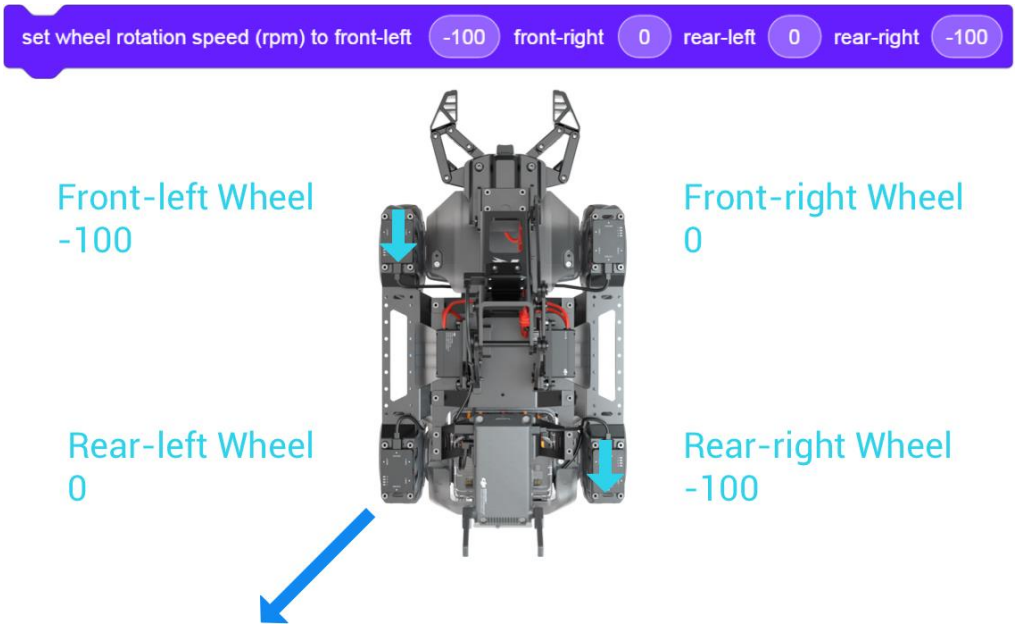
push the robot manually to move it in the desired pattern and observe the rotational direction of each wheel; the rotation speed's value should be positive for wheels rotating forward and negative for wheels rotating backward.

For example:

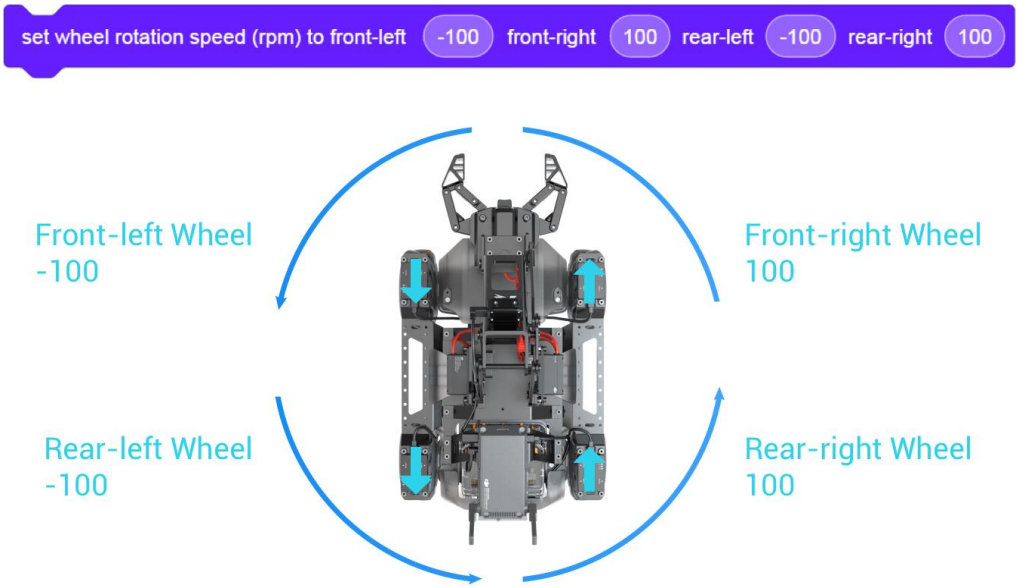
When the robot translates to the right, the front-left wheel and rear-right wheel will rotate forward, so their rotation speed values should be positive, while the front-right wheel and the rear-left wheel will rotate backward, so their rotation speed values should be negative.



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.



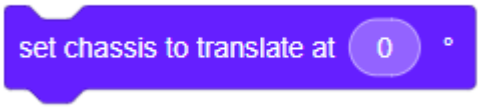
Python API:

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

6. Set chassis to translate at (0)°

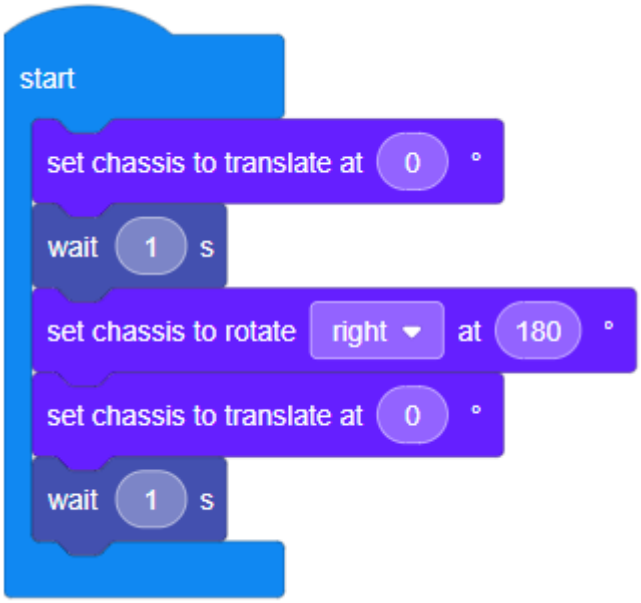


(1) Description: Set the chassis to translate in a specified direction

(2) Type: Execution block

(3) Example: Make a round trip

This will control the RoboMaster EP CORE 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 (x) s,” or other command that controls the chassis movement.

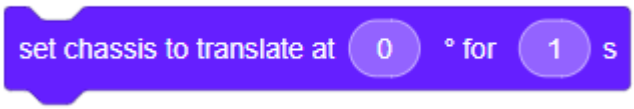
Python API:

Function: chassis_ctrl.move(degree)

Parameters:

- degree (int): [-180, 180] °

7. Set chassis to translate at (0)° for (1)s

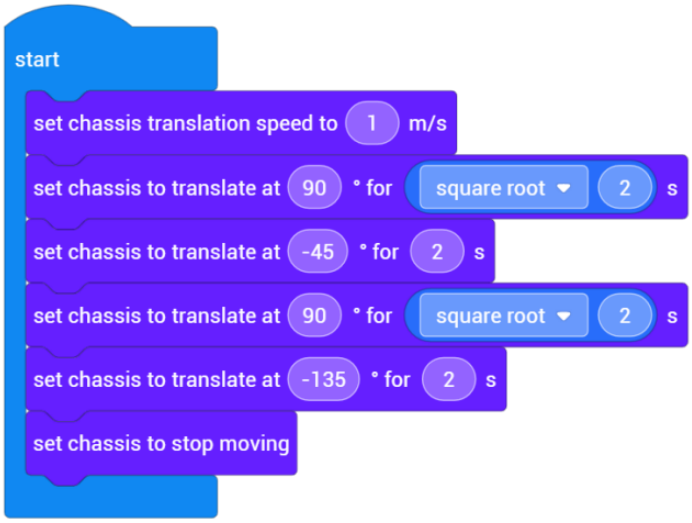


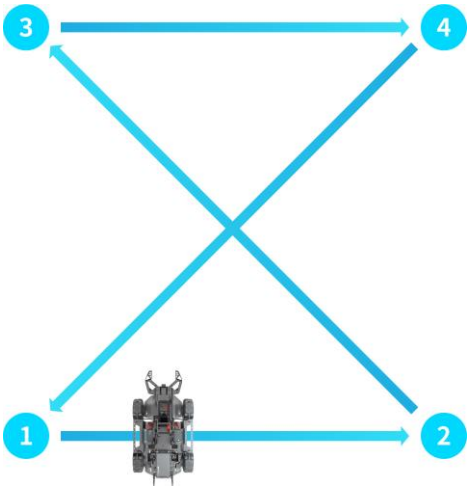
(1) Description: Set the duration of time the chassis will translate in a specified direction

(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.





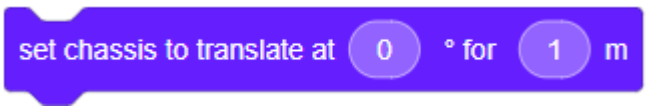
Python API:

Function: chassis_ctrl.move_with_time(degree, time)

Parameters:

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

8. Set chassis to translate at (0)° for (1) m

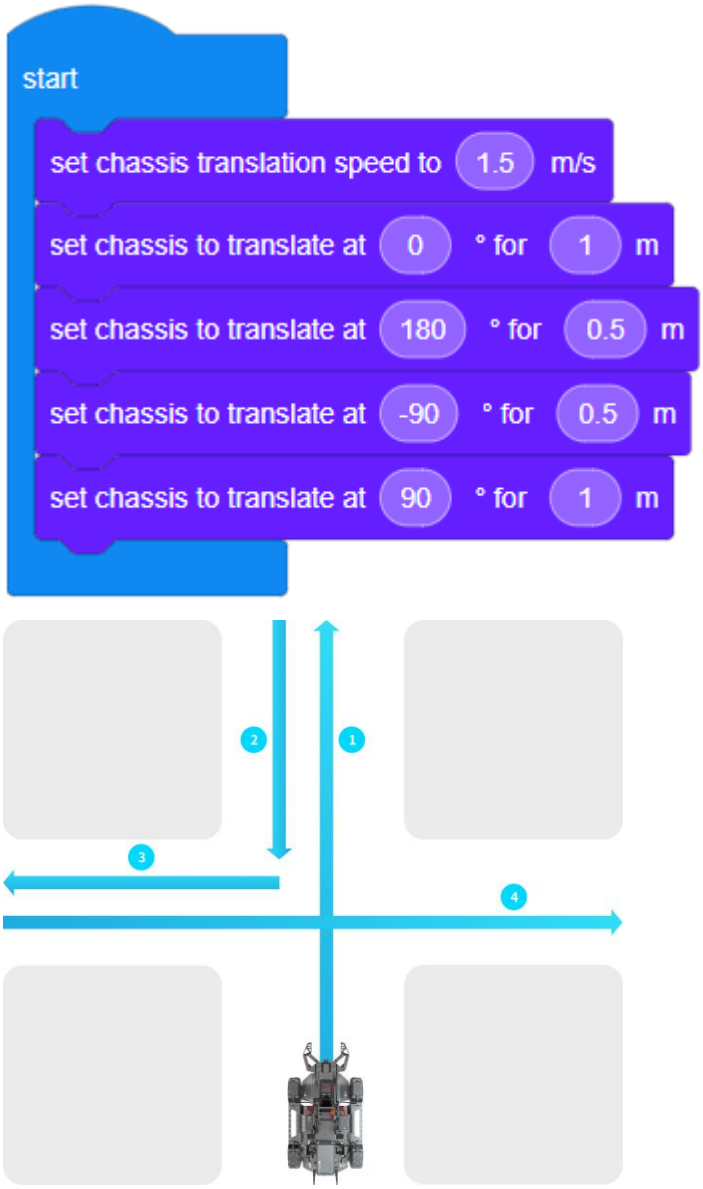


(1) Description: Set the distance the chassis will translate in a specified direction

(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

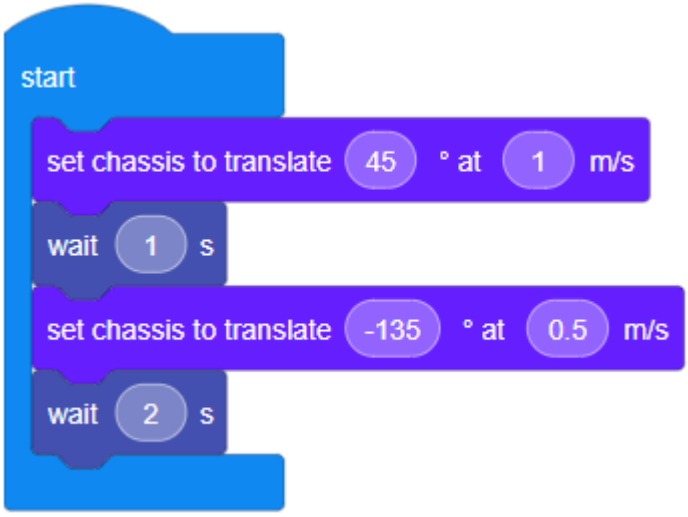
9. Set chassis to translate (0)° at (0.5)m/s

(1) Description: Set 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 0.5m/s to return to the original position.



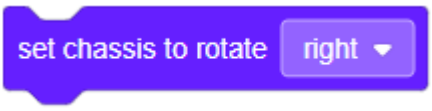
Python API:

Function: chassis_ctrl.move_degree_with_speed(speed, degree)

Parameters:

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

10. Set chassis to rotate (right)



(1) Description: Set 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.

Notes:

1) Be sure there are no obstacles around the robot before setting a high chassis rotation speed.

2) This block will set the chassis to rotate constantly in a specified direction until it receives a "set chassis to stop moving", "wait (x) s" or other command that controls the chassis movement.

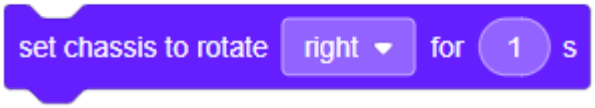
Python API:

Function: chassis_ctrl.rotate(direction_enum)

Parameters:

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

11. Set chassis to rotate (right) for (1) s

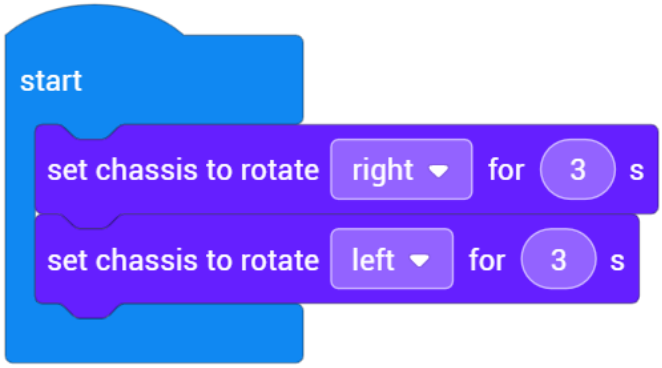


(1) Description: Set the duration the chassis will rotate in a specified direction

(2) Type: Execution block

(3) Example: Rotate left and right

The chassis will rotate right for 3 seconds and then rotate left for 3 seconds to return to the original position.



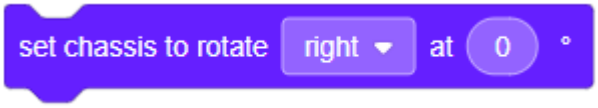
Python API:

Function: chassis_ctrl.rotate_with_time(direction_enum, time)

Parameters:

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

12. Set chassis to rotate (right) at (0) °

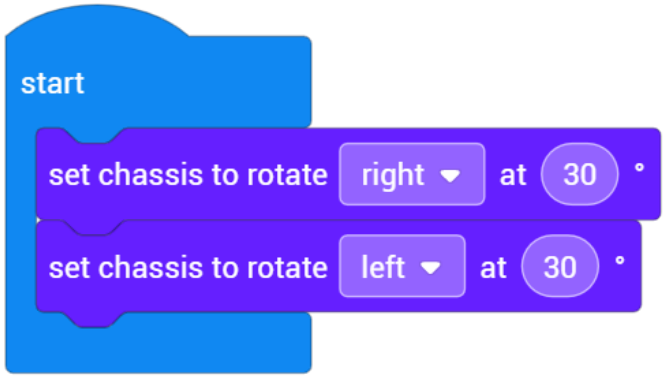


(1) Description: Set the angle and direction of chassis rotation

(2) Type: Execution block

(3) Example: Rotate left and right

The chassis will rotate right at 30° and then rotate left at 30°.



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] °

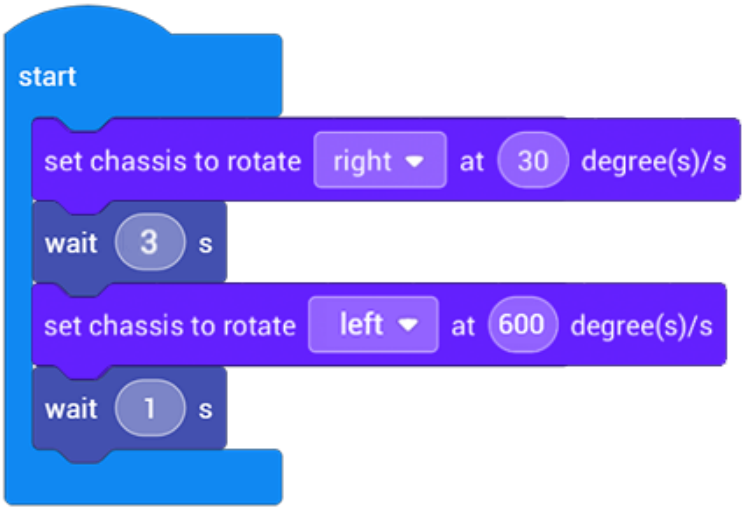
13. Set chassis to rotate (right) at (30)°/s

(1) Description: Set the chassis to rotate in a specified direction and at a specified rotation rate

(2) Type: Execution block

(3) Example: Rotate at variable speed

The chassis will rotate right slowly and then rotate left fast.



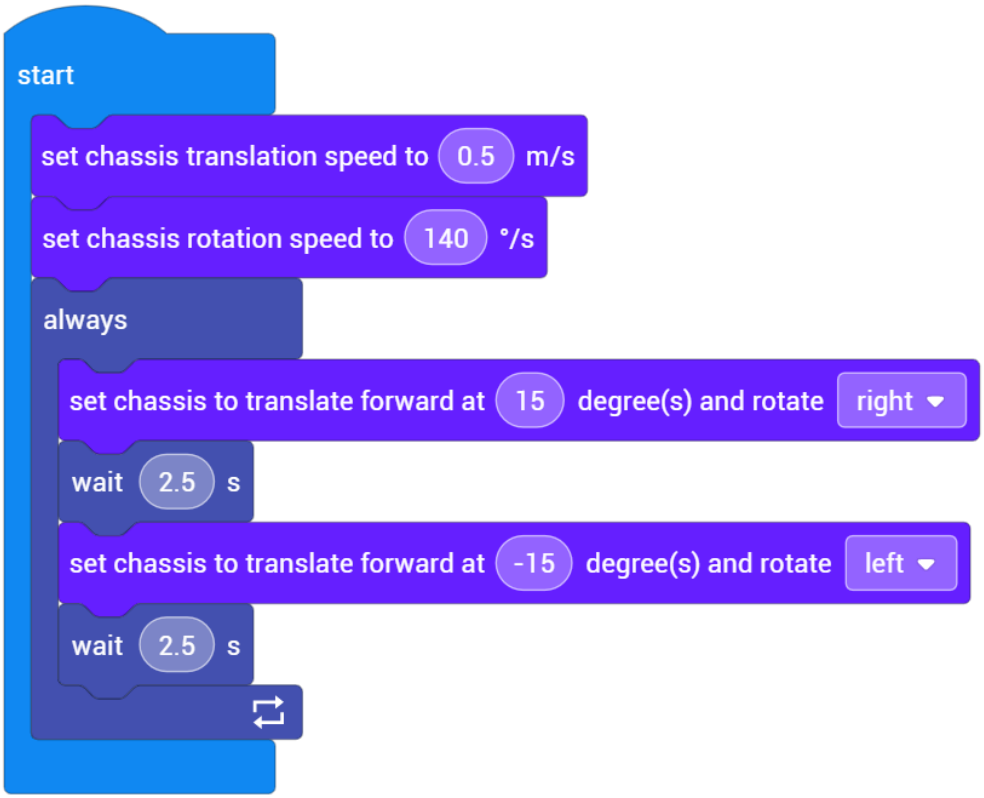
14. Set chassis to translate forward at (0)° and rotate (right)

(1) Description: Set 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 pattern.



Python API:

Function: chassis_ctrl.move_and_rotate(degree, direction)

Parameters:

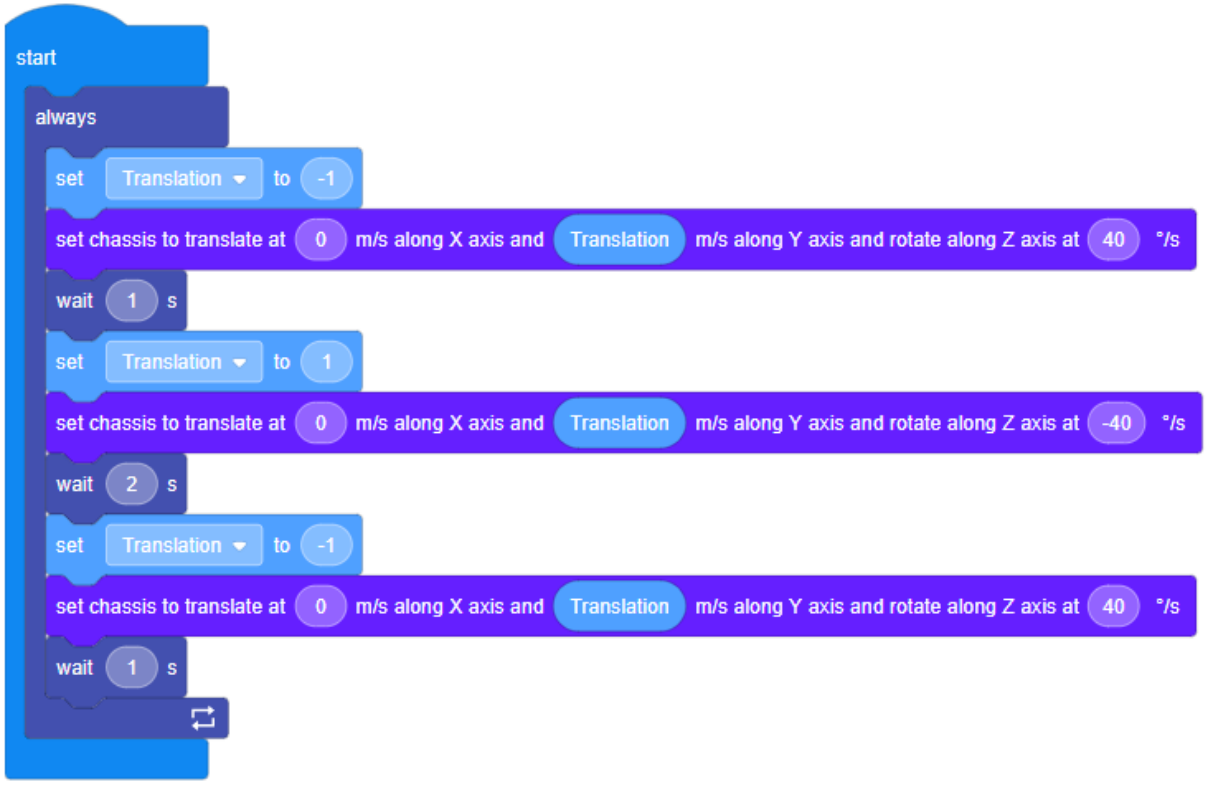
- degree(int): [-180, 180] °
- direction_enum(enum):
  - rm_define.clockwise
  - rm_define.anticlockwise

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) Description: Set the chassis to translate in a specified direction and at a specified speed

(2) Type: Execution block

(3) Example: Dishwashing moves



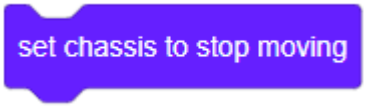
Python API:

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

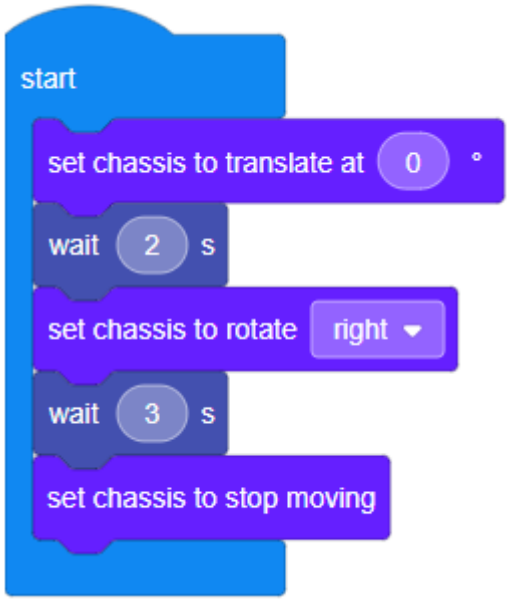


(1) Description: Stop all chassis movements

(2) Type: Execution block

(3) Example: EP CORE rotates right and stops

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( )

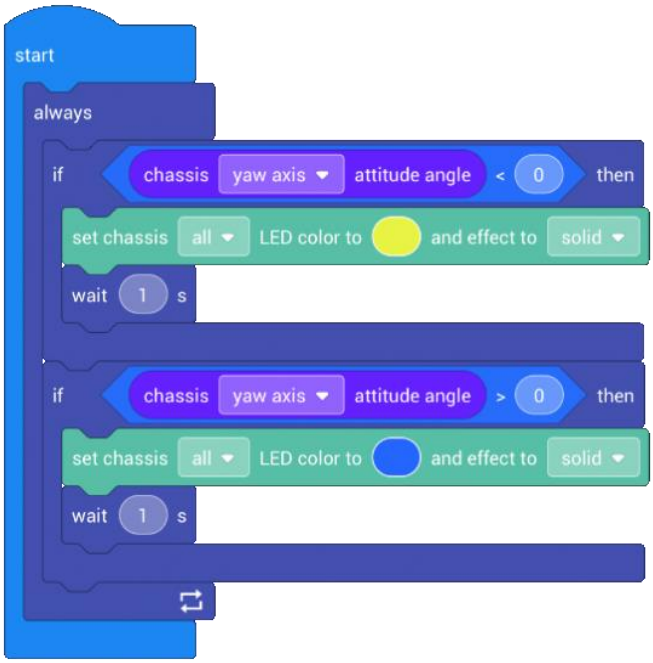
17. Chassis (yaw) axis attitude angle

(1) Description: Obtain 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 data)

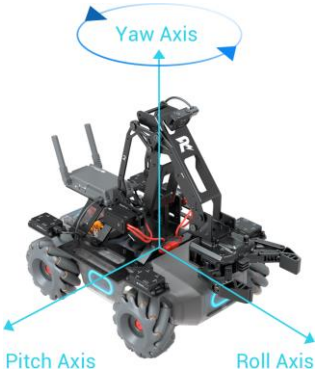
(3) Example: Signal turn

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.

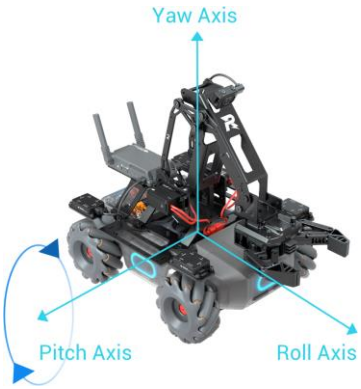


Notes:

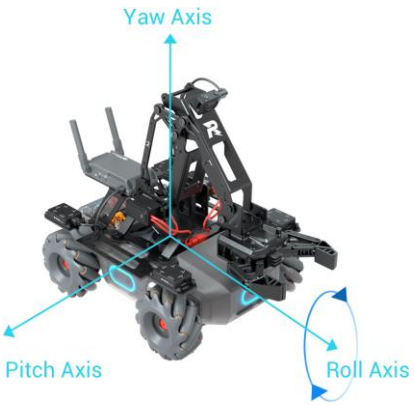
1. Rotating the chassis along the yaw axis results in left and right rotation.



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)

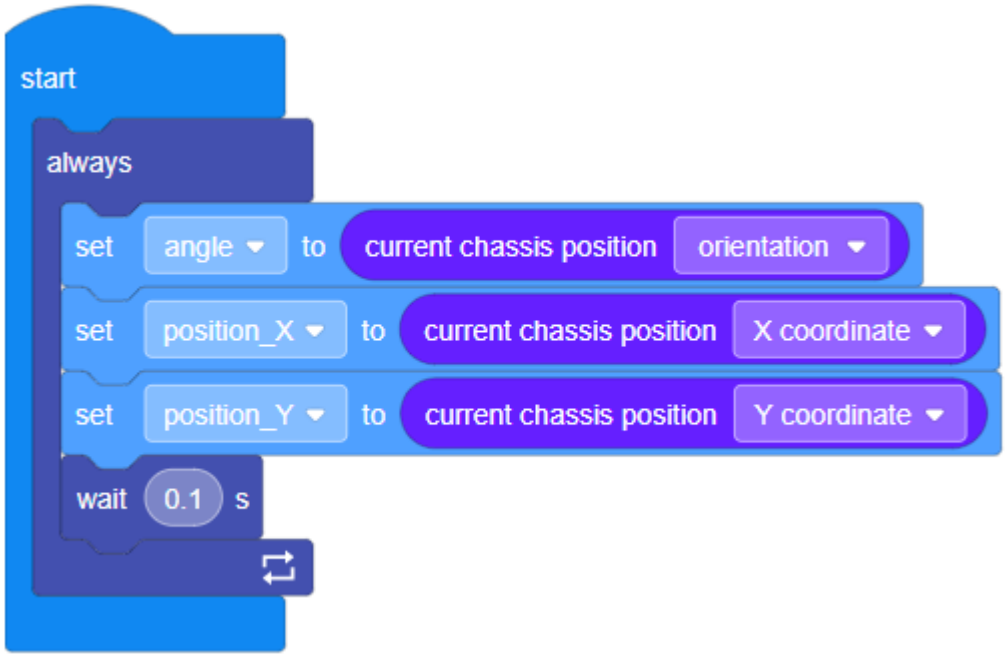
18. Current chassis position (x coordinate)

(1) Description: Obtain the current location coordinates and orientation of the chassis

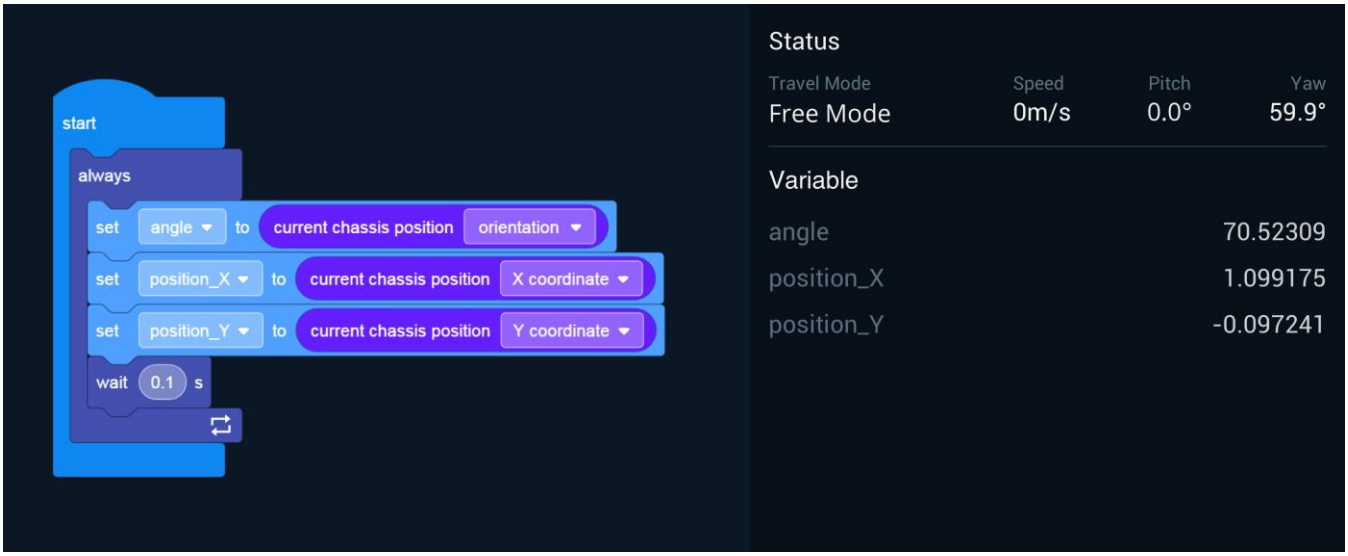
(2) Type: Information block (variable-type data)

(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 check details using the FPV window:



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:

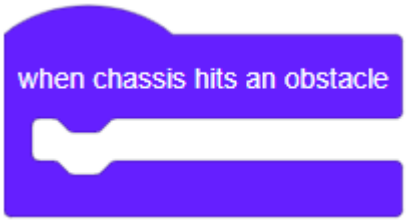
- action_enum(enum):

- rm_define.chassis_forward
- rm_define.chassis_translation
- rm_define.chassis_rotate

Return value:

- position(float)

19. When chassis hits an obstacle

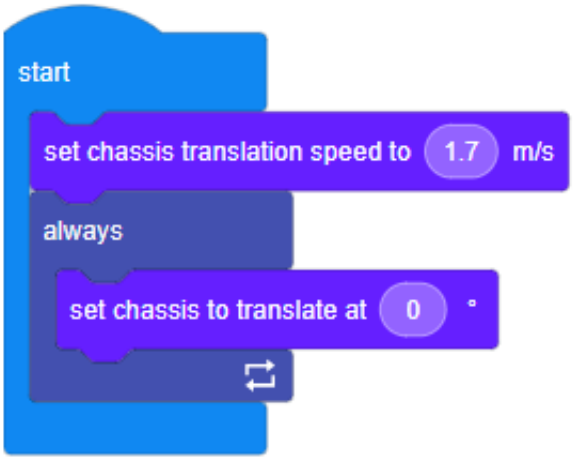


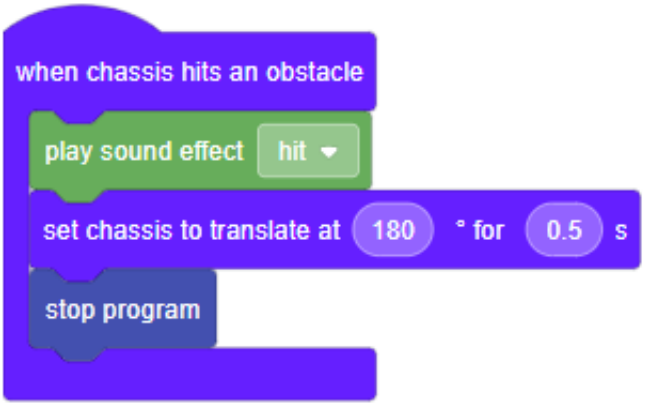
(1) Description: Run the program for the corresponding block when the chassis hits an obstacle (people, table legs etc.) while driving

(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

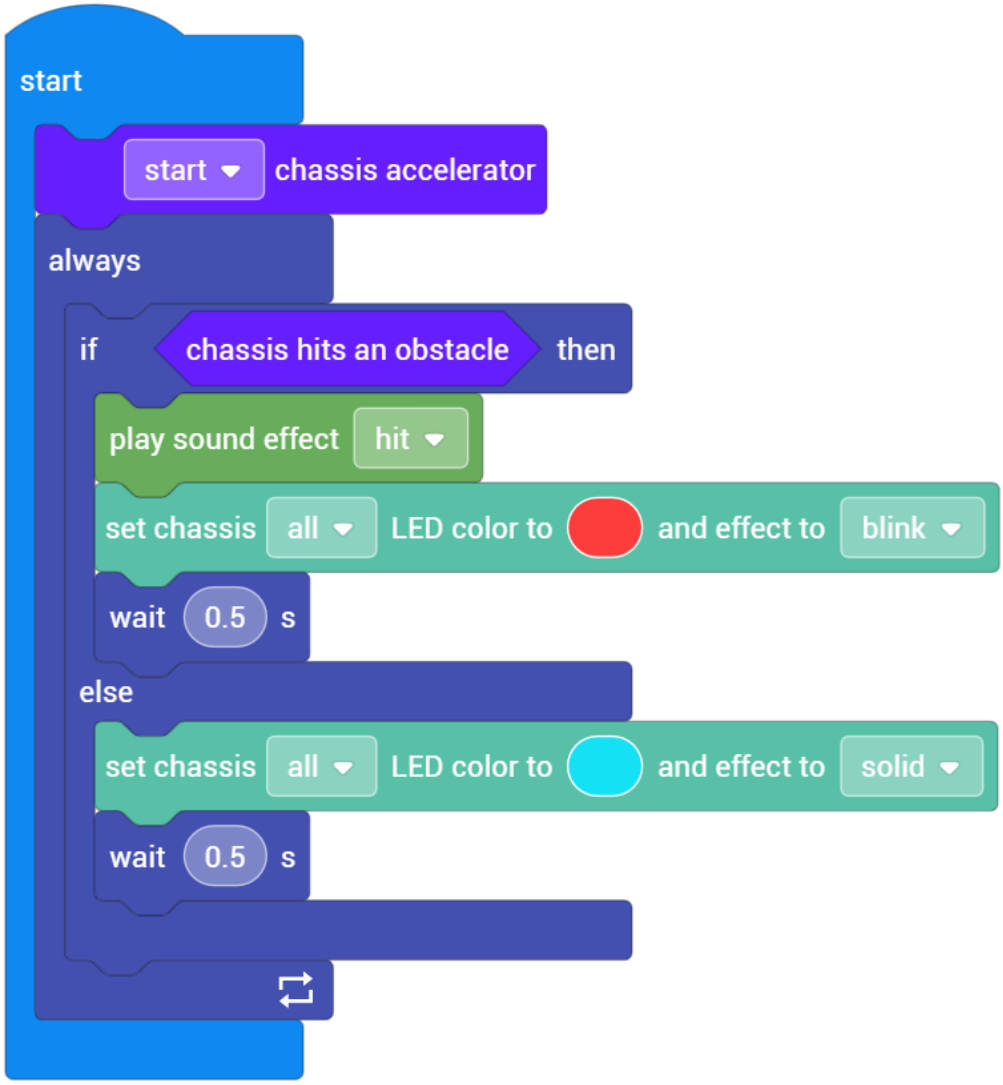
20. Chassis hits an obstacle

(1) Description: Return “True” when the chassis hits an obstacle (people, table legs etc.) while driving; otherwise, returns “False”

(2) Return value: Boolean

(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 will blink; otherwise, all LED chassis indicators will remain in the default color.



Python API:

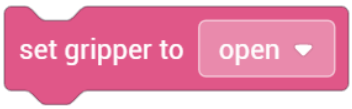
Function:chassis_ctrl.is_impact( )

Return value:

● impact_status(bool)

Extension Module

1. Set gripper to (open)

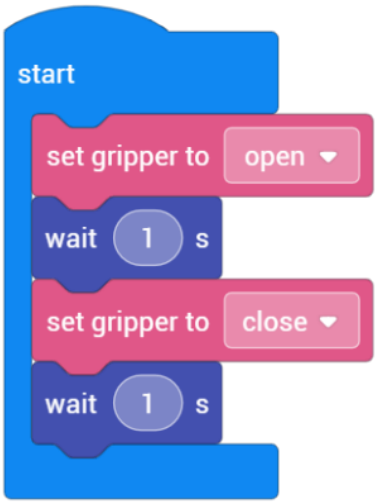


(1) Description: Set gripper to open, close, or stop.

(2) Type: Execution block

(3) Example: Flexible gripper movement

Set gripper to first open then close.



Notes:

1) "Set gripper (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.

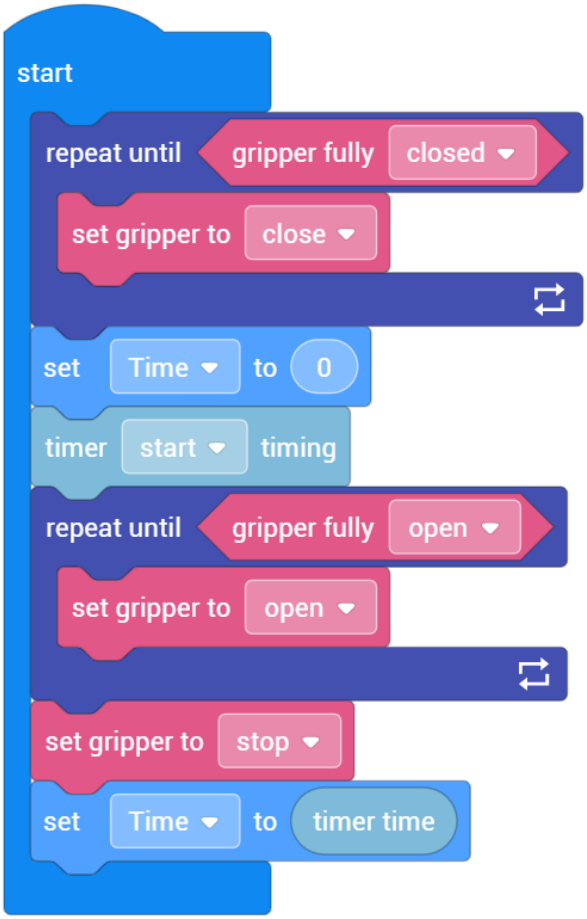
2. The gripper fully (open)

(1) Description: Return "True" when the current gripper status meets the condition; otherwise, returns "False".

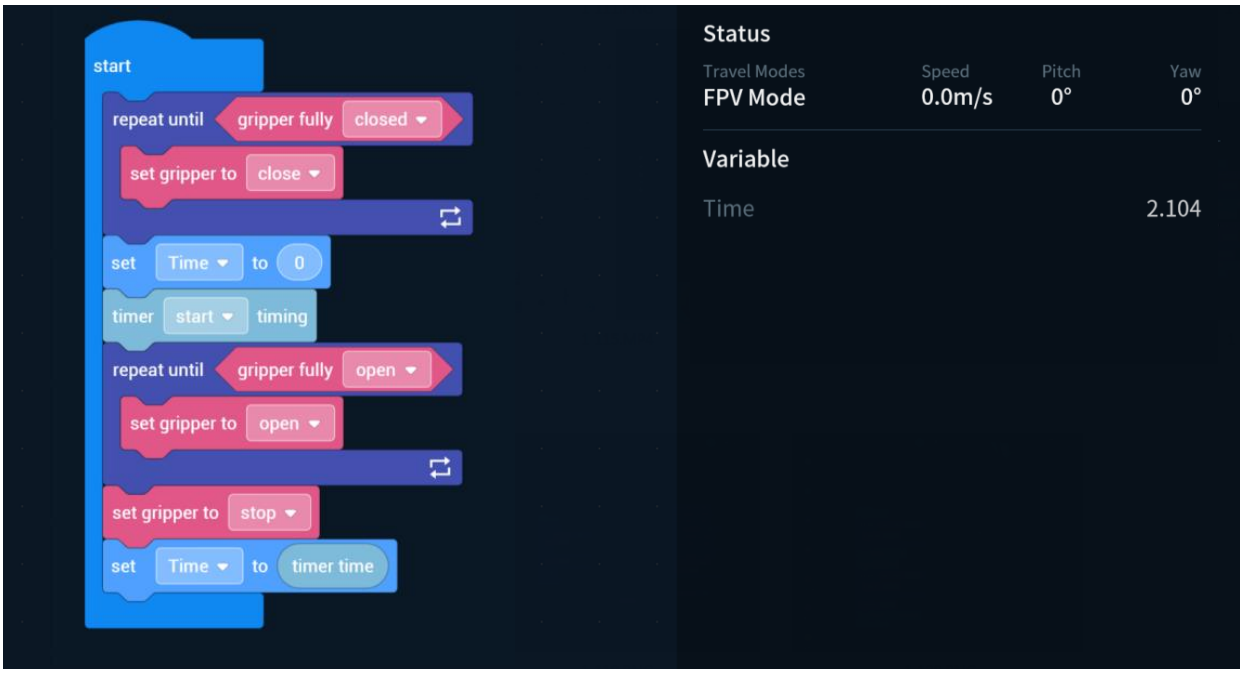
(2) Return value: Boolean

(3) Example: Gripper opening duration

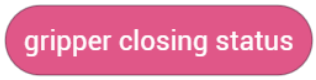
Learn the time taken for the gripper to fully open from being fully closed.



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



3. Gripper’s closing status

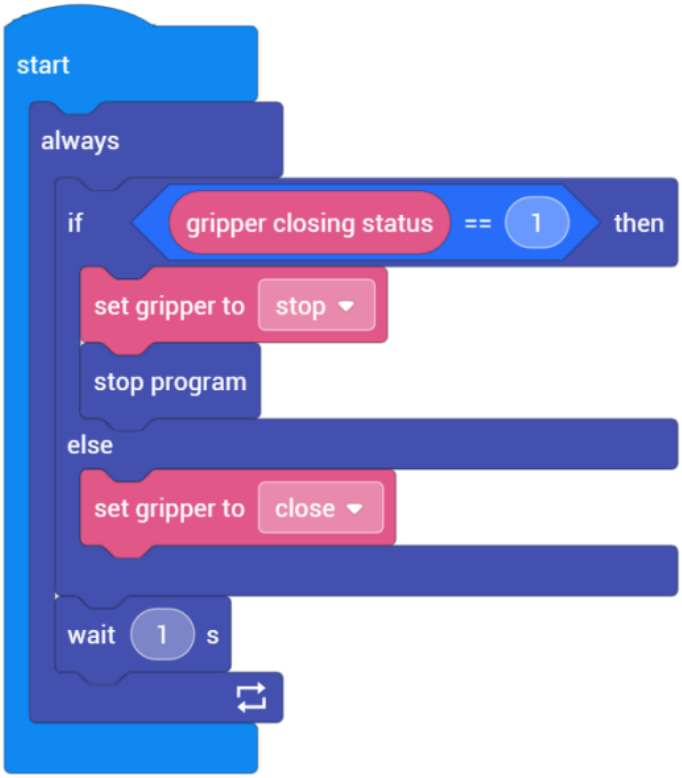


(1) Description: Learn whether the gripper is fully closed. If the return value is “1”, the gripper is fully closed; if the value is “0”, the gripper is not fully closed.

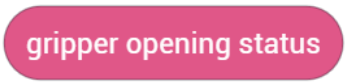
(2) Type: Information block

(3) Example: Gripper closes

Control the gripper’s movement according to its current state until it is fully closed.



4. Gripper’s opening status

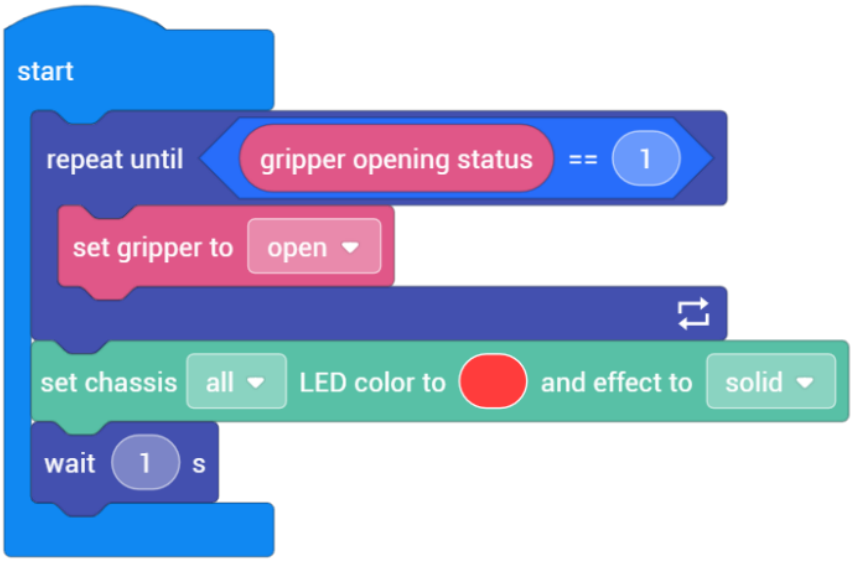


(1) Description: Learn whether the gripper is fully opened. If the return value is “1”, the gripper is fully opened; if the value is “0”, the gripper is not fully opened.

(2) Type: Information block

(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.



5. Set robotic arm to move forward for (50) millimeters

(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 block

(3) Example: Robotic arm dance

Control the robotic arm to move forward, backward, upward, and downward.

Notes:

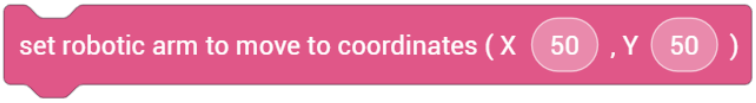
1) The robotic arm of this robot is a parallel robotic arm. It has a more complex structure which comes with a high load-bearing capacity, rigidity, and precision. It is driven by two servos installed at the bottom of the robotic arm.

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.

3) Due to the difference in assembly and calibration environment of each robot, the robot coordinate systems of different robots

may differ, and the robot has built-in limit protection in software, please use the slider to manually control the robotic arm to the target position on the operation interface, use the “robotic arm current position” block to obtain the current robotic arm coordinate information, and then debug the program.

6. Set robotic arm to move coordinates (X (50), Y (50))

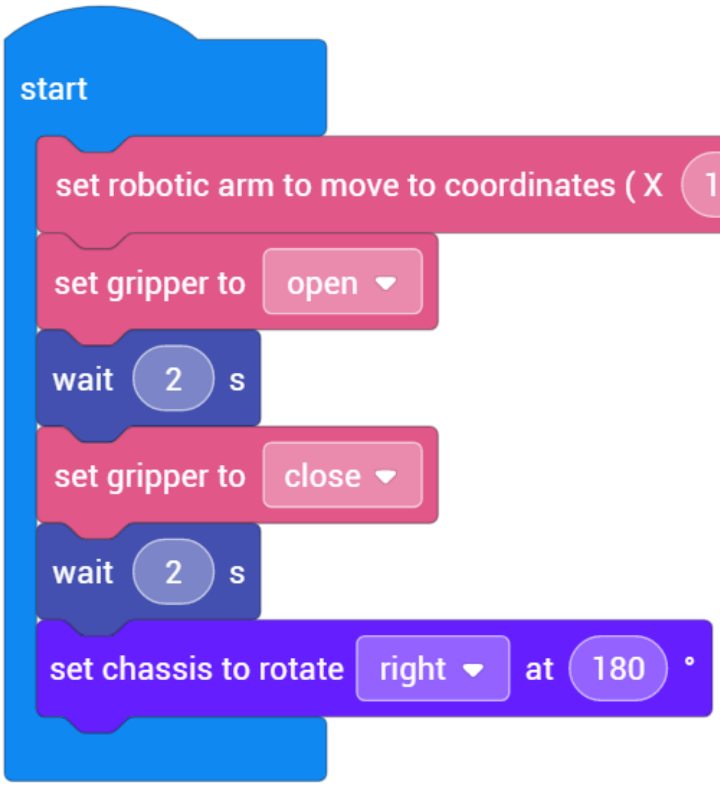


(1) Description: Set robotic arm to move to a specified position. The front and back direction along the body is X while the vertical direction along the body is Y. The unit is millimeter.

(2) Type: Execution block

(3) Example: Grasp objects

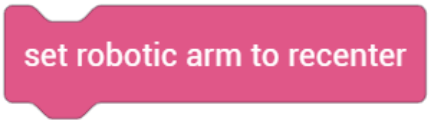
Place a water bottle in front of the robot and control the robotic arm so that it can accurately reach the target position. After that, use the gripper to grasp the bottle then finally turn the robot around.



Notes:

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.
3) Due to the difference in assembly and calibration environment of each robot, the robot coordinate systems of different robots may differ, and the robot has built-in limit protection in software, please use the slider to manually control the robotic arm to the target position on the operation interface, use the “robotic arm current position” block to obtain the current robotic arm coordinate information, and then debug the program.

7. Set robotic arm to recenter

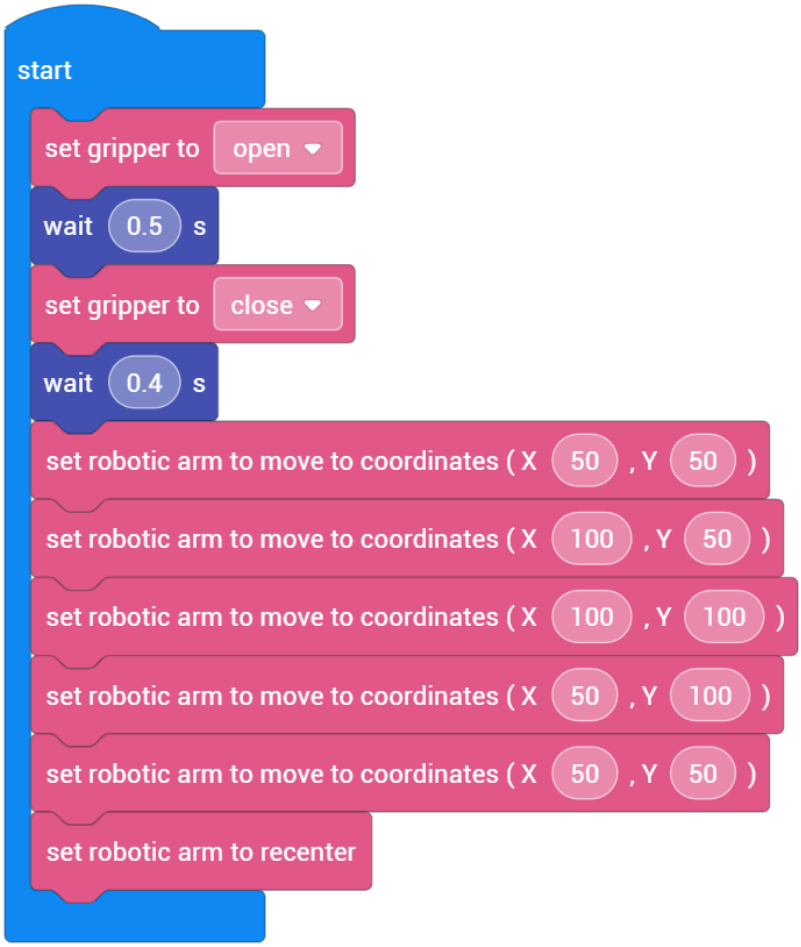


(1) Description: Control the robotic arm so that it can return to the original position, which is (0, 0) of the body coordinate system.

(2) Type: Execution block

(3) Example: Draw a square

Control the gripper to grip a writing brush and draw a square by moving the robotic arm and then recenter.

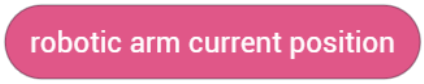


Note:

Due to the difference in assembly and calibration environment of each robot, the robot coordinate systems of different robots may differ, and the robot has built-in limit protection in software, please use the slider to manually control the robotic arm to the target

position on the operation interface, use the “robotic arm current position” block to obtain the current robotic arm coordinate information, and then debug the program.

8. Robotic arm current position



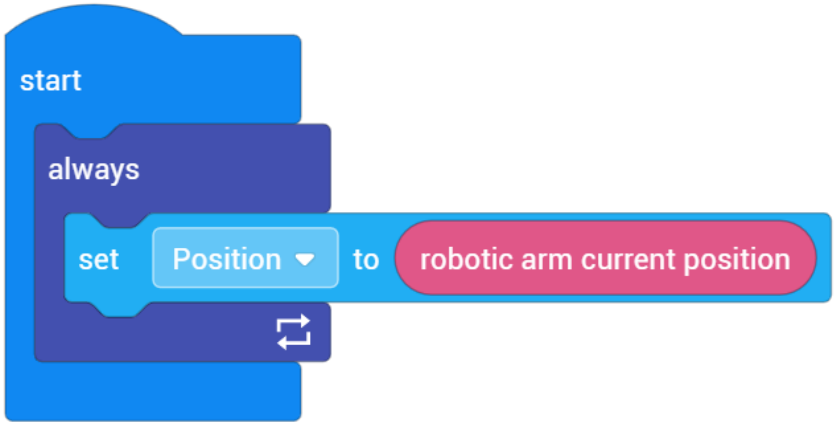
(1) Description: Obtain the robotic arm current position. The position parameter is (X (abscissa), Y (ordinate)). The unit is millimeter.

(2) Type: Execution block

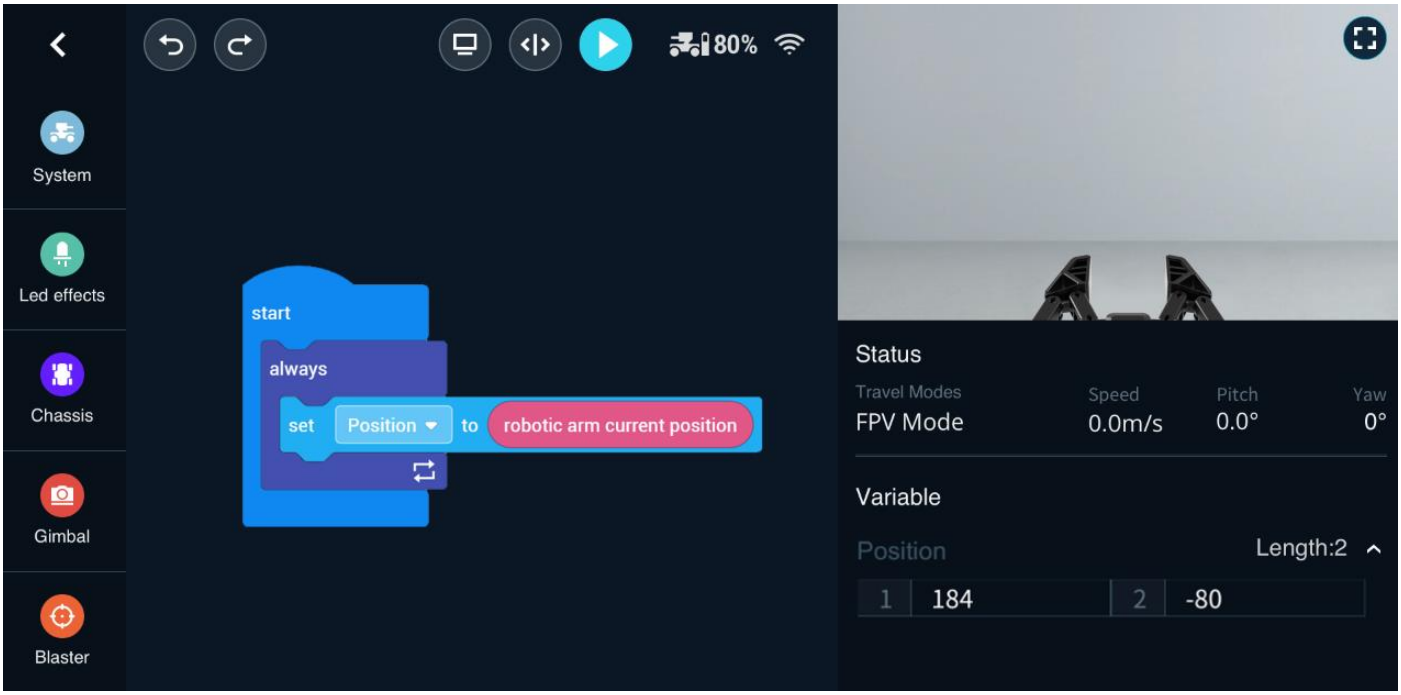
(3) Example: robotic arm current position, set the robotic arm to recenter

① The current coordinate position of the robotic arm

After moving the robotic arm to the best position for grasping the object, the current coordinate information is obtained.

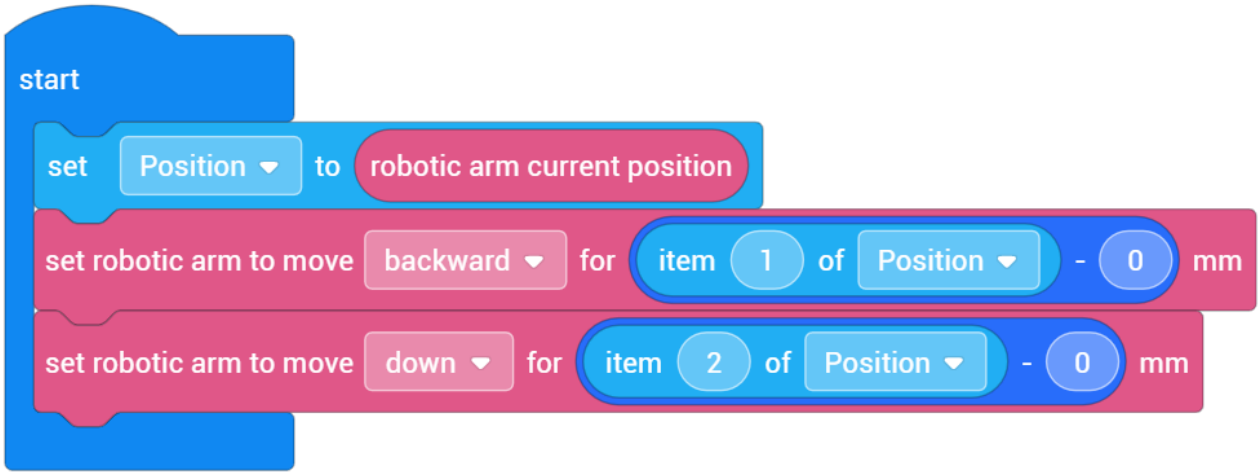


It can be observed through the FPV window:

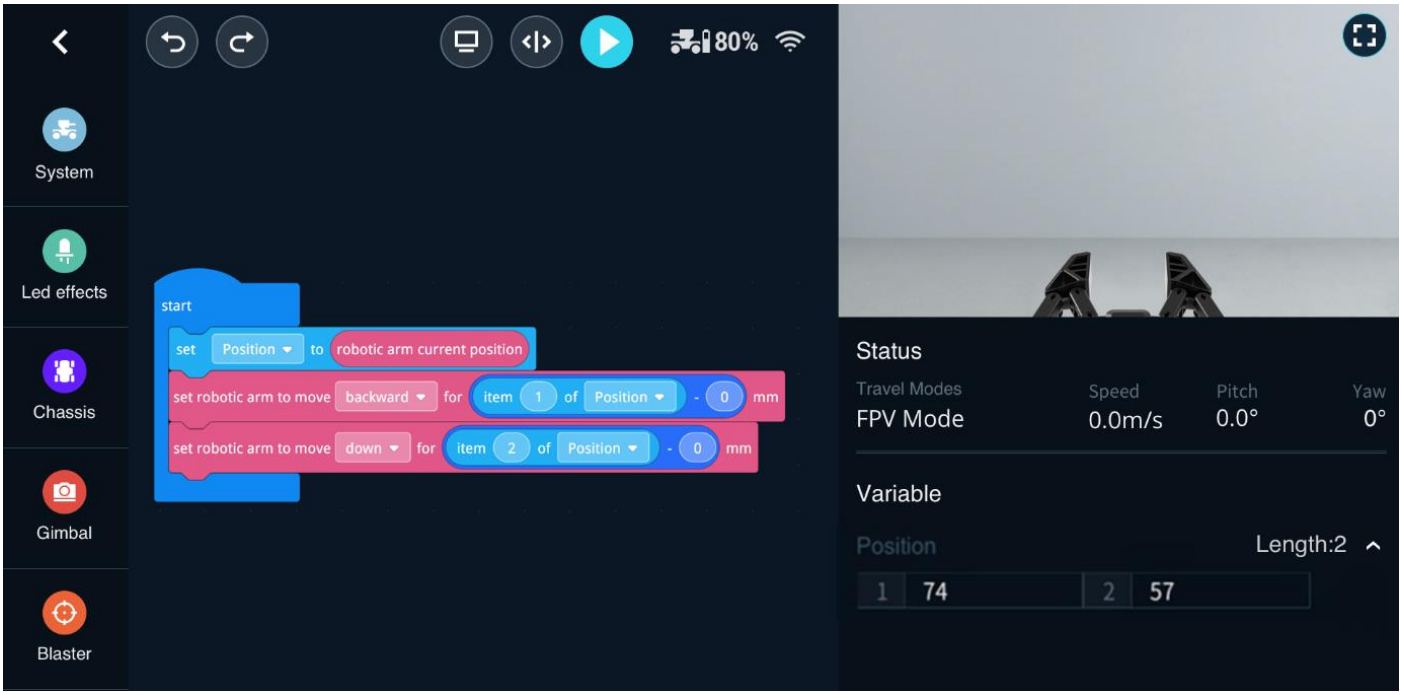


② Set robotic arm to recenter

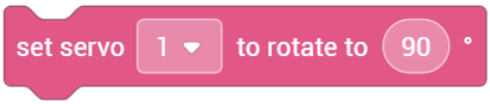
By calculating the difference between the current horizontal and vertical coordinates of the robotic arm and those of the original position, the robotic arm is controlled to return to the center.



It can be observed through the FPV window:



9. Set servo No. (1) to rotate to (90)°

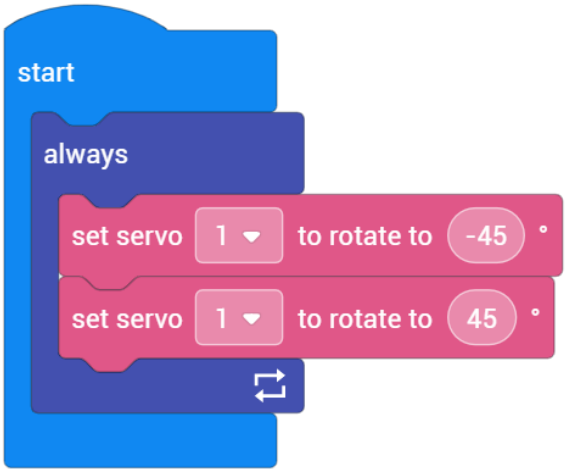


(1) Description: Set a specified servo (No.) to rotate to a set angle. If the set angle is a positive value, the servo rotates clockwise; if the set angle is a negative value, the servo rotates counterclockwise.

(2) Type: Execution block

(3) Example: Pendulum

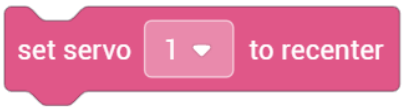
Make a pendulum using a servo, a servo gear, and adhesive tape, then control the servo so that it rotates back and forth between -45° and 45°.



Notes:

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.

10. Set servo No. (1) to recenter

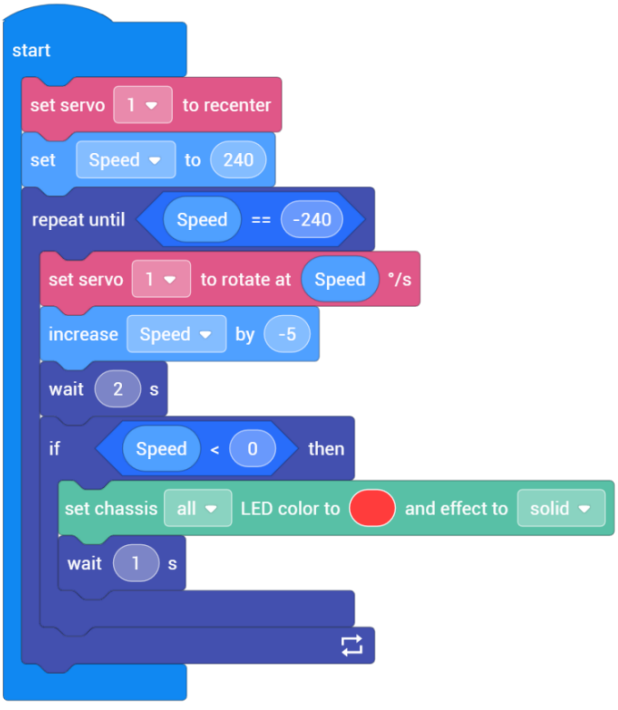


(1) Description: Set a specified servo (No.) to return to its original position.

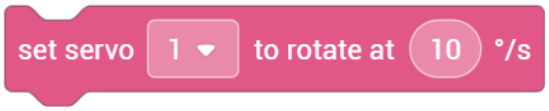
(2) Type: Execution block

(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.



11. Set servo No. (1) to rotate at (10)°/s

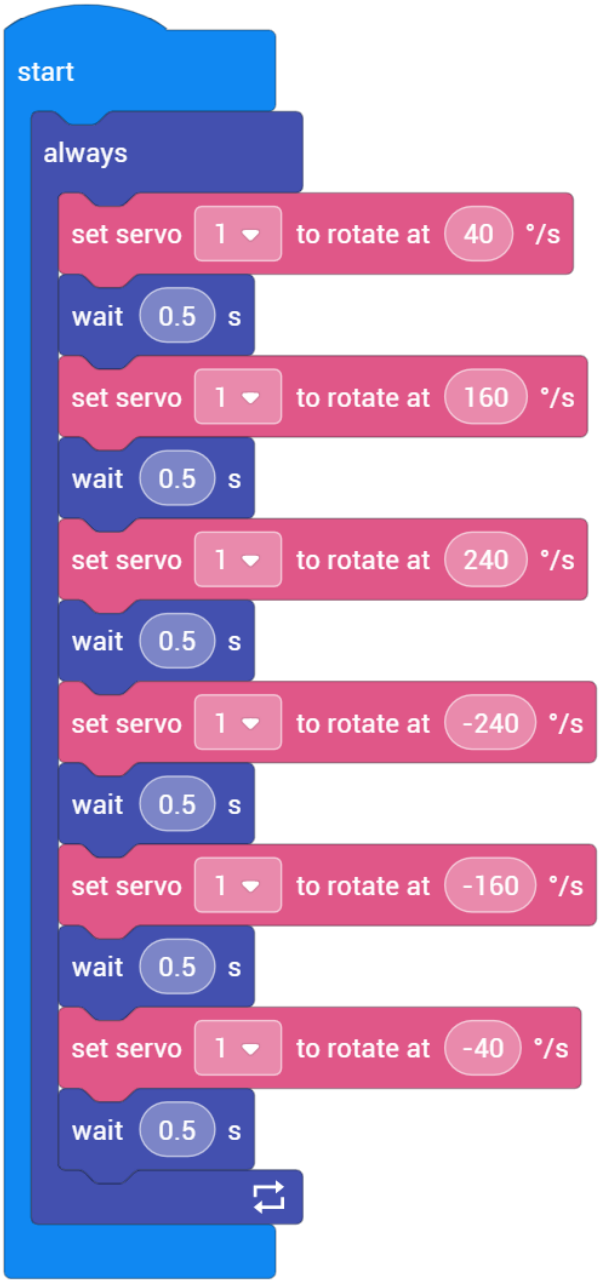


(1) Description: Set a specified servo (No.) to rotate at a set speed. If the speed is a positive value, the servo rotates clockwise; if the speed is a negative value, the servo rotates counterclockwise.

(2) Type: Execution block

(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.

12. Servo No. (1) angle

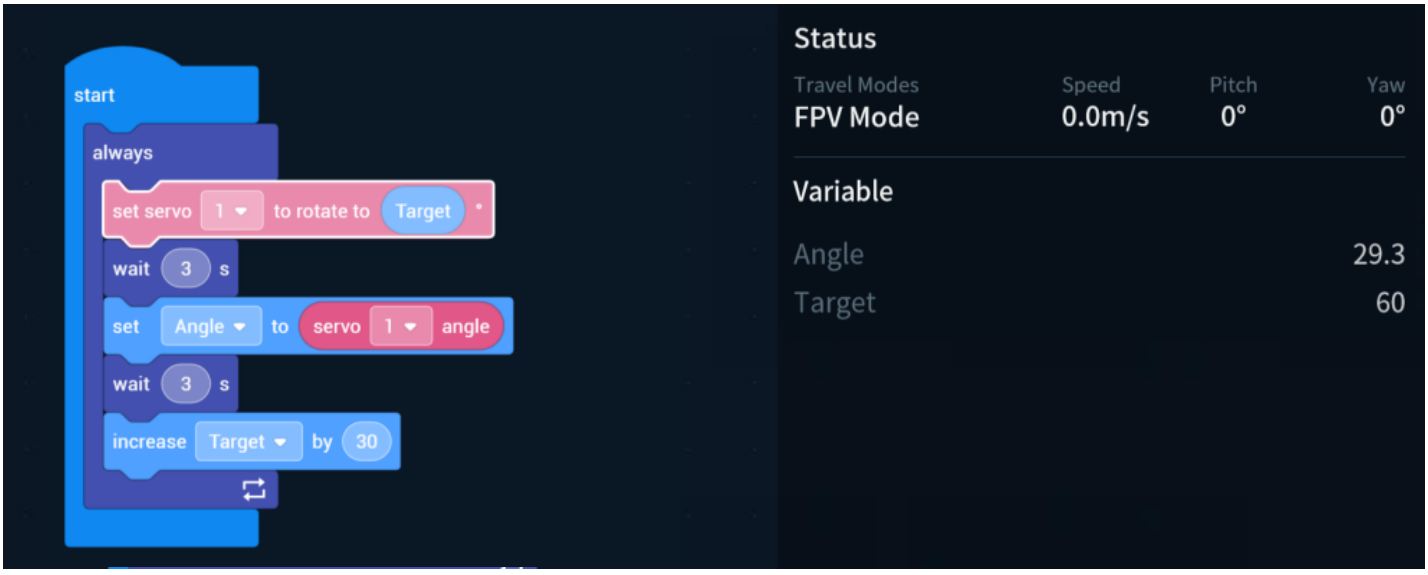
(1) Description: Obtain the angle of a specified servo (No.).

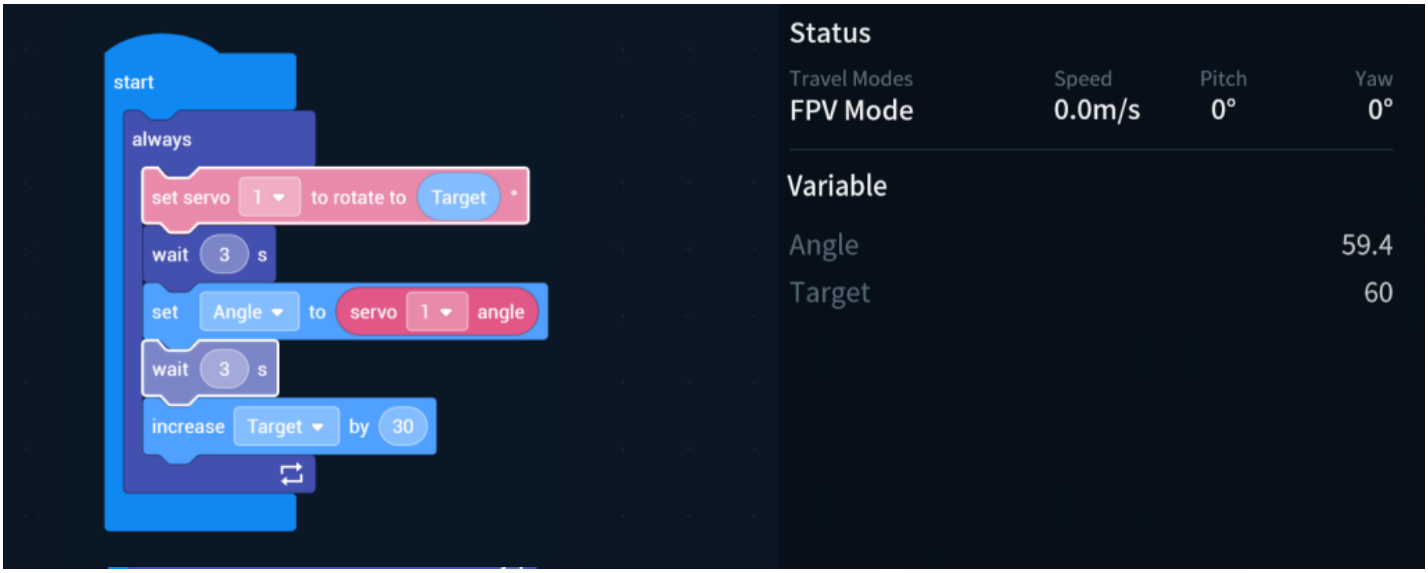
(2) Type: Information block

(3) Example: Servo angle

Obtain the changes of the servo angle as the servo rotates continuously.

It can be observed through the FPV window.





Smart

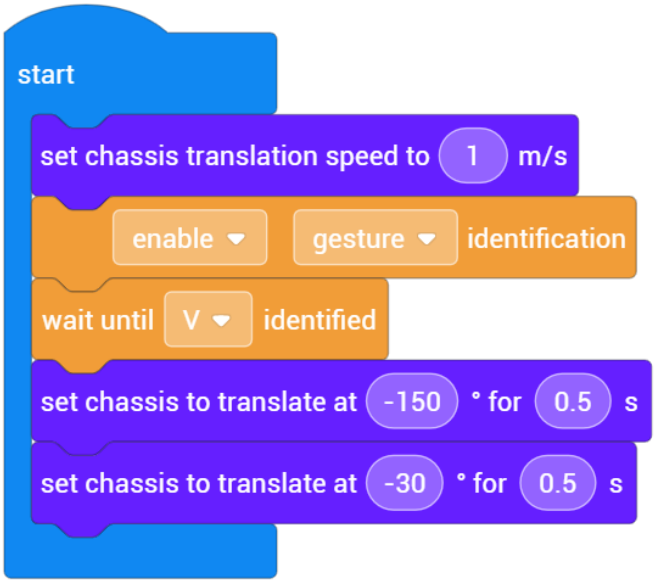
1. (Enable) (Vision Marker) identification

(1) Description: Enable or disables the RoboMaster EP CORE's visual identification functions for vision marker, gesture, person, and other RoboMaster robot

(2) Type: Settings block

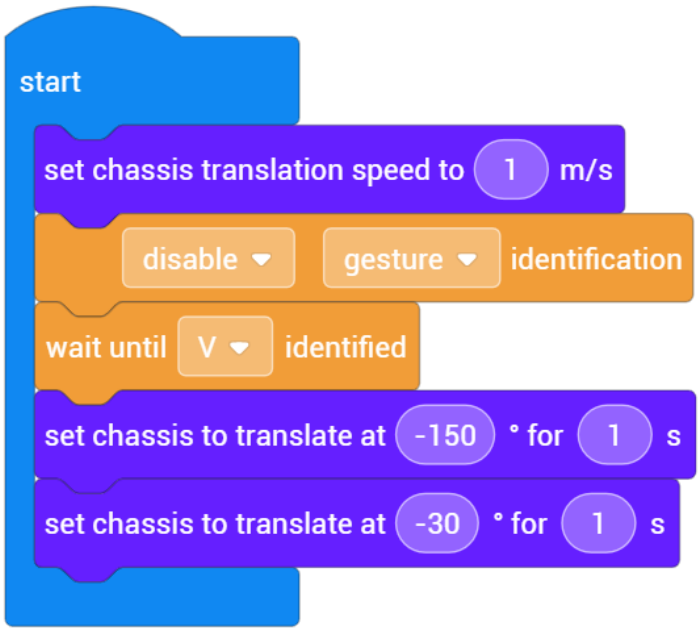
(3) Example: Imitate V-signal hand gesture

Set the chassis to translate in a V shape when the RoboMaster EP CORE identifies the V-sign hand gesture.

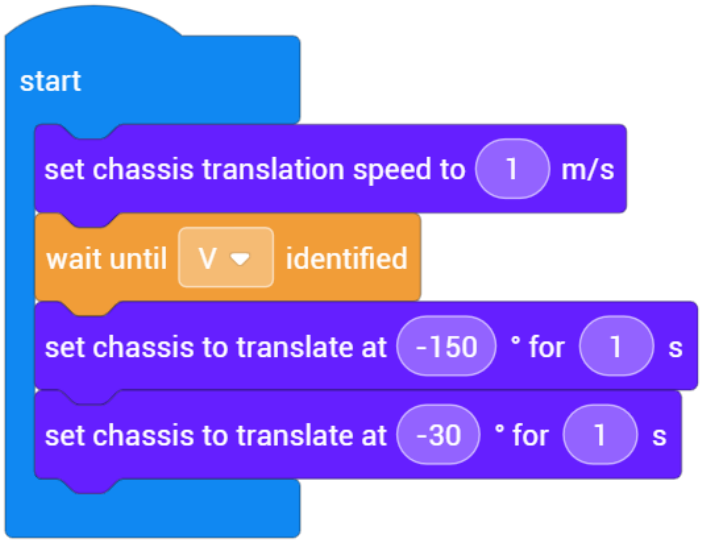


Note:

The RoboMaster EP CORE's smart identification function is disabled by default. If the identification function is not enabled for the robot, it will respond to identifiable objects.



or



In the above two example programs, the RoboMaster EP CORE will be unable to identify V and the chassis will not move.

Python API:

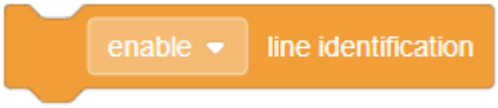
Function: vision_ctrl.enable_detection(function_enum)

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

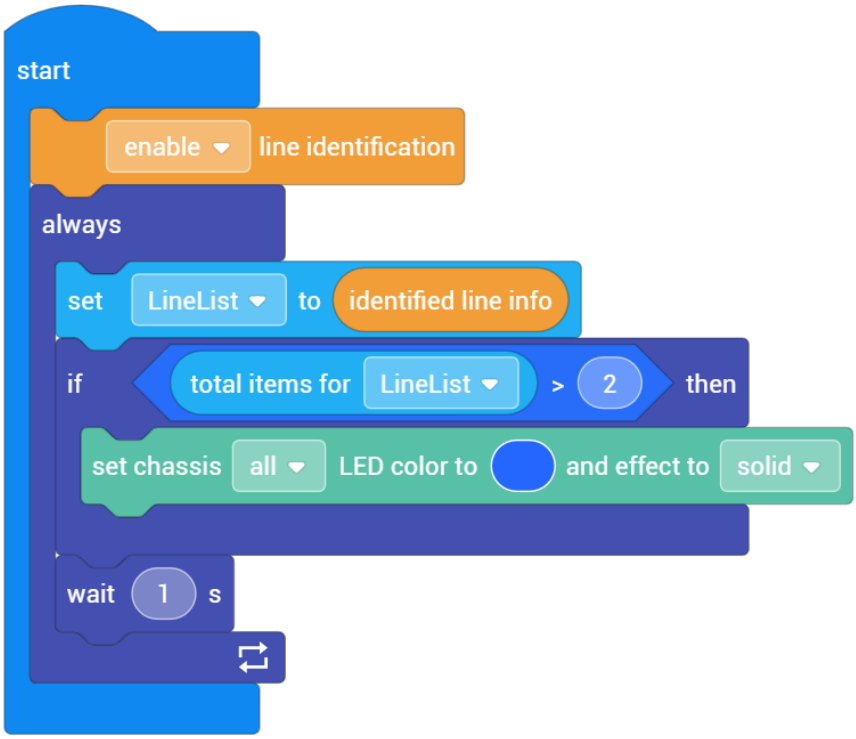


(1) Description: Enable/disable line identification

(2) Type: Settings block

(3) Example: Identify blue lines

When the robot identifies blue lines, all chassis LEDs will be solid blue.



Notes:

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

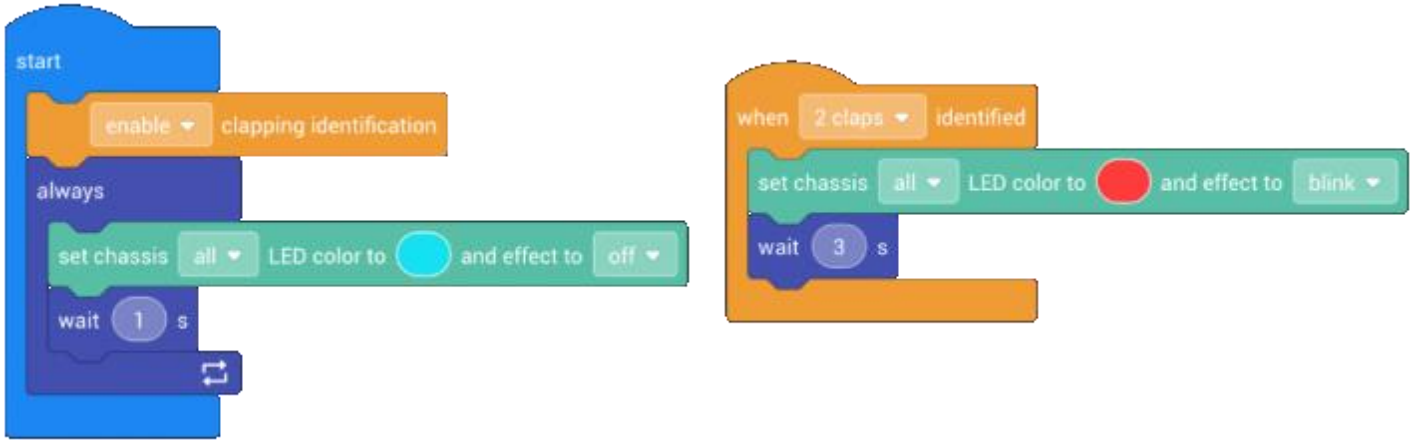
3. (Enable) Clapping identification

(1) Description: Enable/disable clapping identification

(2) Type: Settings block

(3) Example: Wake up the robot by clapping

Initially, the LED chassis indicators are turned off. When the robot identifies a double clap, all red LED chassis indicators will blink.



Note:

Clapping identification is disabled by default.

If clapping identification is not enabled first, RoboMaster EP CORE will not 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

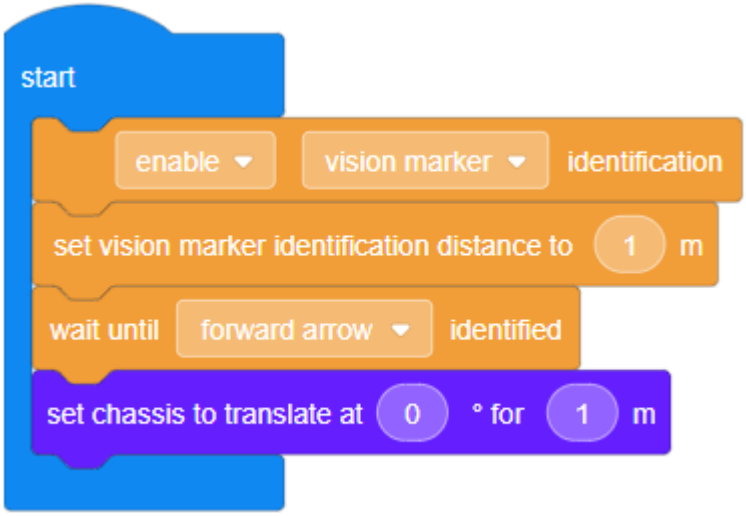
4. Set vision marker identification distance to (1) m

(1) Description: Set 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.



Notes:

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]

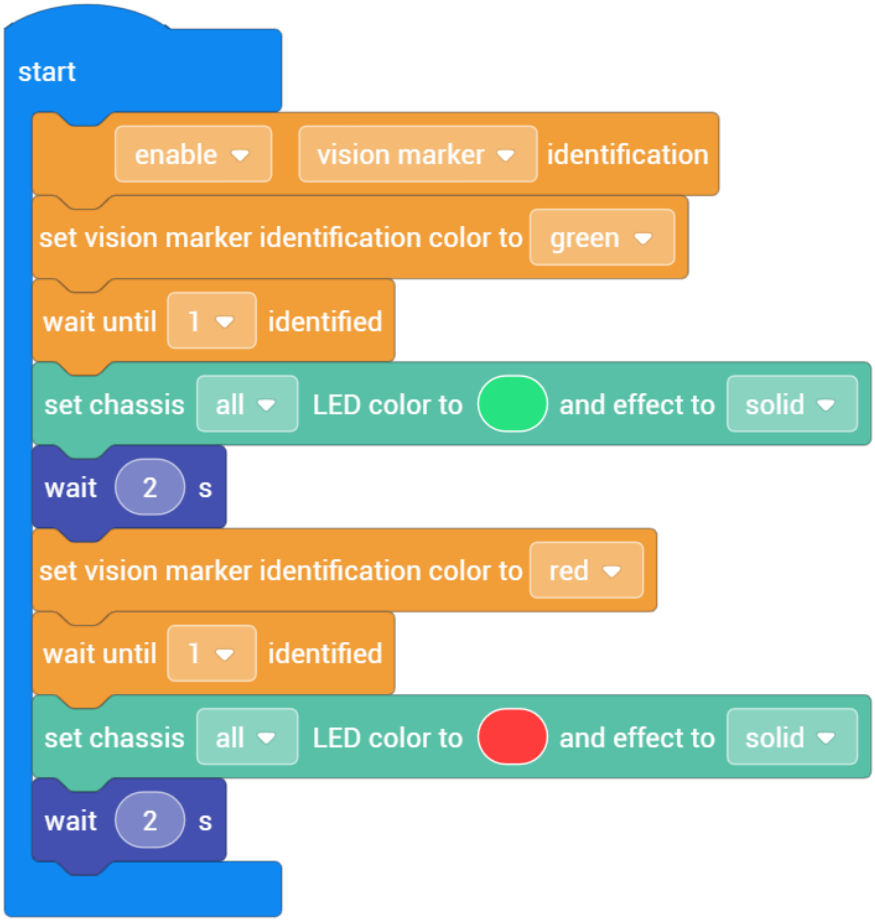
5. Set vision marker identification color to (red)

(1) Description: Set vision marker colors that the robot can identify.

(2) Type: Settings block

(3) Example: Switch different vision marker 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 vision marker color is red.

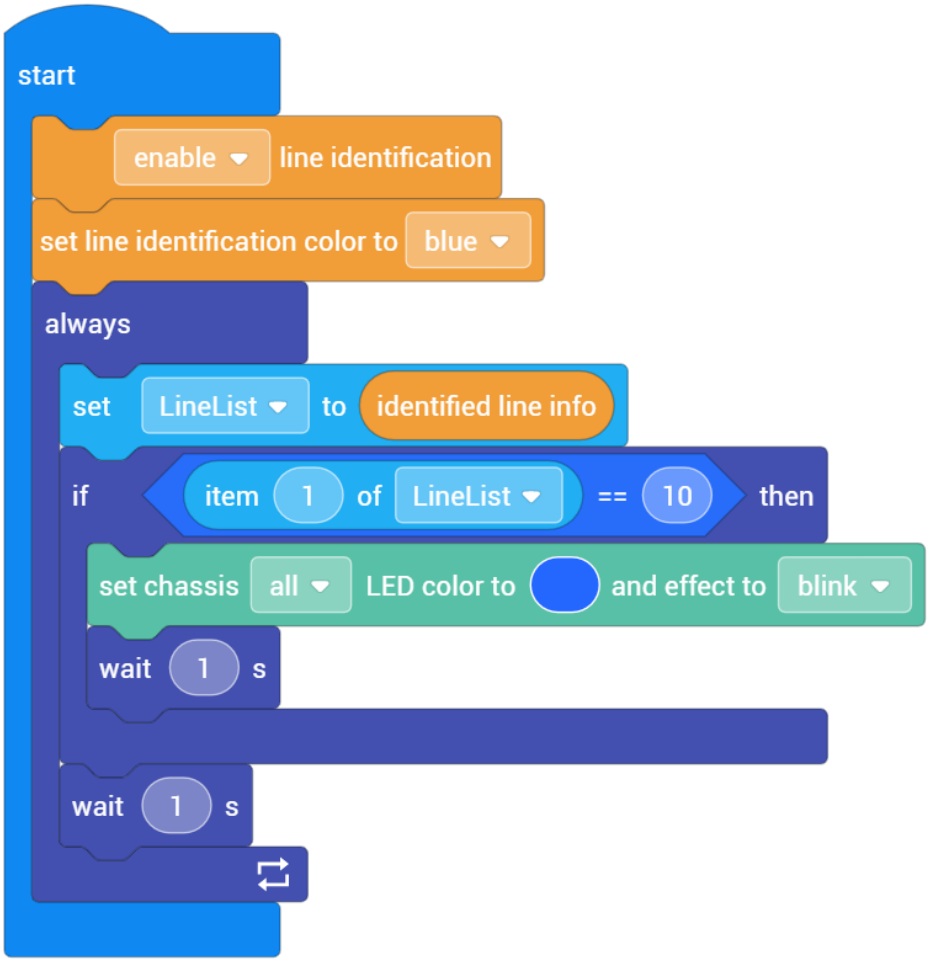
6. Set line identification color to (blue)

(1) Description: Set 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 before the camera, the robot's chassis LED will blink blue for 3 seconds.



Note:

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

Python API:

Function: vision_ctrl.line_follow_color_set(color_enum)

Parameters:

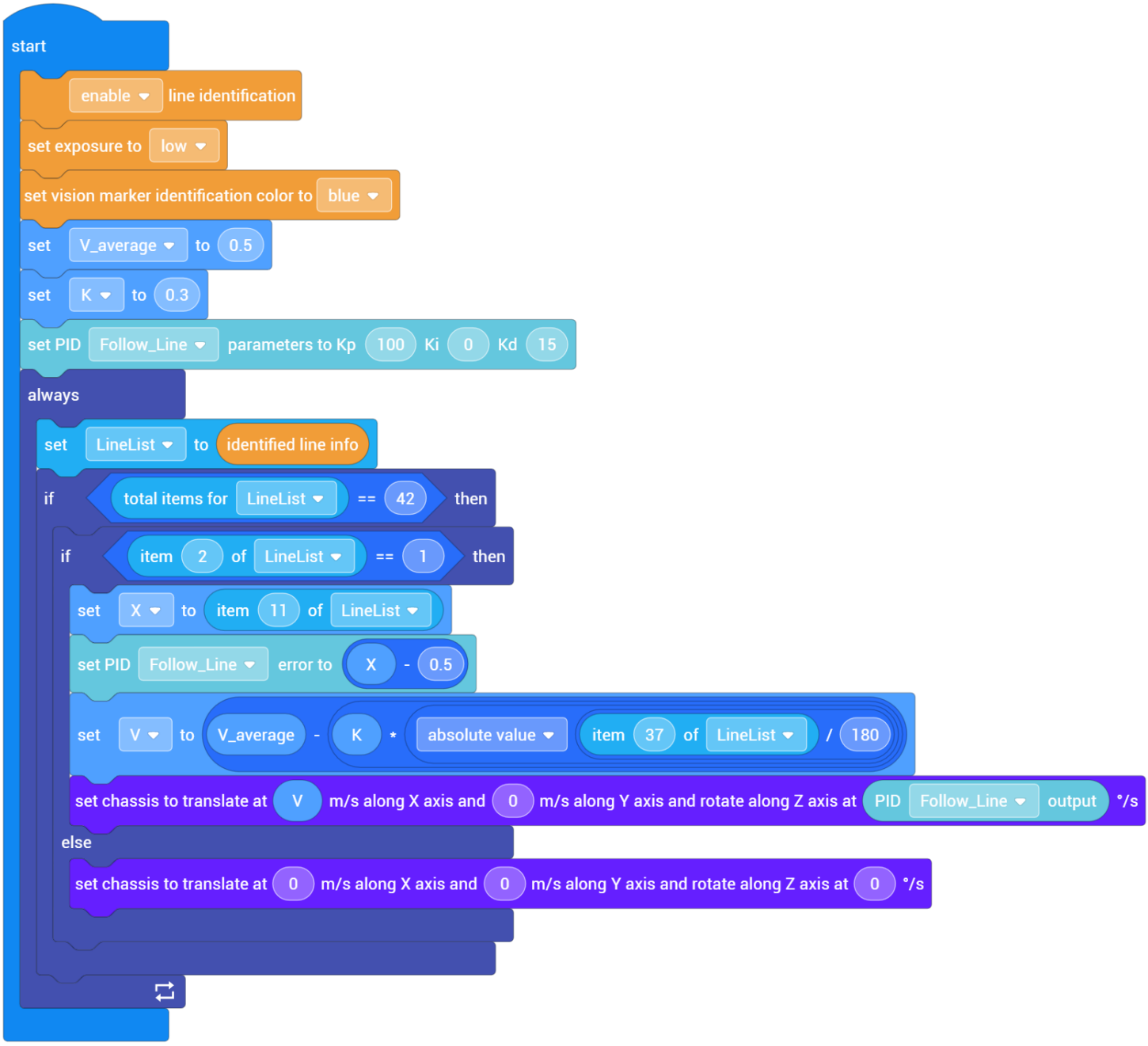
- color_enum(enum):
  - rm_define.line_follow_color_blue
  - rm_define.line_follow_color_red
  - rm_define.line_follow_color_green

7. Set exposure to (high)

(1) Description: Reduce 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: EP CORE's line follow



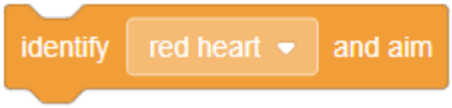
Python API:

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

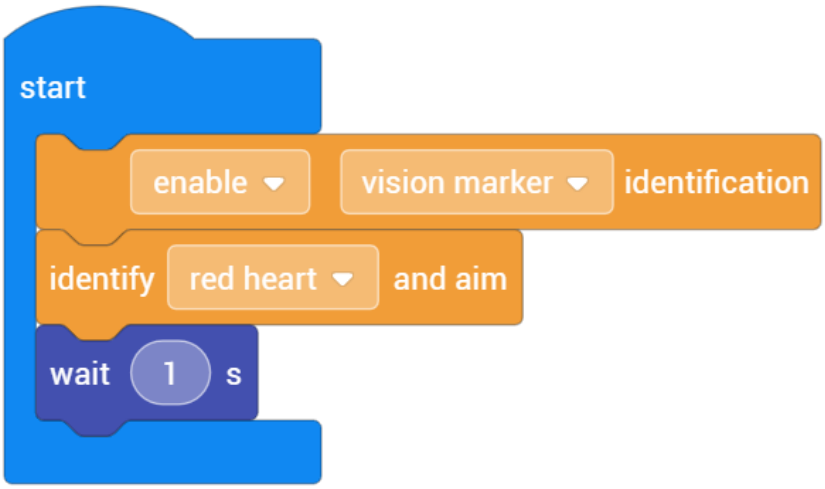
8. Identify (red heart) and aim



(1) Description: Set 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



Notes:

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 5 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

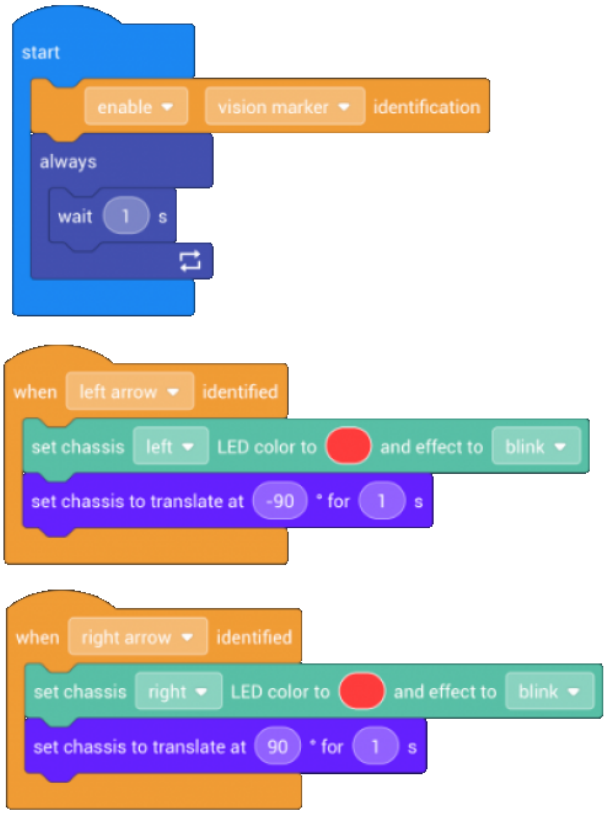
(1) Description: Set the corresponding block to run its internal program when specified information is identified

(2) Type: Event block

(3) Examples: Indicate turn, Control attitude

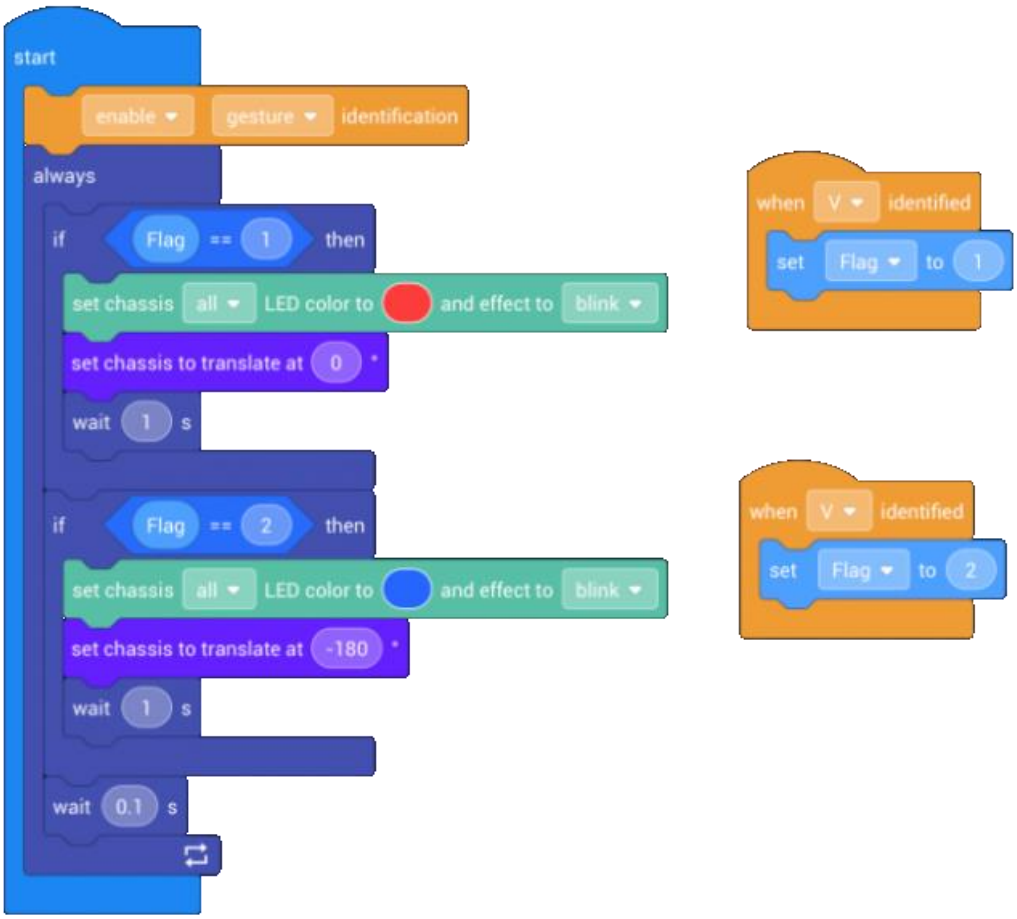
① Indicate turn

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



Notes:

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 robotic arm 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

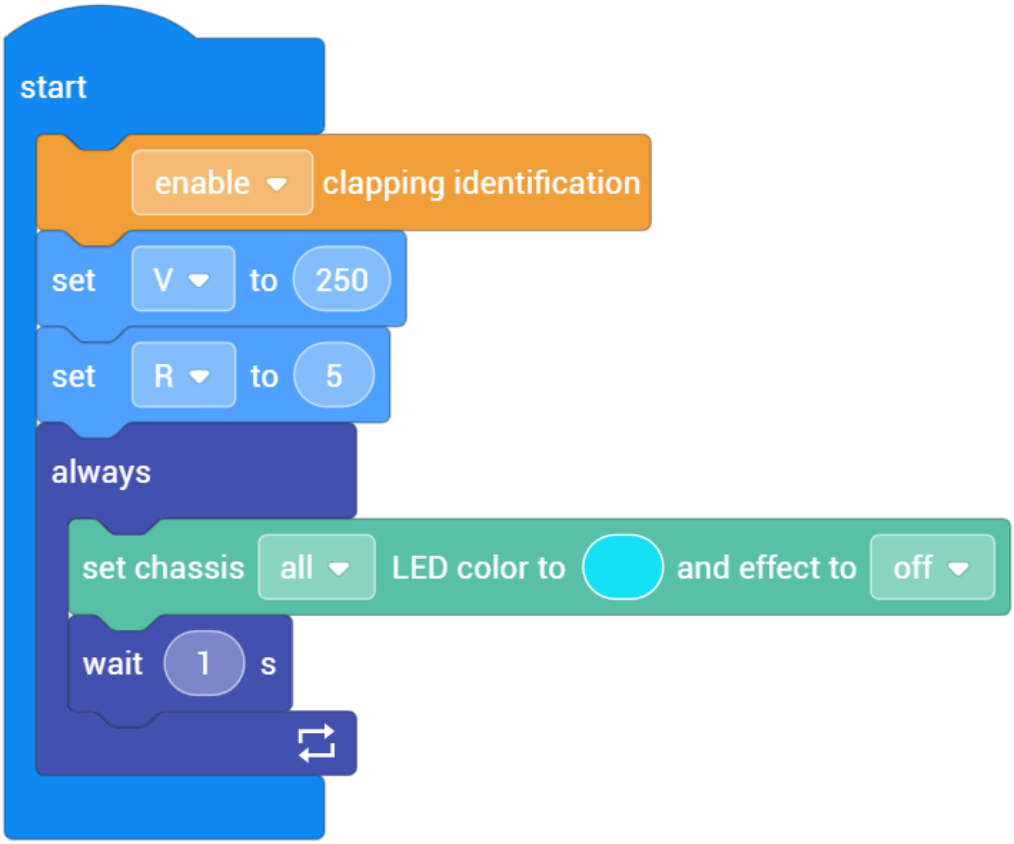
10. When (2) claps identified

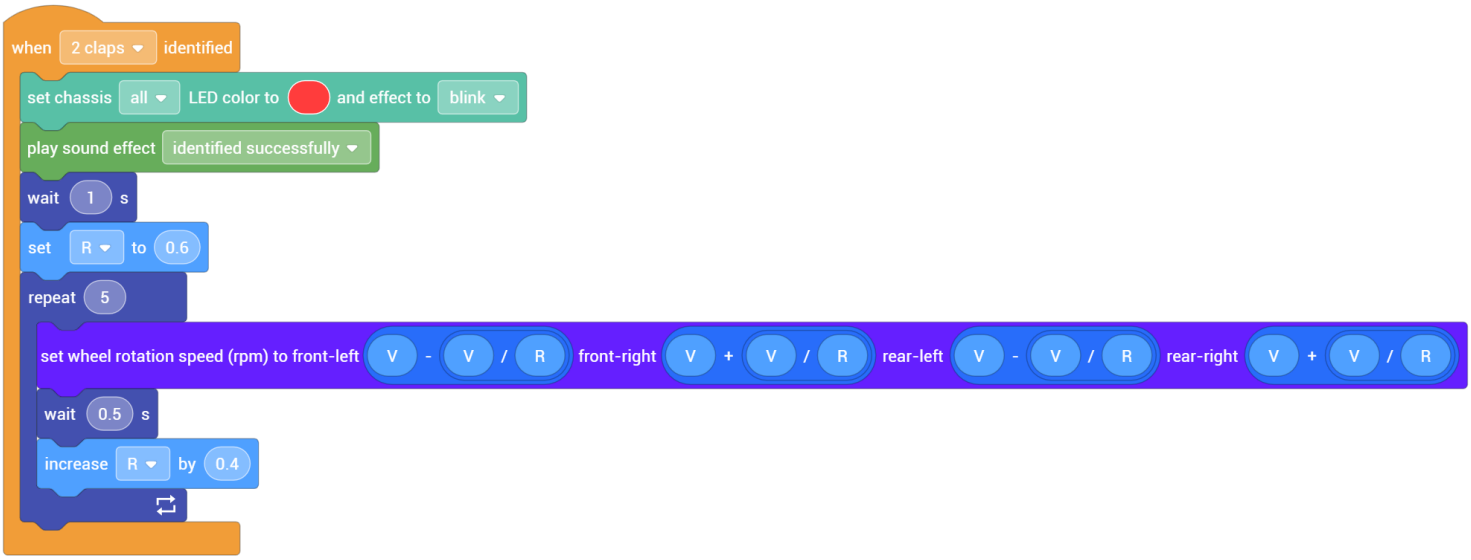
(1) Description: Set the block to run its internal program when a specific clapping pattern is identified

(2) Type: Event block

(3) Example: Translate in a spiral pattern

Initially, all chassis 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

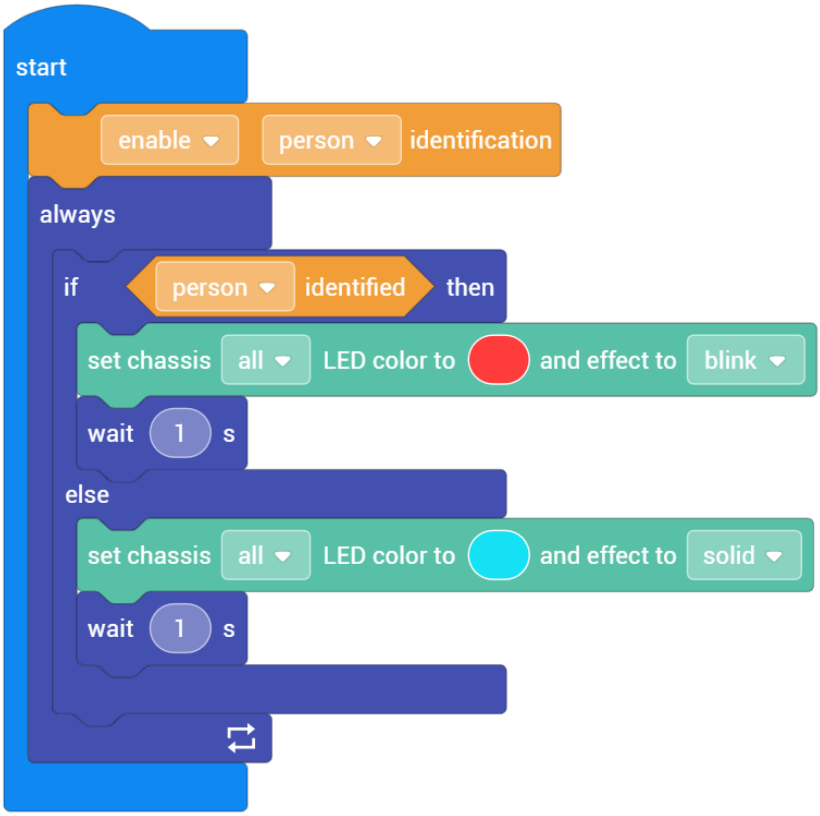
11. (Person) identified

(1) Description: Set 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) Return value: Boolean

(3) Example: Identify people

When the robot identifies a person, all red LED chassis indicators will blink; otherwise, all LED chassis indicators will remain in the default color.



Note:

Try to use this block with conditional statements whenever possible.

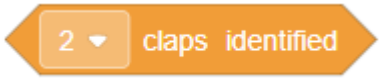
Python API:

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

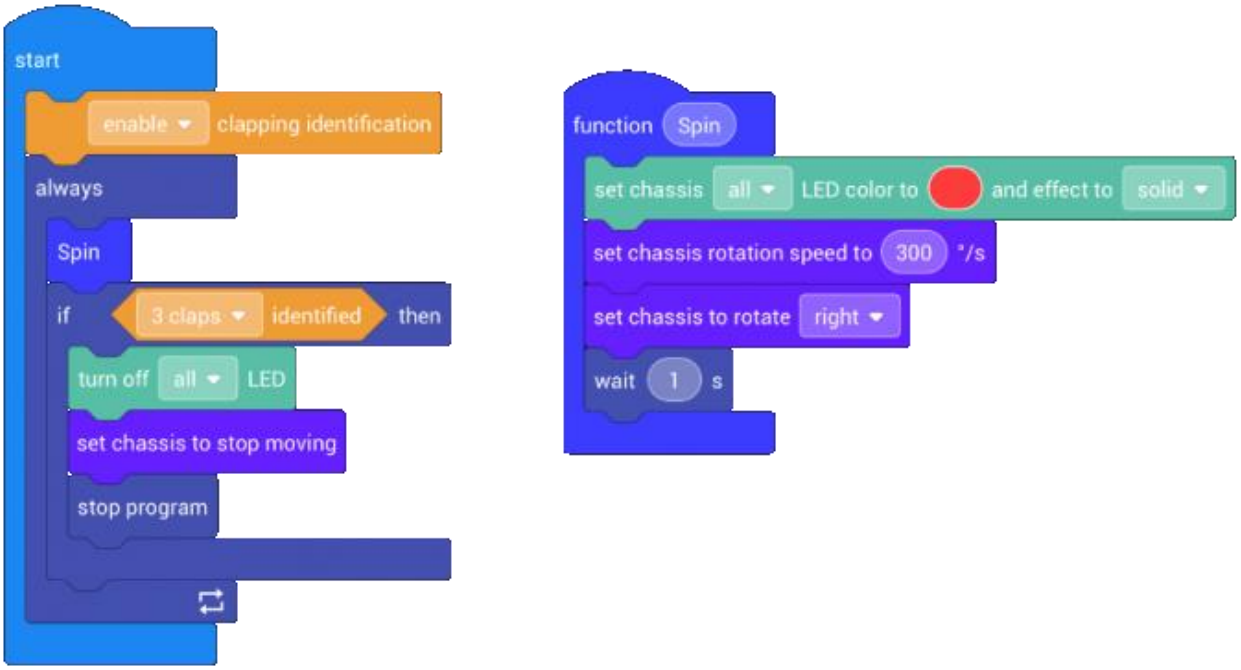


(1) Description: Set the robot to return the condition “True” when a specific clapping pattern is identified; otherwise, the condition “False” is returned.

(2) Return value: Boolean

(3) Example: Stop motion with clapping

This sets the robot to turn right and stop all motion when it identifies 3 claps.



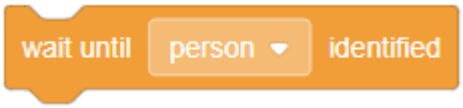
Python API:

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

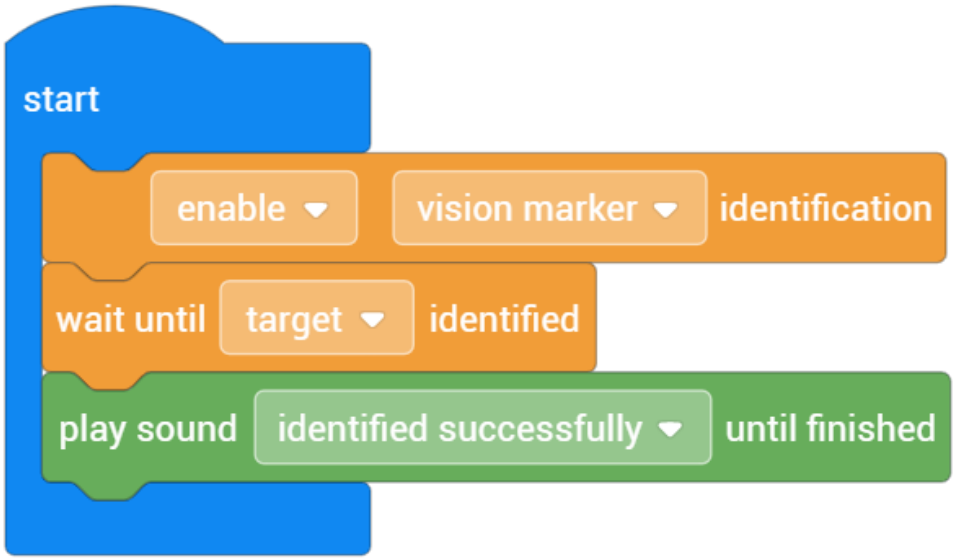


(1) Description: When specific information (such as an object, vision marker, gesture, and so on) is identified, the system will continue executing commands; otherwise, it will continue to wait.

(2) Type: Execution block, Blocking block

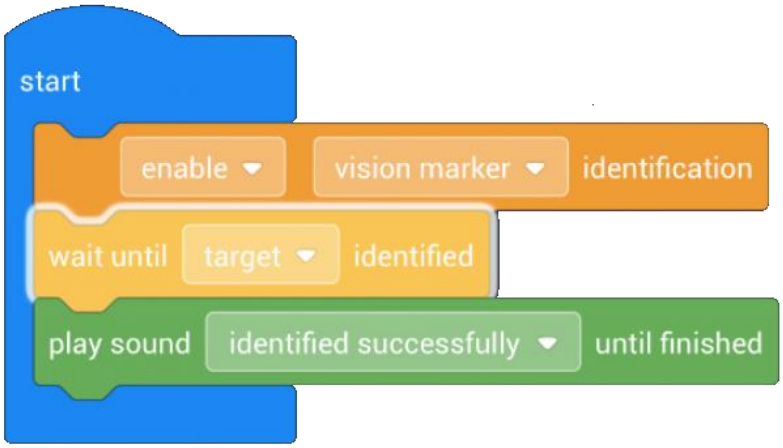
(3) Example: Practice shooting

Play sound effect when target is identified successfully



Note:

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



Python API:

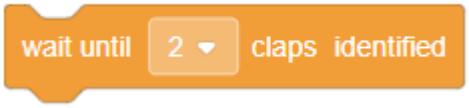
Function: vision_ctrl.cond_wait(condition_enum)

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

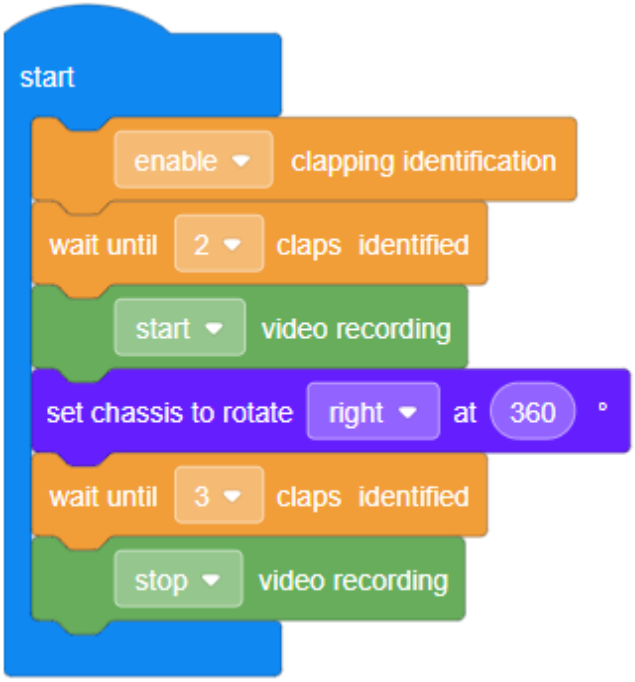


(1) Description: Set the system to continue to execute commands when a specific clapping pattern is identified; otherwise, it will continue to wait.

(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

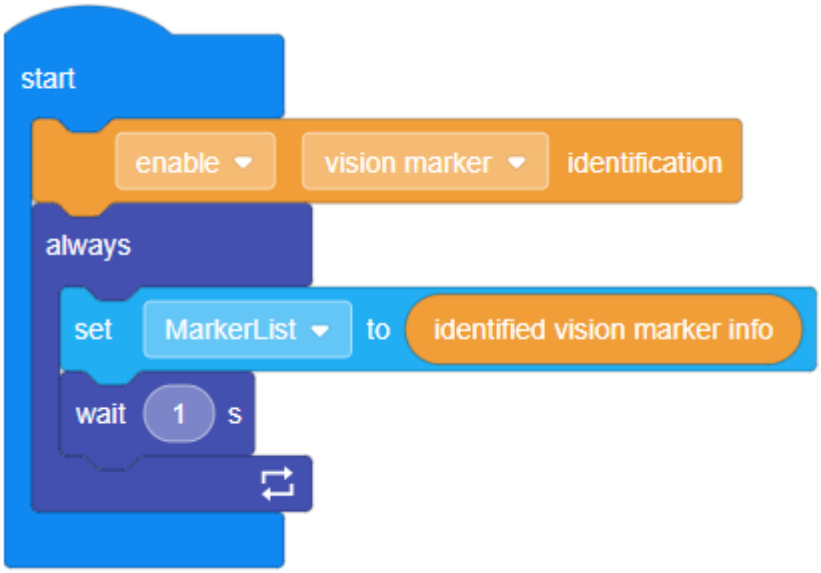
15. Identified vision marker info

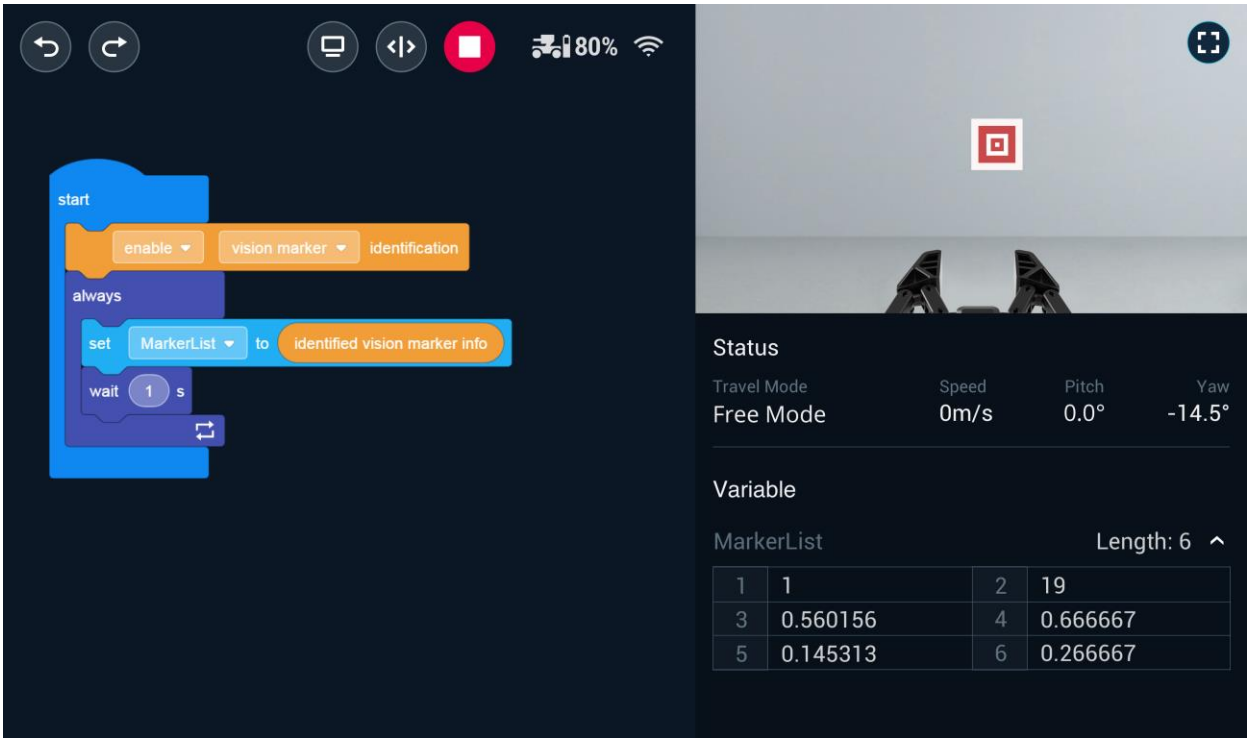
(1) Description: Obtain information for an identified Vision Marker, returned as N, ID, X, Y, W, and H parameters.

(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.





Notes:

1) The vision marker format is as follows:

The first item is N, 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, followed by gesture ID, X-axis of the center point, Y-axis of the center point, W-width and H-height of the second gesture, the third, ...

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

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)

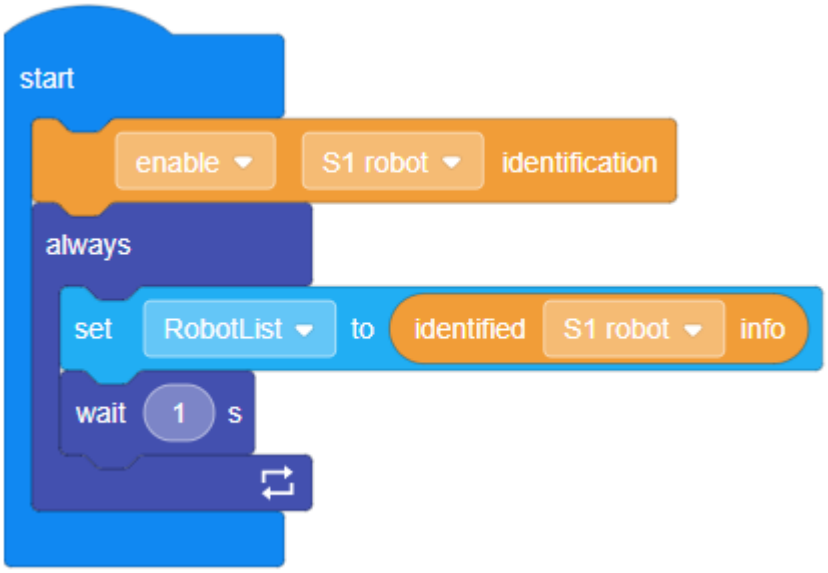
16. Identified (person) info

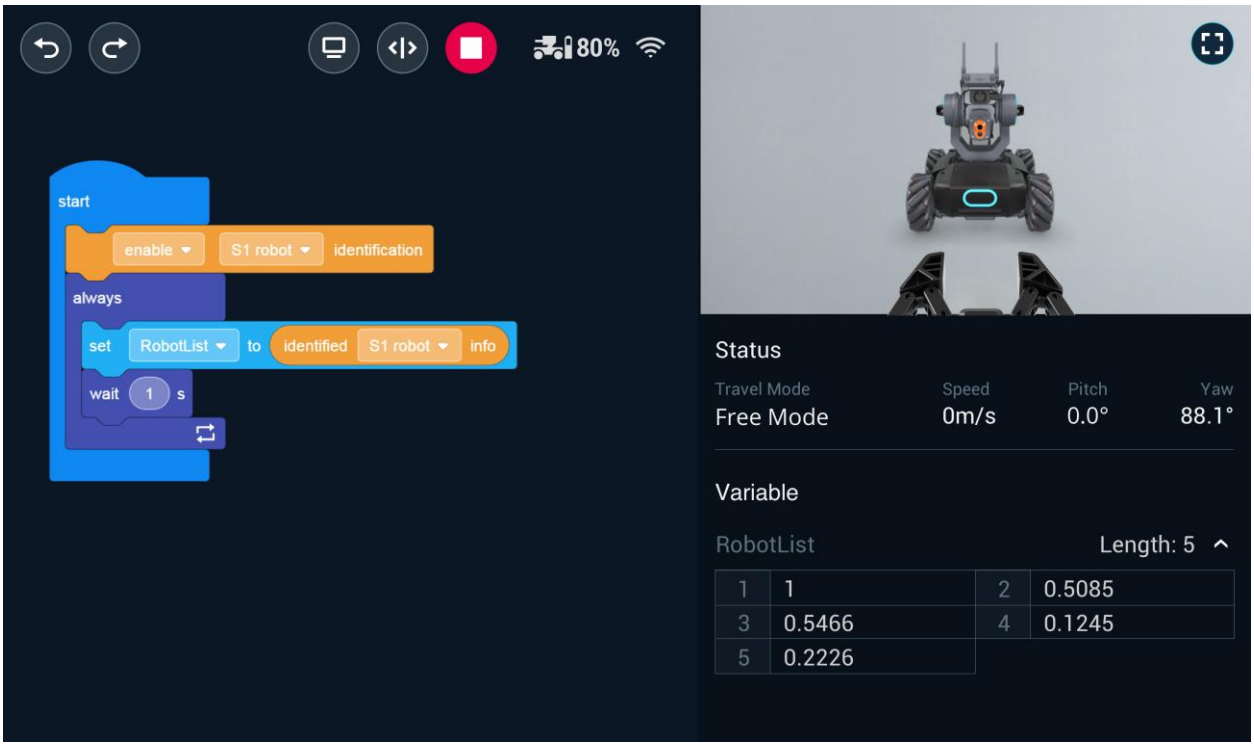
(1) Description: Obtain the information of an identified person or such robot in terms of the parameters such as N, X, Y, W, and H

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

(3) Example: Identify another EP CORE robot

When another EP CORE robot appears in the robot's FOV, 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.

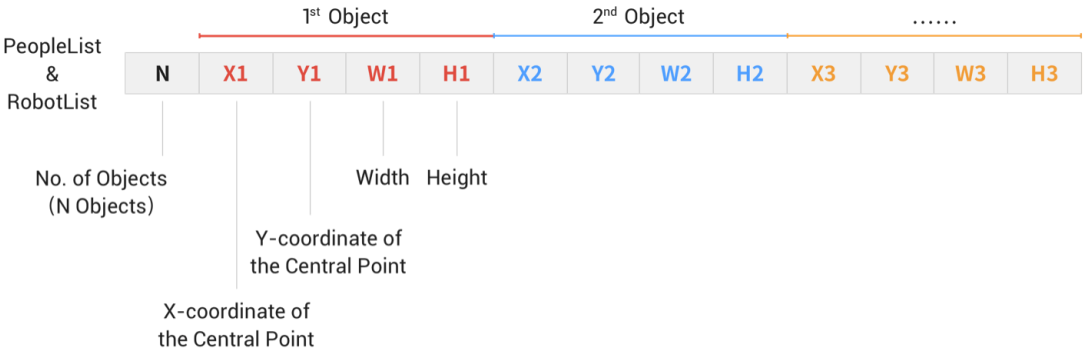




Notes:

The format of object information is as follows:

The first item is N, the number of objects the robot has identified, followed by a group of 4 data: the abscissa X, ordinate Y, width W and height H of the position of the center point of the first object in the robot's FOV; such data of the second gesture, the third, ...



Python API:

Function: vision_ctrl.get_people_detection_info( )

vision_ctrl.get_car_detection_info( )

Return value:

- detection_info(list)

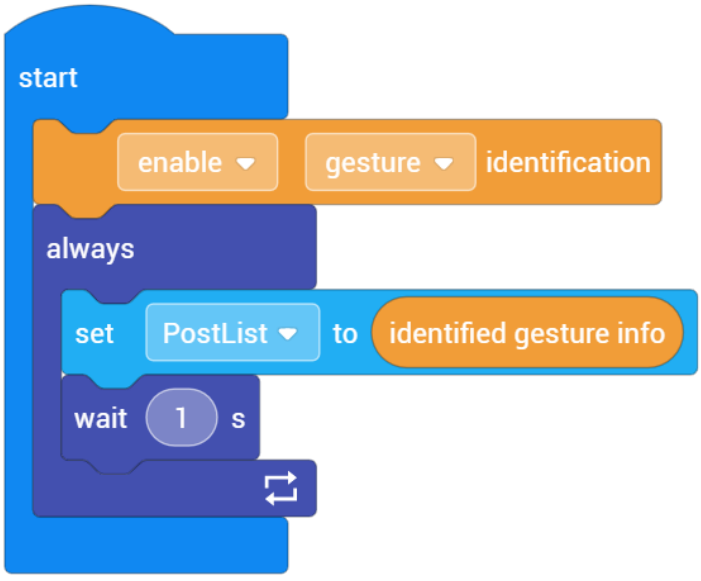
17. Identified gesture info

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

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

(3) Example: Gesture information

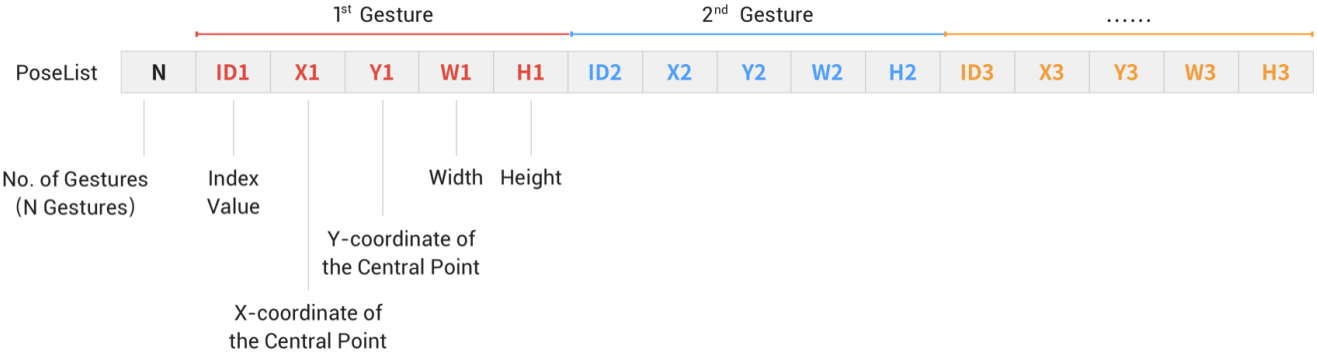
You can observe changes in gesture-related data using the FPV window.



Notes:

1) The format for gesture information is as follows:

The first item is N, 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, followed by gesture ID, X-axis of the center point, Y-axis of the center point, W-width and H-height of the second gesture, the third, ...



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)

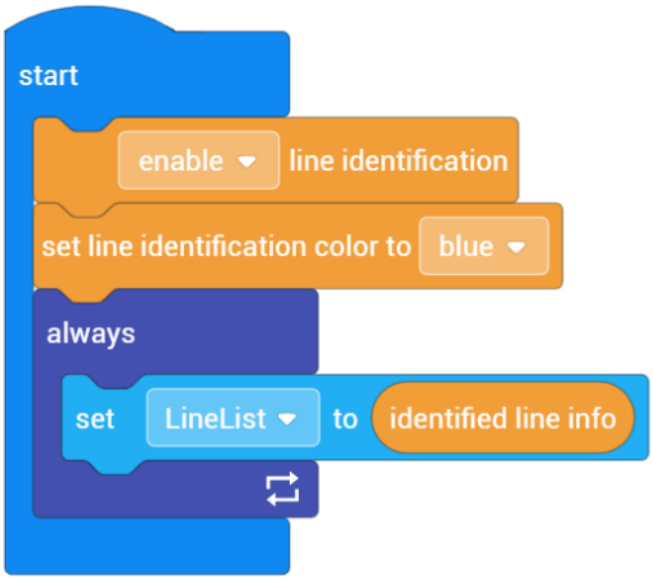
18. Identified line info

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

(2) Type: Information block (list-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
FPV Mode	2.4m/s	-20.0°	0.0°

Variable

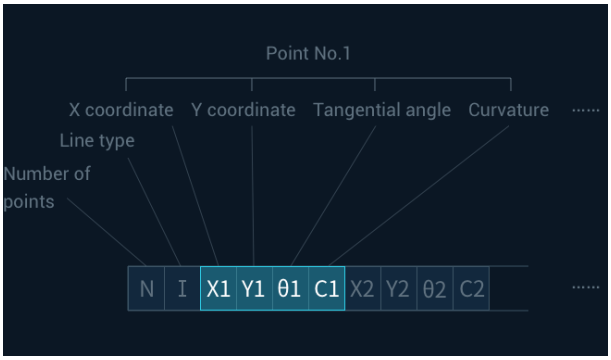
LineList Length: 42

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

Notes:

1) The single line information format that is identifiable by the robot is as follows:

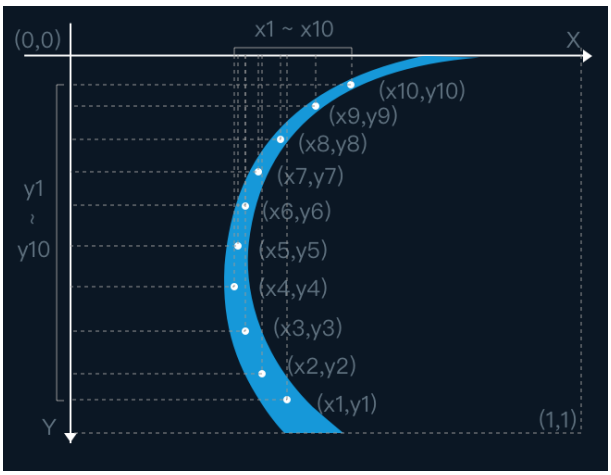
The first item “N” is the number of points identified on the line, the second item ‘Info’ is the line type, and the subsequent items are grouped into 4 data groups: they are the (Coordinate X, ordinate Y, actual tangent angle θ, curvature C) of ten points that are equidistant from each other on the line (from the nearest to the farthest). There are altogether 42 sets of data.

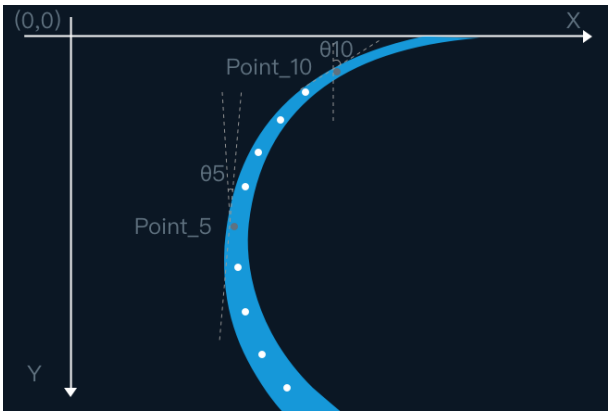


The first data set, N: Number of points. The fixed value is 10 or 0 with “10” indicating line detection and “0” indicating the opposite.

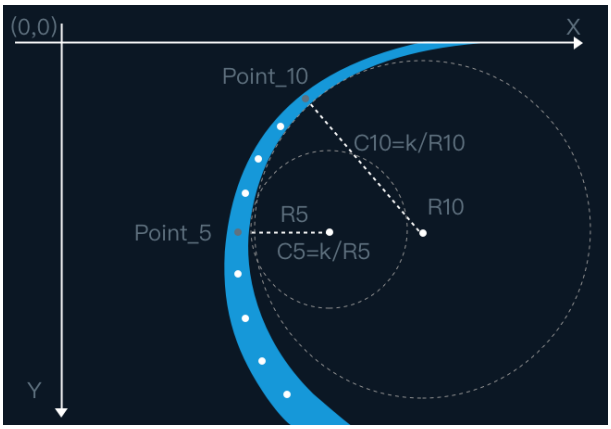
The second data set, I: line type 4 cases: “0”, indicates no line is identified; “1”, indicates one line within the FOV; “2”, indicates a Y intersection; “3”, indicates a crossroad.

The third and fourth data sets (X1, 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.



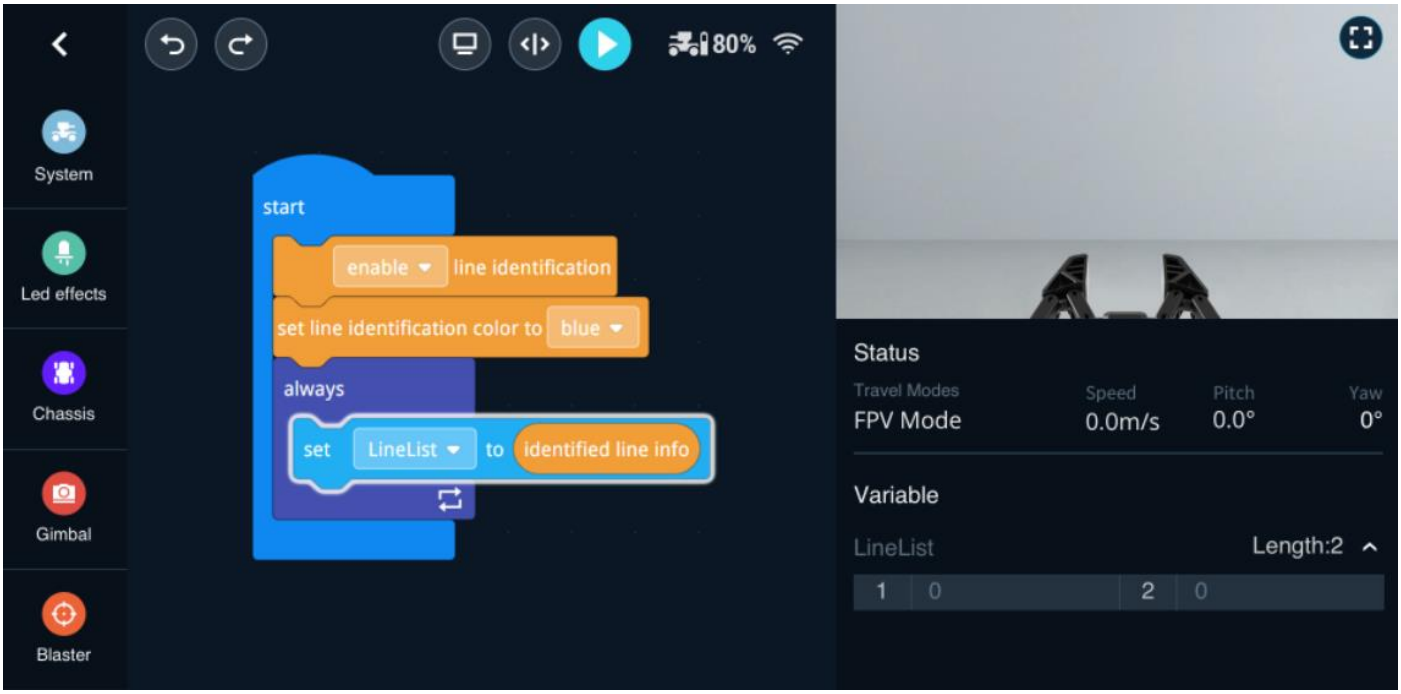


The fifth data set θ1: Tangent angle of the first point. When the tangent angle is 0, it means that the angle between the tangent and the vertical direction of this point is 90 degrees; when the tangent angle is 90, it proves that the line that this point is very curved.



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.



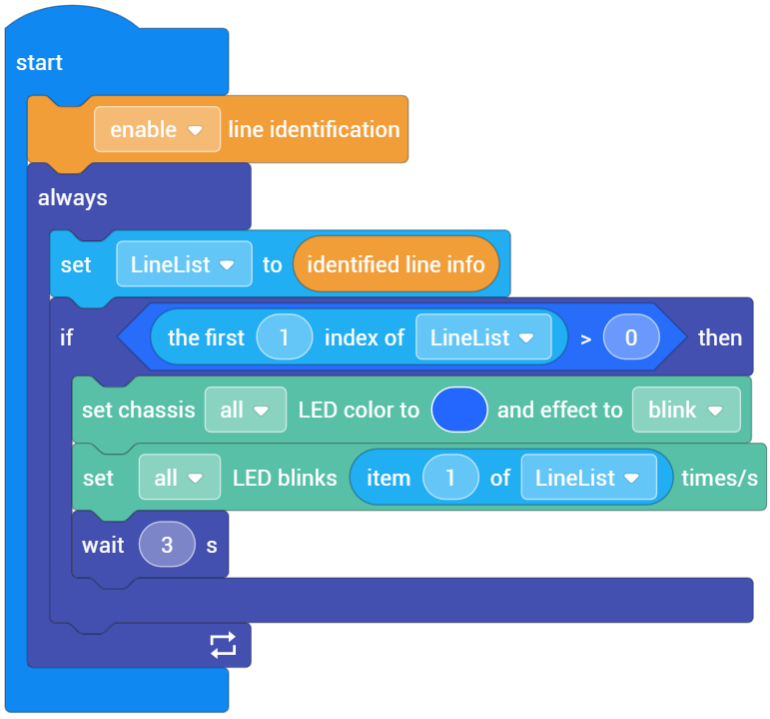
19. Identified multi-line info

(1) Description: Obtain identified multi-line information. The parameters are n, X, Y, θ, and C.

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

(3) Example: Multi-line blink indication

When the robot identifies N lines, all the chassis LED indicators blink blue N times.



Notes:

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.

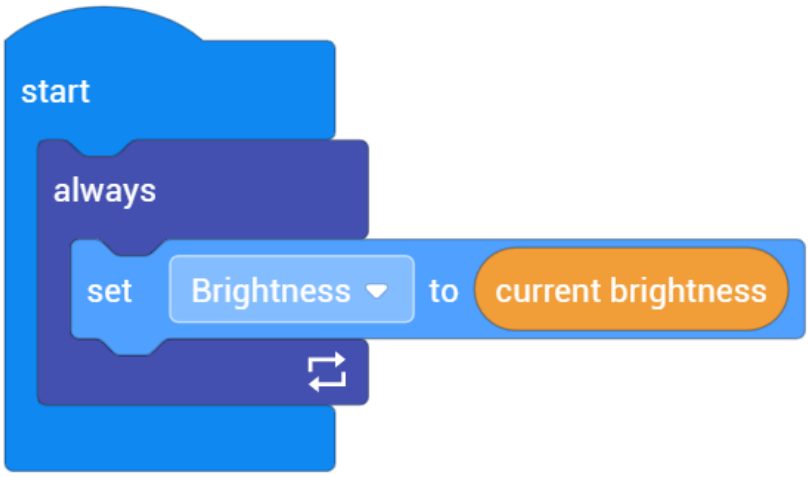
20. Current brightness

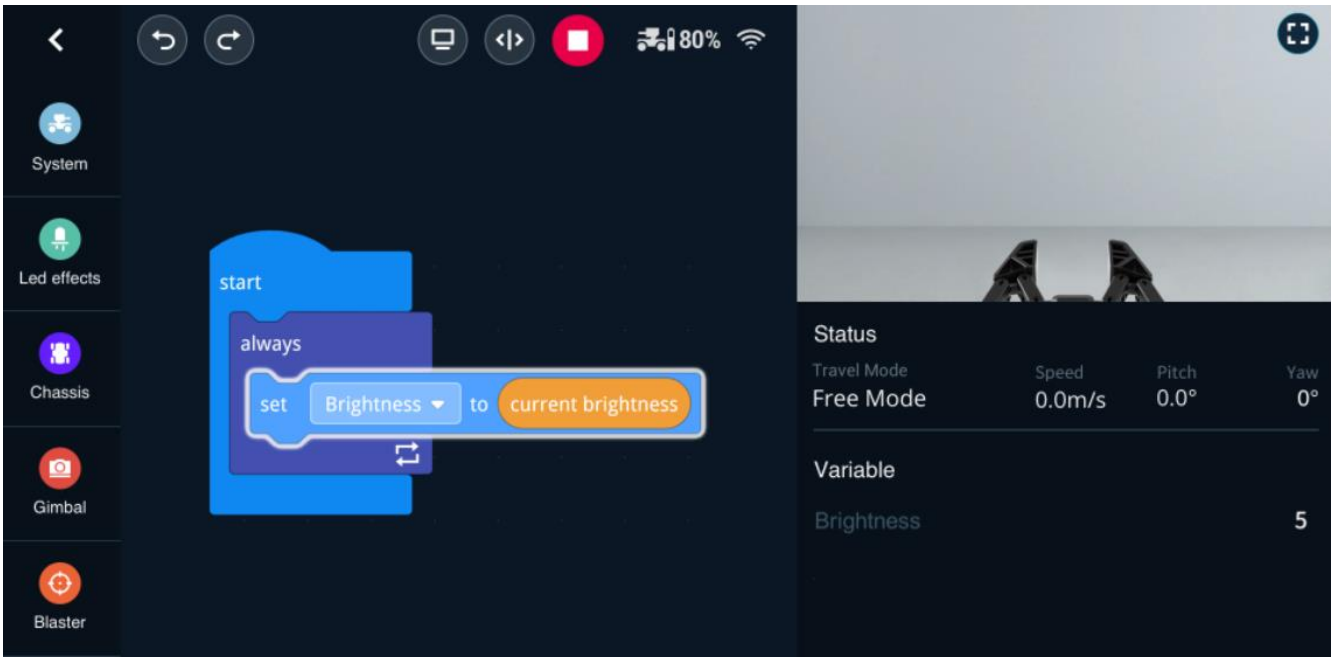
(1) Description: Obtain information about the brightness of the current environment and returns a value of 0-10; the greater the value is, the brighter the environment.

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

(3) Example: Obtain ambient light data

Observe the changes in current ambient light values on the FPV interface.





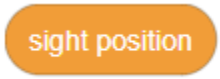
Python API:

Function: vision_ctrl.get_env_brightness( )

Return value:

- brightness_value(int)

21. Sight position

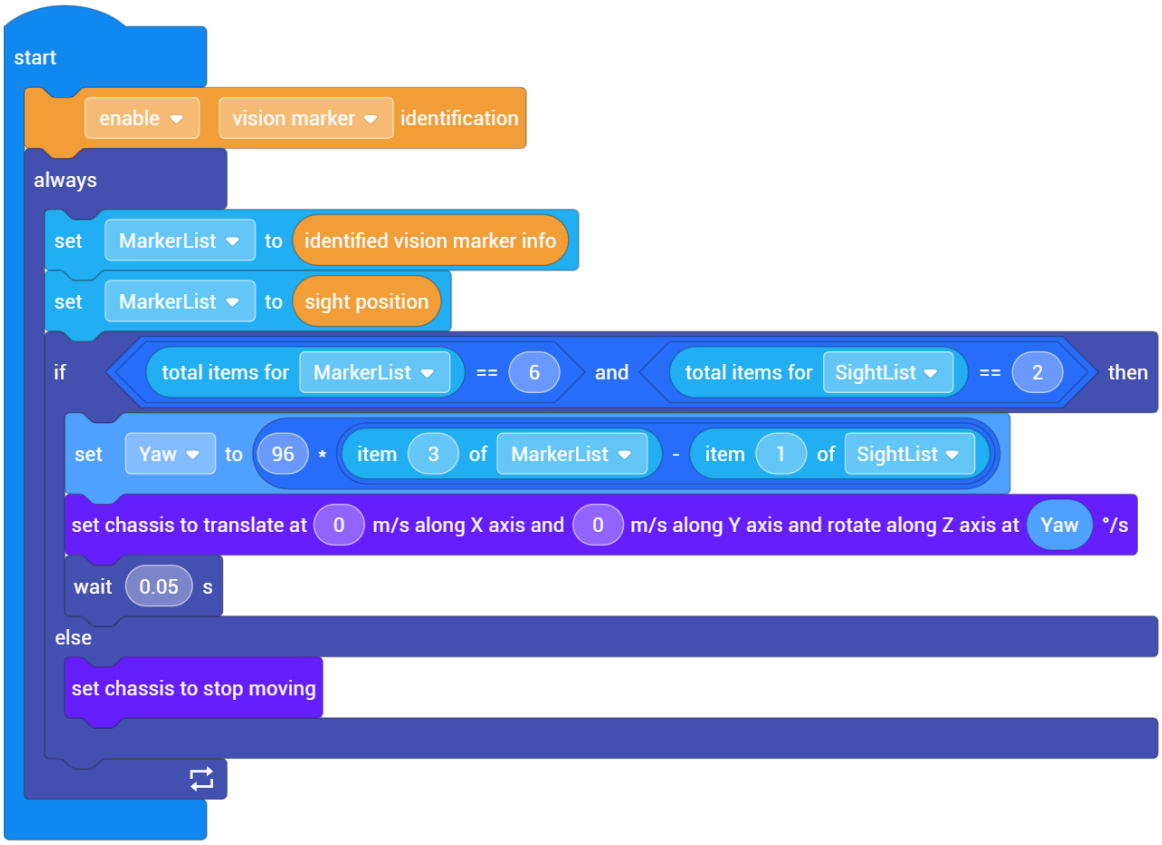


(1) Description: Obtain information for sight position in terms of the parameters X and Y.

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

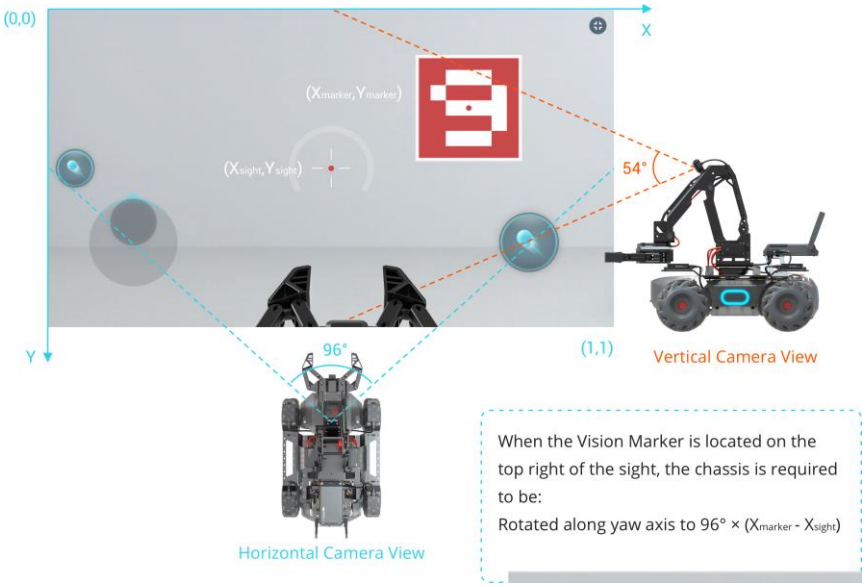
(3) Example: Follow vision markers

This will convert the difference between the sight position and a Vision Marker in the robot's field of view into a chassis rotation angle value in order to direct the chassis to move toward the Vision Marker.



Note:

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



Python API:

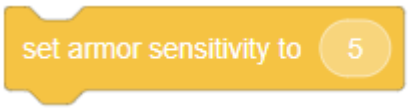
Function: media_ctrl.get_sight_bead_position( )

Return value:

- sight_bead_position(list)

Armor

1. Set armor sensitivity to (5)

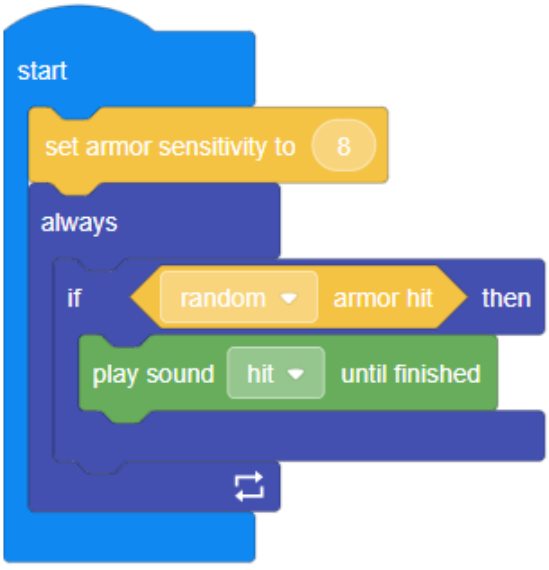


(1) Description: Set armor sensitivity; the larger the value, the higher the armor sensitivity. For testing armor sensitivity with hard objects or by tapping, the recommended values are 6 and 8, respectively.

(2) Type: Settings block

(3) Sample: Response to tapping

Tap anywhere on the EP CORE'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.

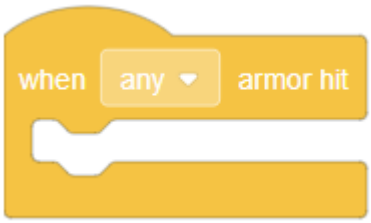
Python API:

Function: armor_ctrl.set_hit_sensitivity(value)

Parameters:

- value(int): [0, 10]

2. When (any) armor hit

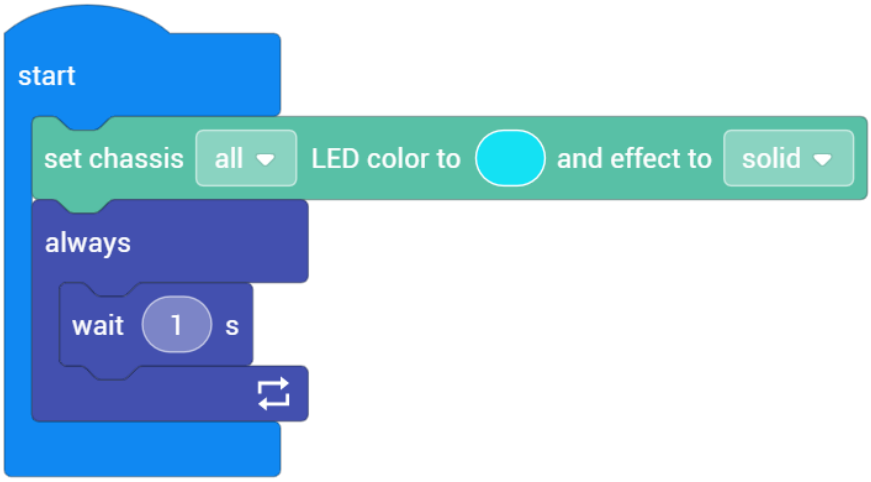


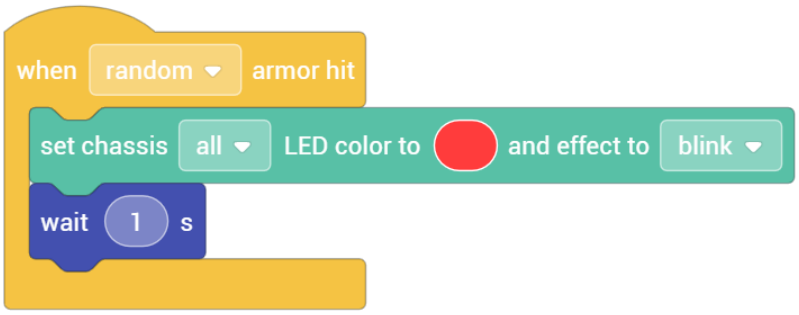
(1) Description: Run the block's program when the armor is hit at a specified section

(2) Type: Event block

(3) Sample: Response to tapping

When any armor of the robot is attacked, all chassis LEDs will blink red to indicate.





Note:

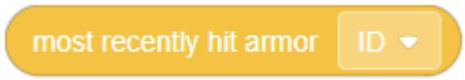
Event blocks have the highest priority and contain programs where main function will pop out and begin to run when certain conditional statements are met, regardless of the main function's current status.

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)

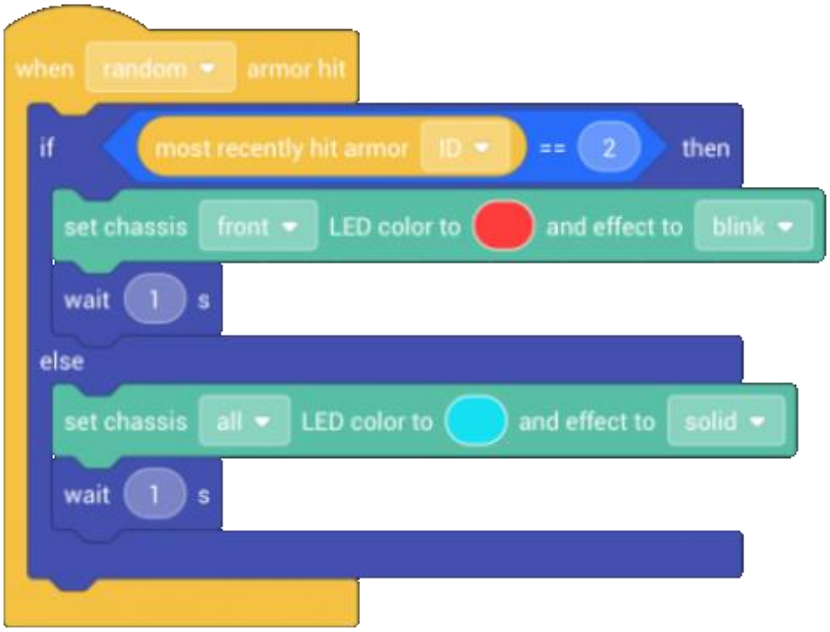


(1) Description: Display information for the last armor section hit; the ID value indicates the specific section that was hit, and the running time shows the time it was hit

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

(3) Example: Hit indicator

If the front armor of the chassis is attacked, the front LED of the chassis will blink red to indicate.



Notes:

The returned ID 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

Python API:

Function: armor_ctrl.get_last_hit_index( )

Return value:

- index(int)

Function: armor_ctrl.get_last_hit_time( )

Return value:

- time(float)

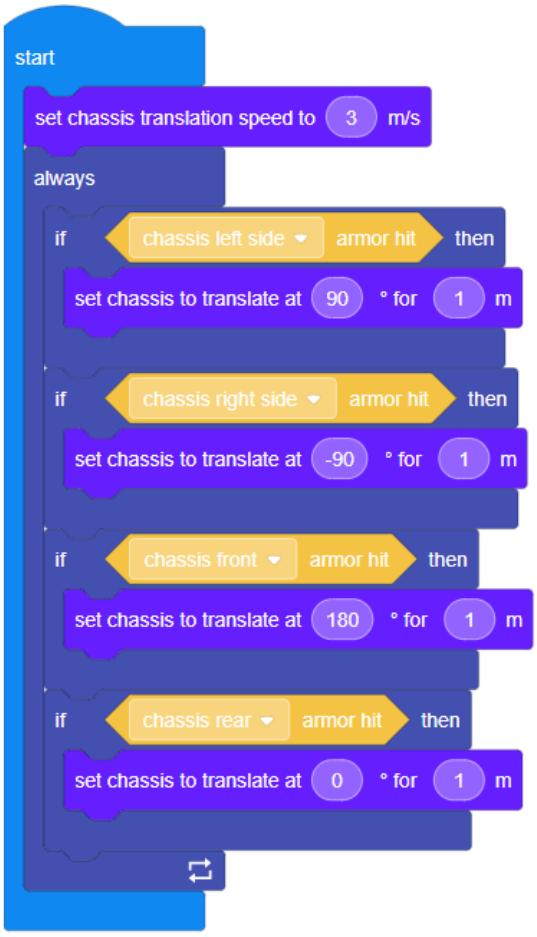
4. (Any) armor hit

(1) Description: Continuously detect 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

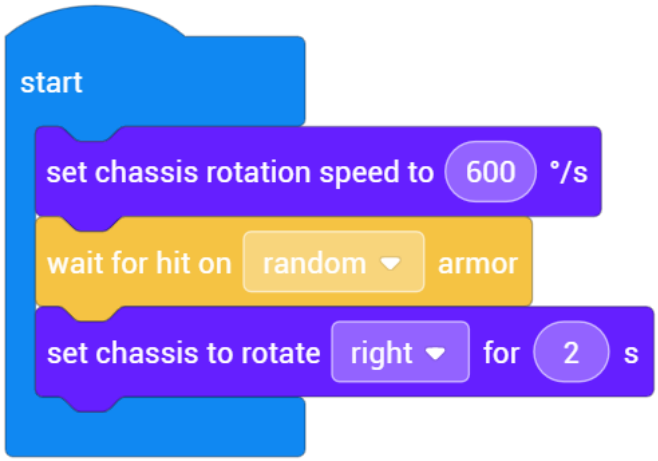
5. Wait for hit on (any) armor

(1) Description: Execute the next command when the specified armor is hit; otherwise, it continues to wait

(2) Type: Execution, blocking block

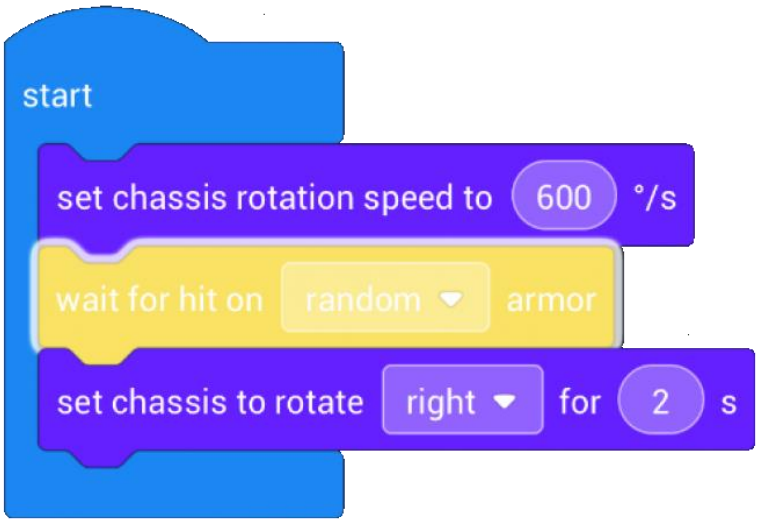
(3) Example: Self-defense

After the robot is attacked, the chassis quickly rotates to form a self-defense circle.



Note:

In this program, if the robot has no armor attacked, the program will stay on this block and wait when the block remains highlighted.



Python API:

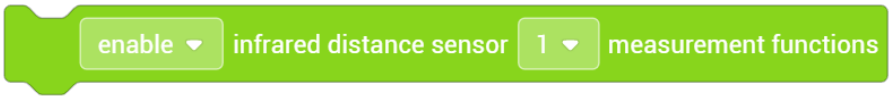
Function: armor_ctrl.cond_wait(condition_enum)

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

Sensor

1. (Enable) infrared distance sensor No. (1) measurement functions

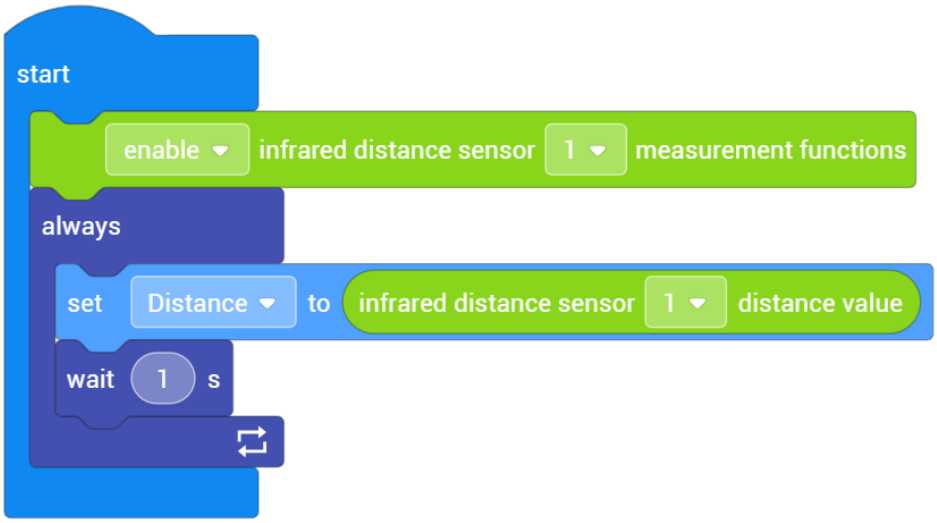


(1) Description: Turn on or turn off the distance measuring function of a specified infrared distance sensor (No.).

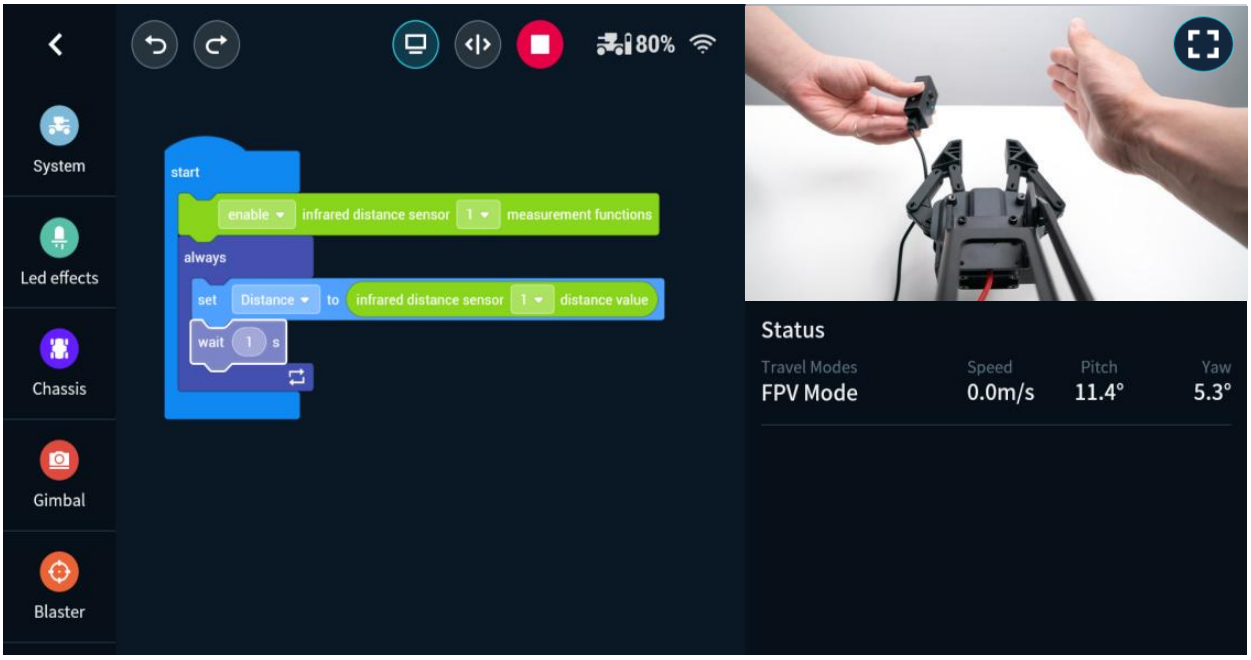
(2) Type: Settings block

(3) Example: Distance measuring using infrared beams

Measure the distance between the infrared distance sensor and your palm.



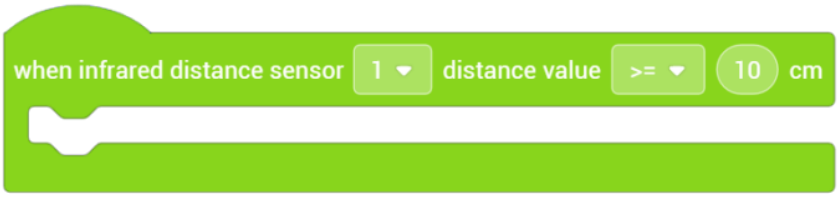
Move your palm forward and backward and observe the real time distance change through the FPV window.



Notes:

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

2. When infrared distance sensor No. (1) distance value (≥) (10) centimeters

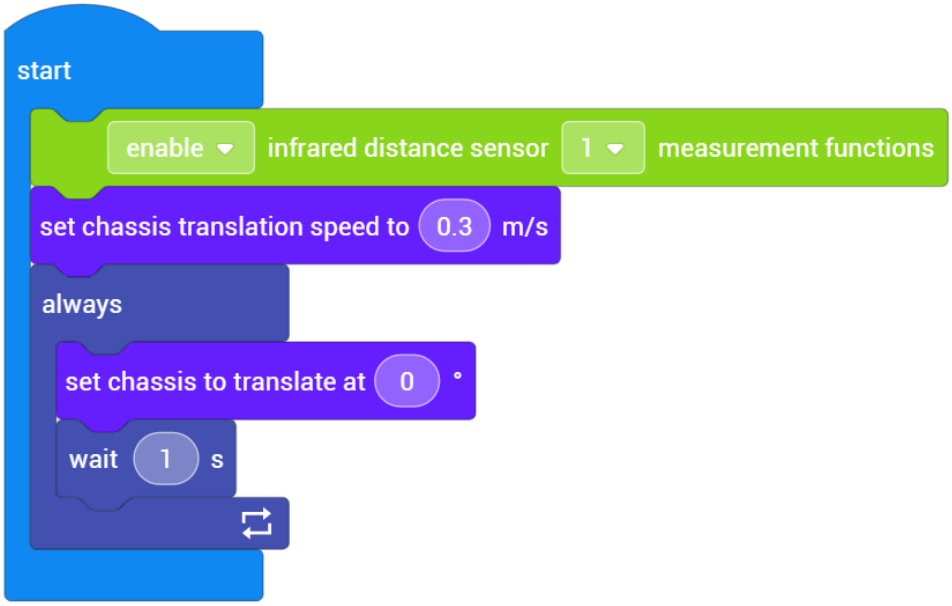


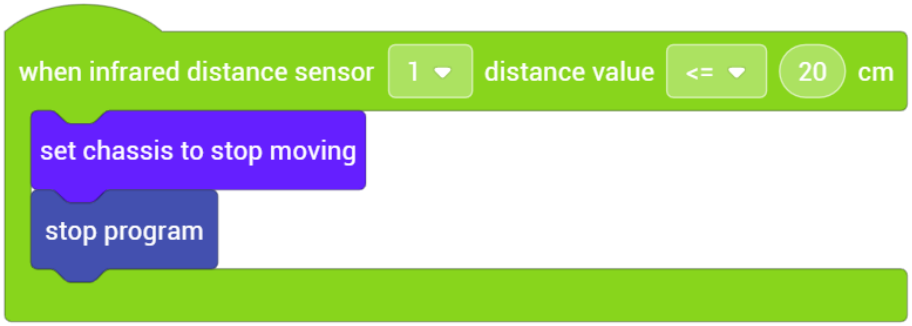
(1) Description: Run the program in this module when the distance measured by the specified infrared distance sensor (No.) meets the condition.

(2) Type: Event block

(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.





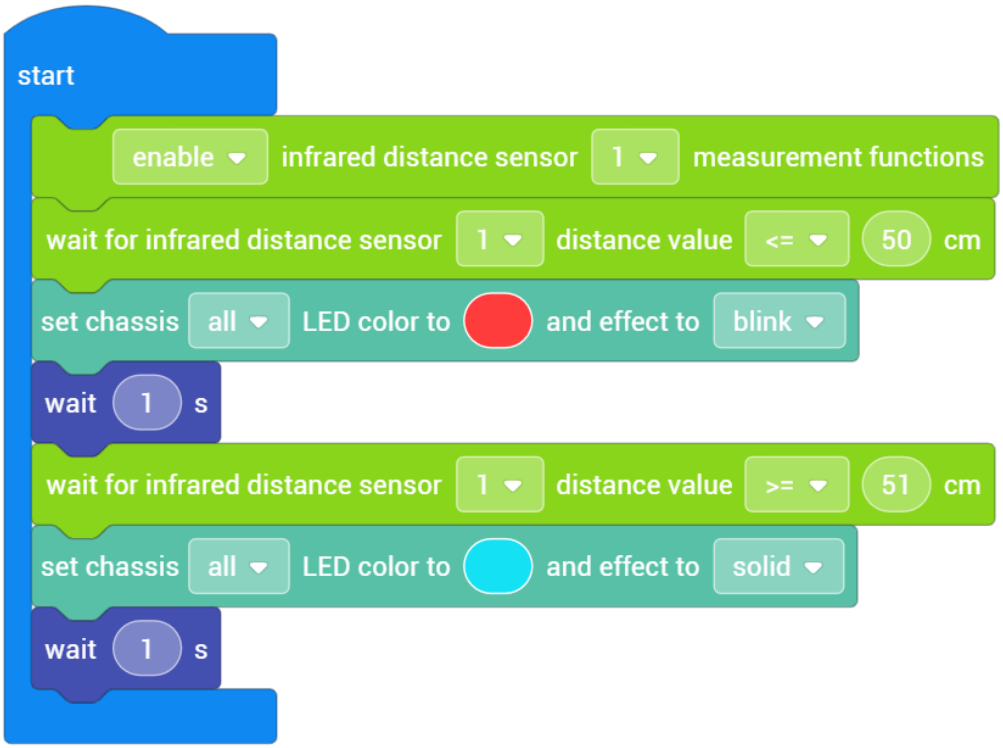
3. Wait for infrared distance sensor No. (1) distance value (≥) (10) centimeters

(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 block

(3) Example: Anti-collision warning

The robot keeps approaching the wall. When the infrared depth sensor detects that the robot is ≥50 cm away from the wall, the chassis LEDs blink red to warn, and the default LED effect is restored when the distance exceeds 50 cm.



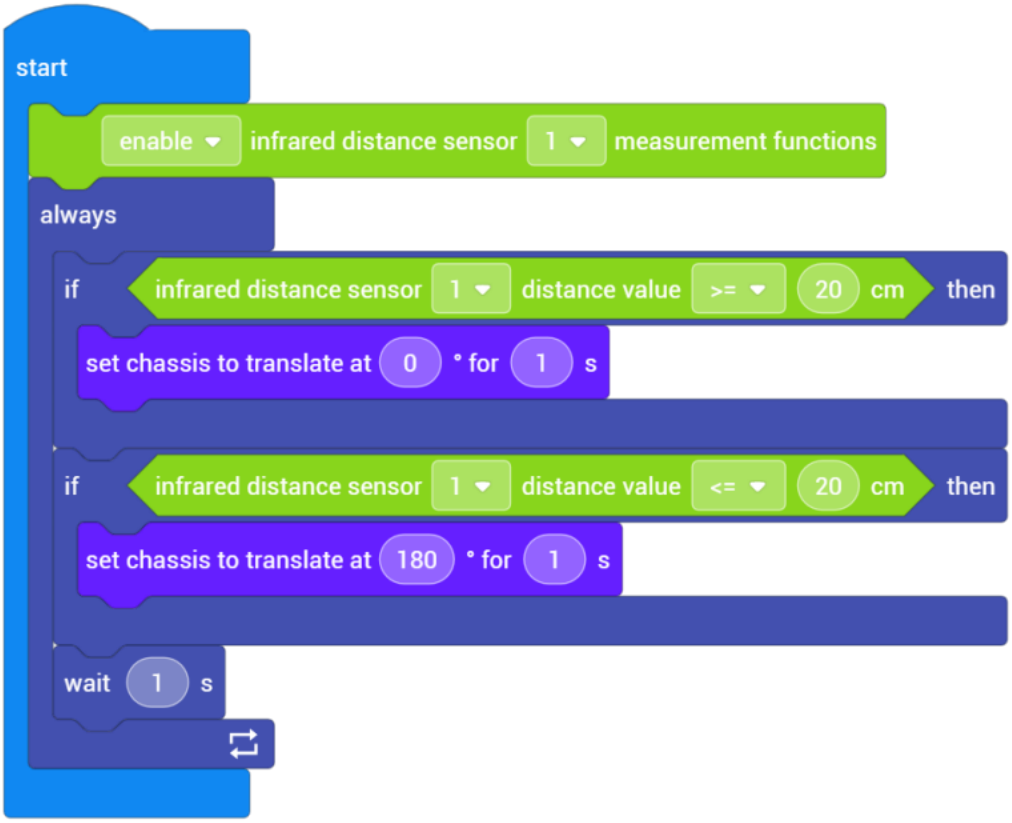
4. Infrared distance sensor No. (1) distance value (≥) (10) centimeters

(1) Description: Return “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.



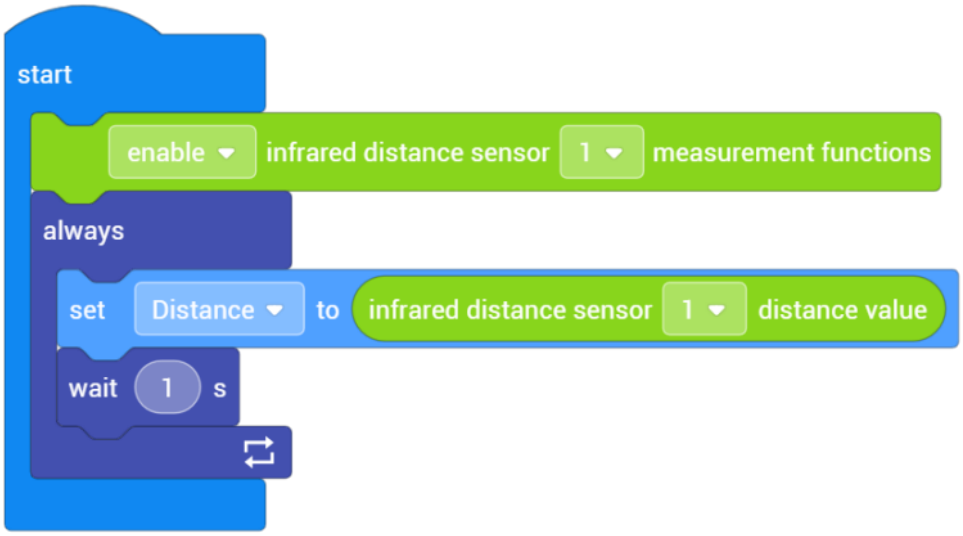
5. Infrared distance sensor No. (1) distance value

(1) Description: Obtain the distance measured by the specified infrared distance sensor (serial No.). The unit is centimeter.

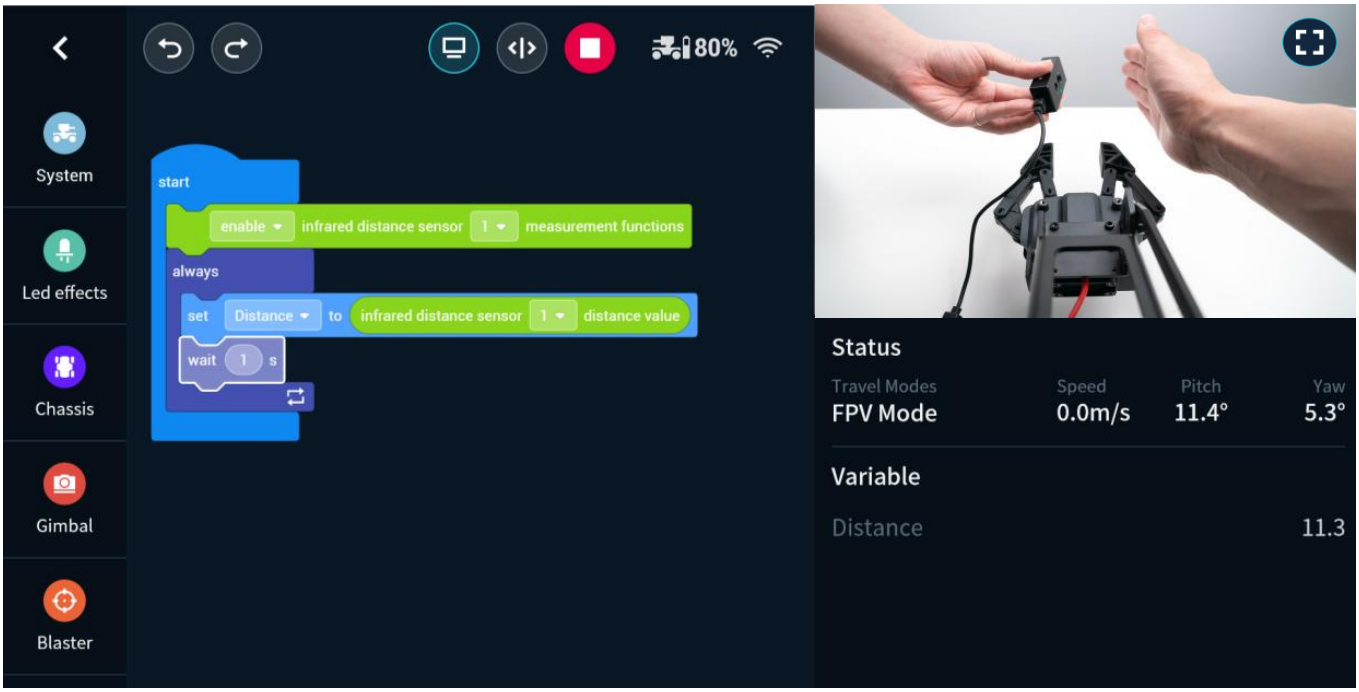
(2) Type: Information block

(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.

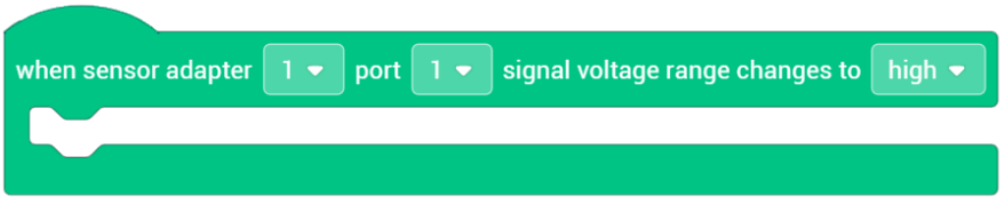


Note:

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

Sensor Adaptor

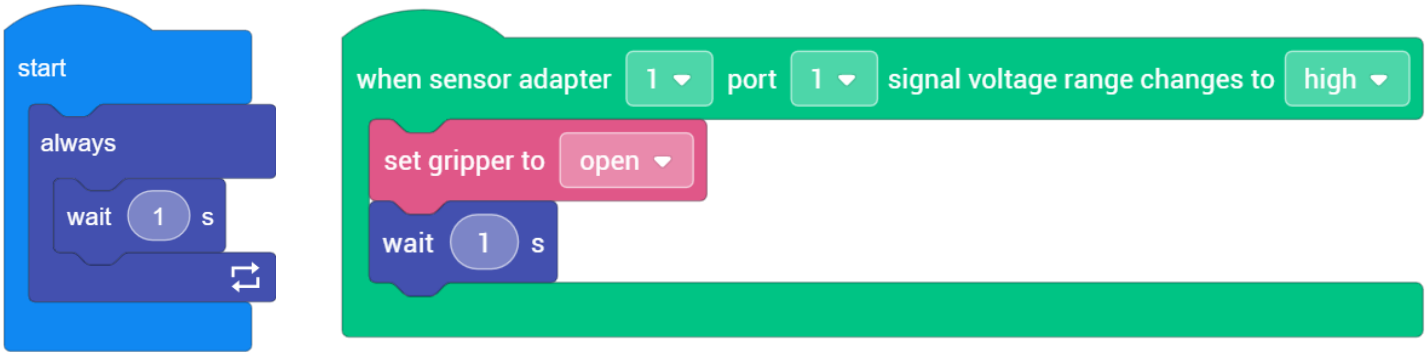
1. When sensor adaptor No. (1) signal voltage range changes to (high)



(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 block

(3) Example: Use a button to control the opening or closing of the gripper

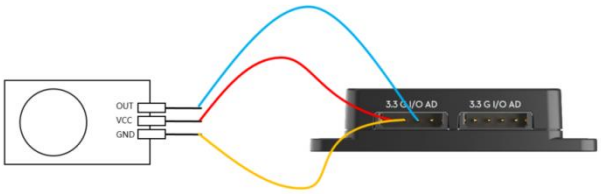


Connect a button to sensor adaptor No. (1). When the button is pressed, the robot gripper opens. When the button is released, the robot gripper closes.

Notes:

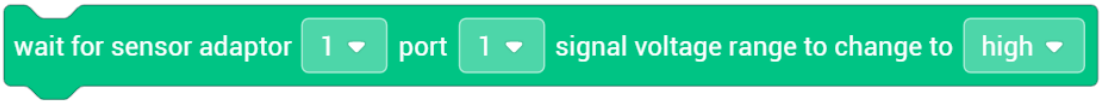
1) The output level of the button can be high or low. When the button is “pressed” and “released”, the output level changes, and the I/O pin of the sensor adaptor receives the level signal change so as to control the opening or closing of the gripper.
2) The three pins on the button need to be connected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



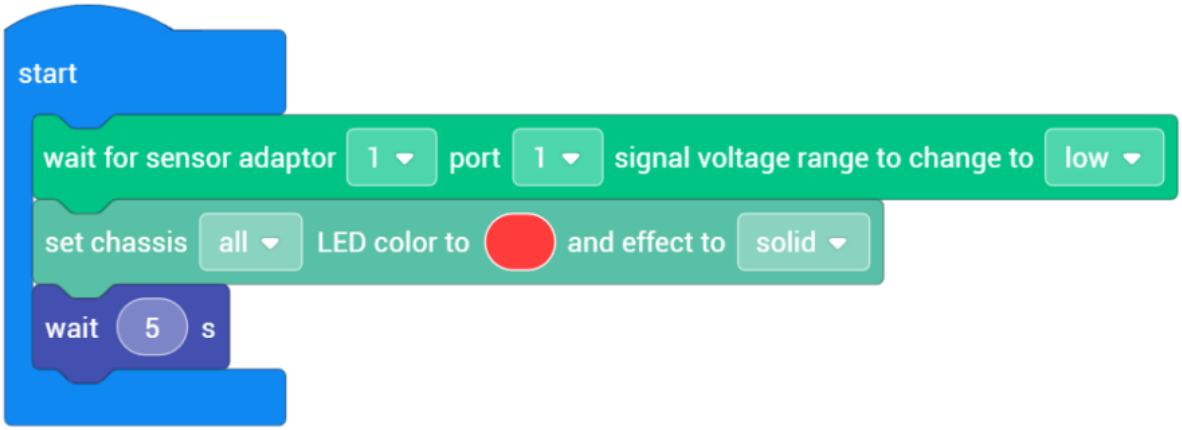
3) A maximum of six sensor adaptors can be connected to the robot at the same time.

2. Wait for sensor adaptor No. (1) port (1) signal voltage range to change to (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 block
(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.



Notes:

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 counting		VCC(5V) GND OUT(TTL)	High or low electrical level
Touch sensor		VCC(5V) GND OUT(TTL)	High or low electrical level

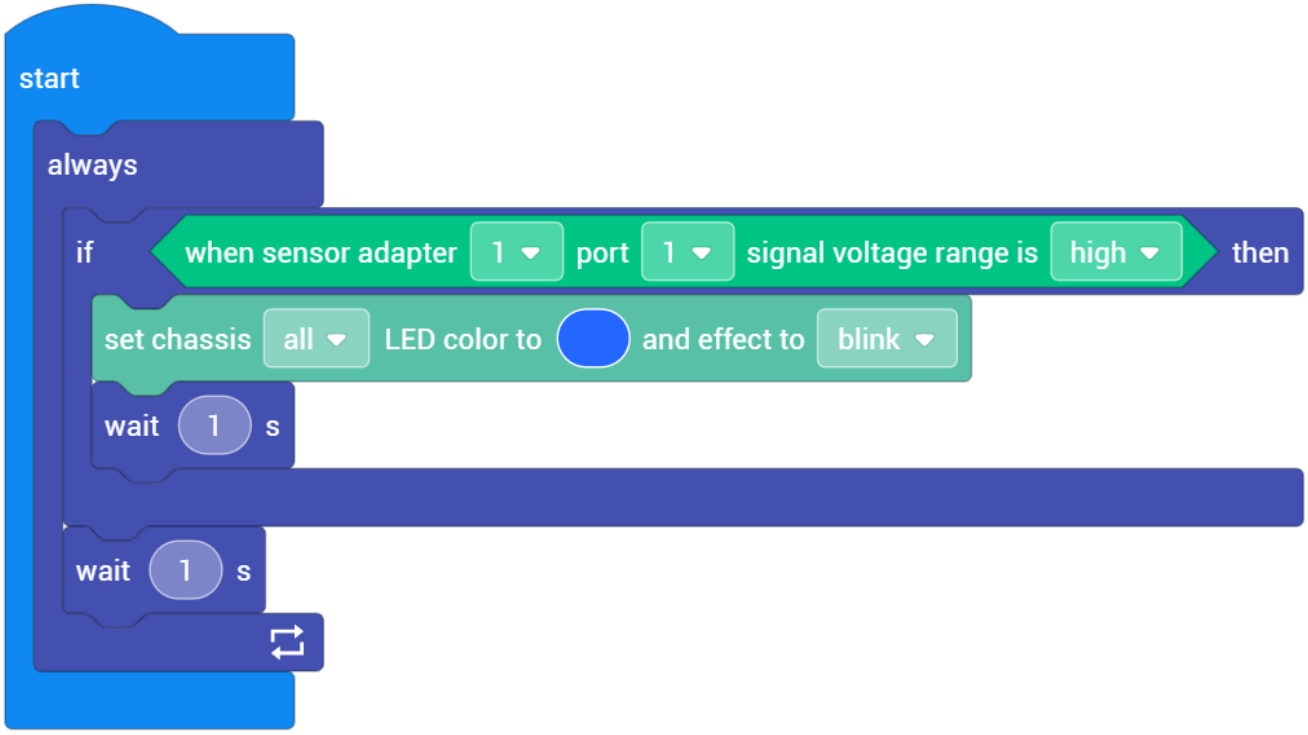
3. When sensor adaptor No. (1) port (1) signal voltage range is (high)

(1) Description: Return “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). When the finger touches the sensor, the chassis LED blinks to indicate.



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.

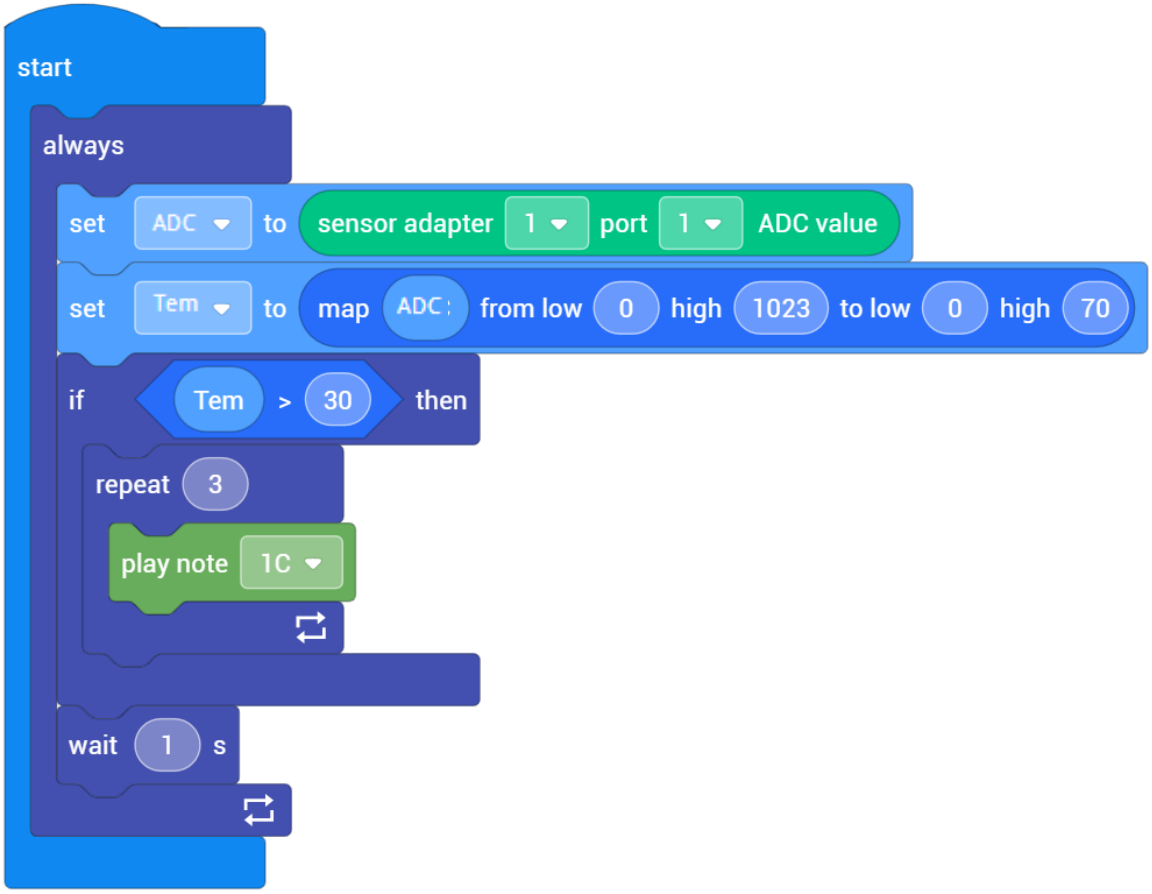
4. Sensor adaptor (1) port (1) ADC value

(1) Description: Read the ADC pin value of a specified port of a specified sensor adaptor.

(2) Return value: Information block

(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.

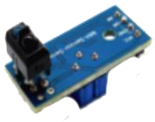


Notes:

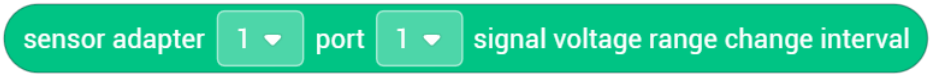
1) The sensor adaptor ADC is a 10-bit ADC which can take data of up to 1024 unique values, so the ADC range is from 0 to 1023.

2) ADC collects the 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. Sensor adaptor (1) port (1) signal voltage range change interval

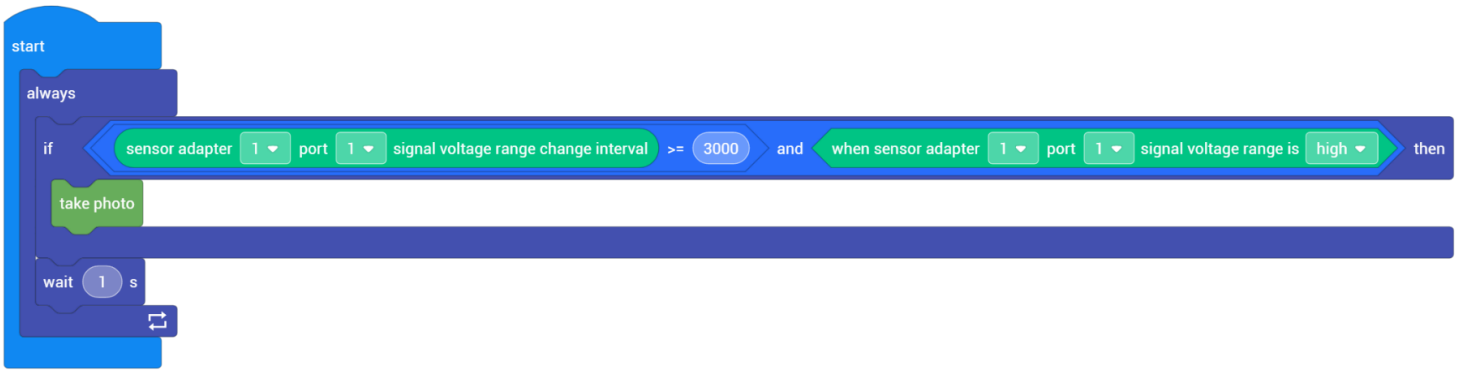


(1) Description: Obtain the duration from when the pulse of the specified I/O pin of the specified sensor adaptor jumps till it is obtained. The unit is millisecond.

(2) Return value: Information block

(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) Description: Obtain the current angle value of a mobile device

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

(3) Example: Dynamic rotation control

Rotate the mobile device to the left or right and observe whether the RoboMaster EP CORE's chassis rotates accordingly.

Notes:

1) Mobile device refers to devices such as smartphones and tablets.

2) The built-in sensor settings of mobile devices may be different in models, obtain the value sent by the sensor in advance.

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

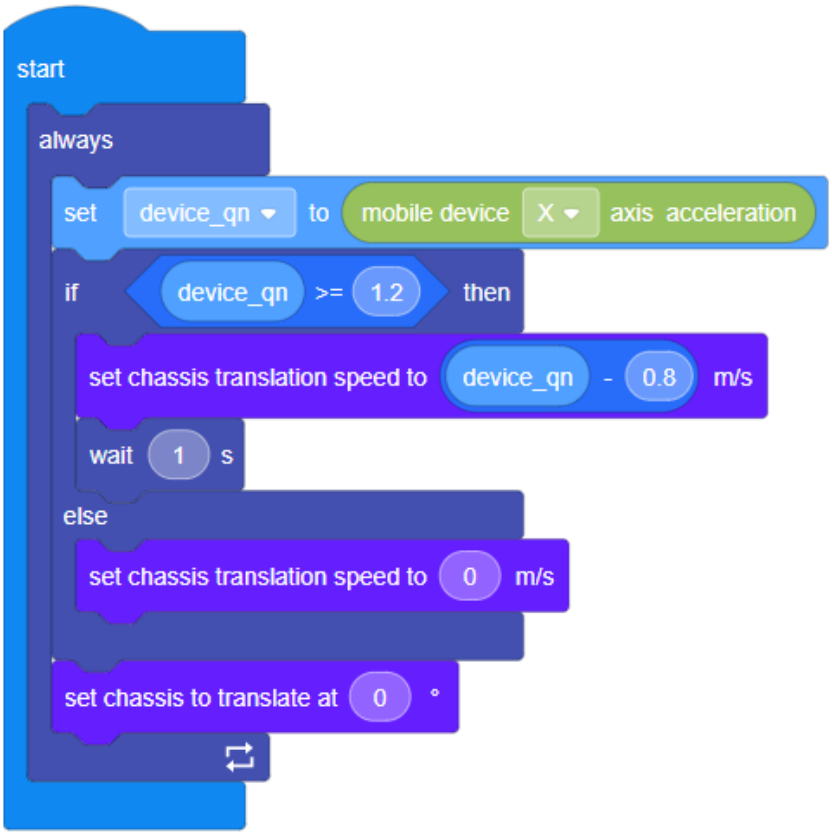
2. Mobile device (x-axis) acceleration

(1) Description: Obtain the exact rate of acceleration for a mobile device

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

(3) Example: Dynamic forward control

Wave your mobile device up and down to control the RoboMaster EP CORE's motion in a forward direction.



Notes:

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.
3) The built-in sensor settings of mobile devices may be differ in models, obtain the value sent by the sensor in advance.

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

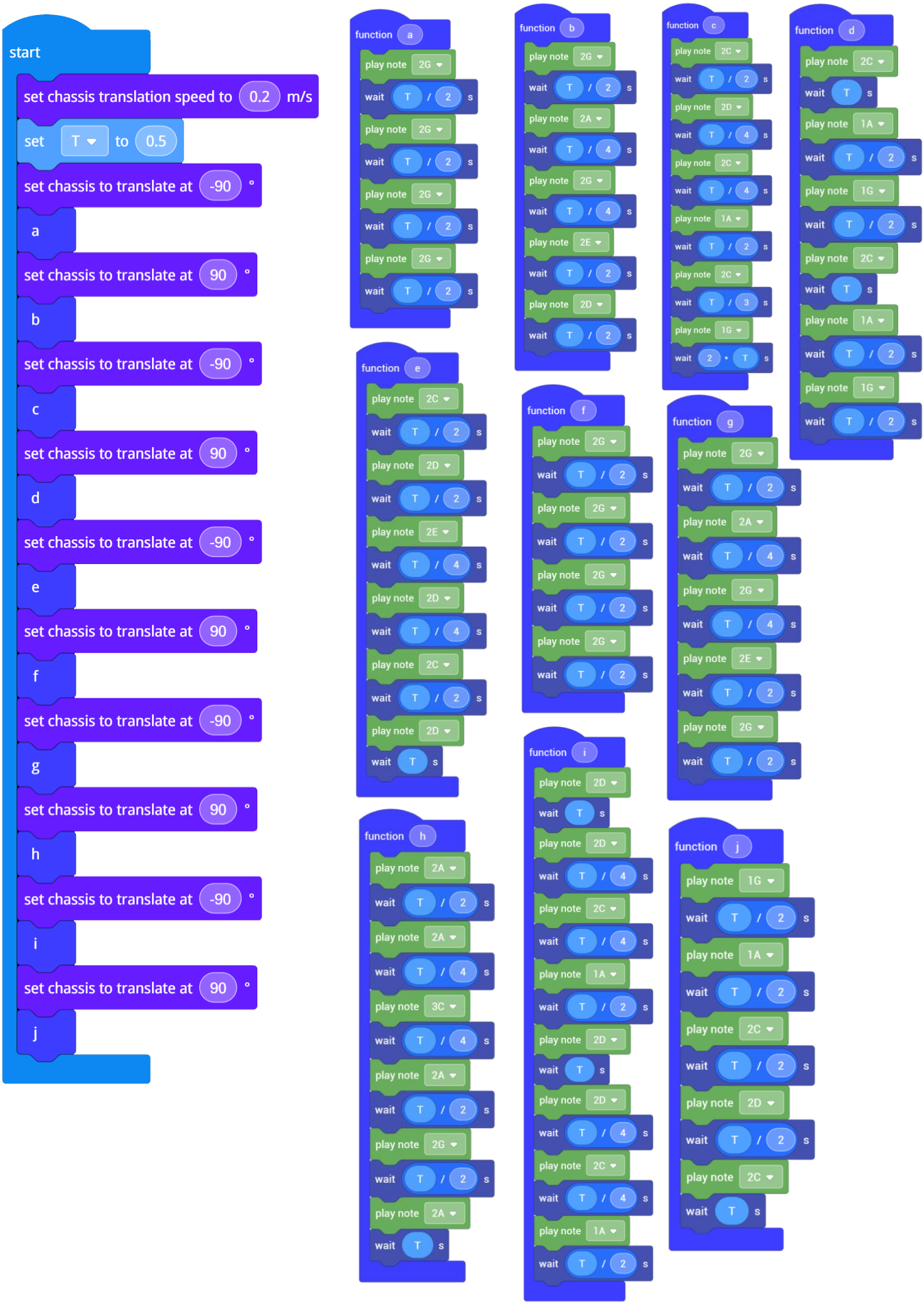
1. Play note (1C)

(1) Description: Choose notes and play them in a piano tone.

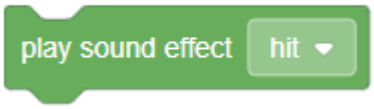
(2) Type: Execution, Non-blocking block

(3) Example: Pink Memory

Control the robot to play the notes of the song "Pink Memory".


start
set chassis translation speed to 0.2 m/s
set T to 0.5
set chassis to translate at -90 °
a
set chassis to translate at 90 °
b
set chassis to translate at -90 °
c
set chassis to translate at 90 °
d
set chassis to translate at -90 °
e
set chassis to translate at 90 °
f
set chassis to translate at -90 °
g
set chassis to translate at 90 °
h
set chassis to translate at -90 °
i
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)

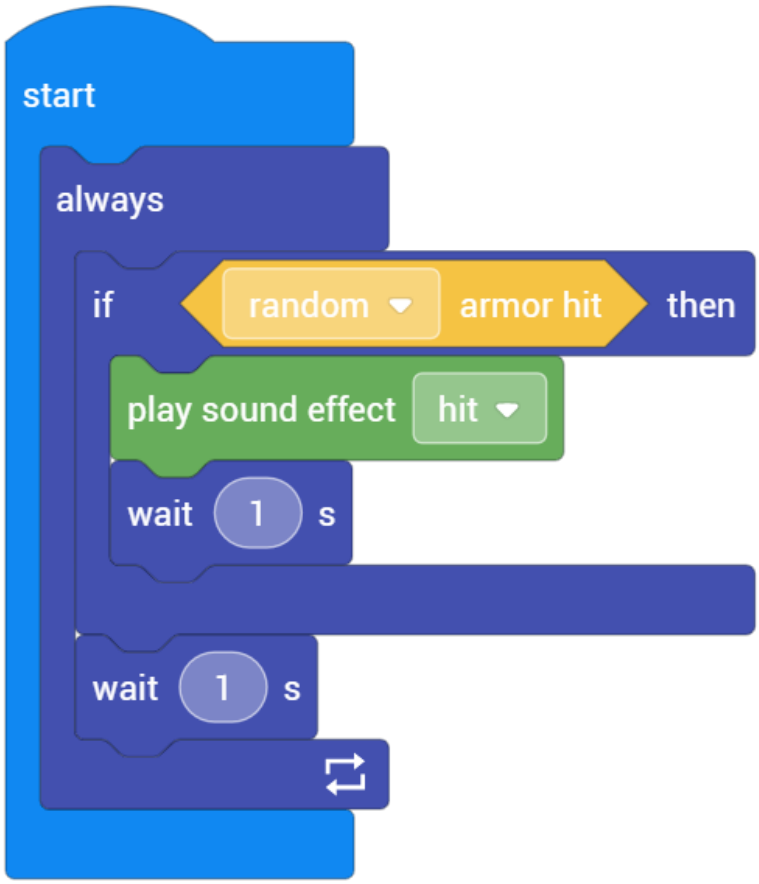


(1) Description: Set the EP CORE to emit a sound and executes the next command

(2) Type: Execution, Non-blocking block

(3) Example: Hit indicator

If the armor on either side of the chassis is attacked, it will play the hit sound effect.



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

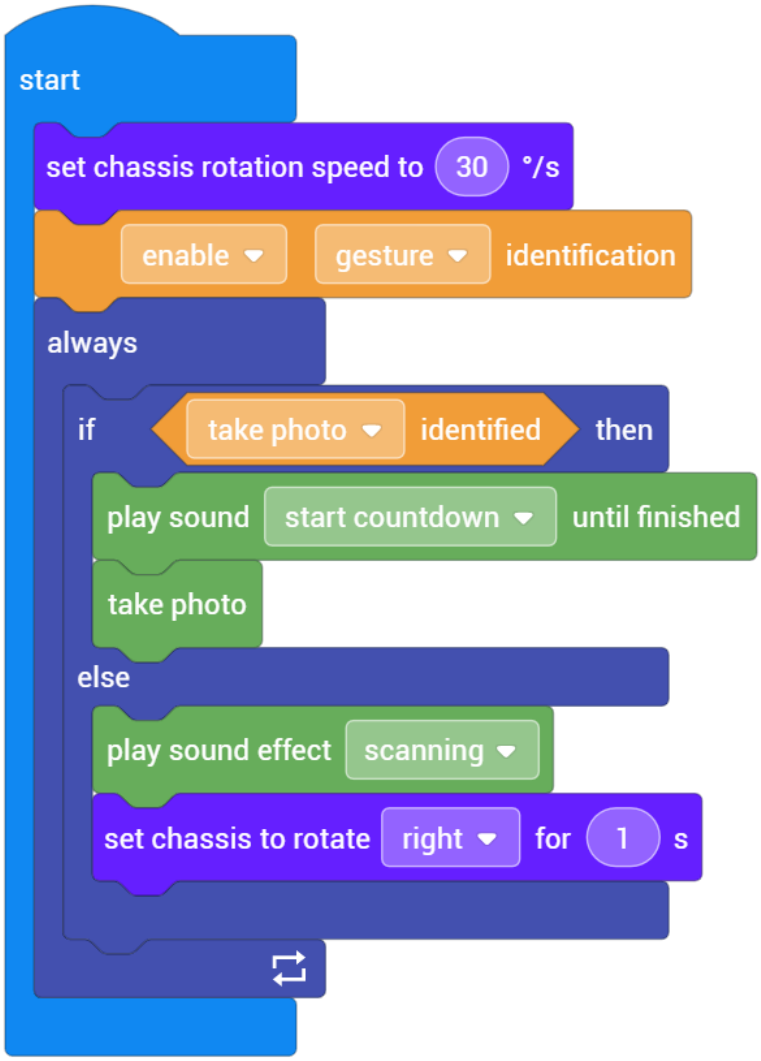
3. Play sound (hit) until finished

(1) Description: Ensure the next command will not be executed until a specified sound has been emitted

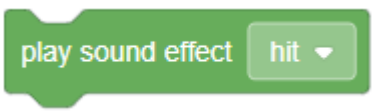
(2) Type: Execution, Blocking block

(3) Example: Remote control photograph

When the RoboMaster EP CORE 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.



Notes:



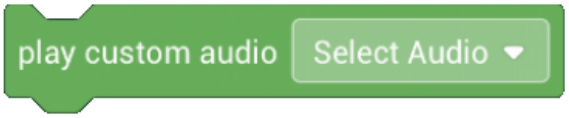
“play sound” refers to simultaneous execution of the music and the next command, similar to a musical accompaniment;

“play sound until finished” refers to the execution of playing sound and following commands one by one, similar to a musical solo (for example, it does not dance until the sound finishes playing).

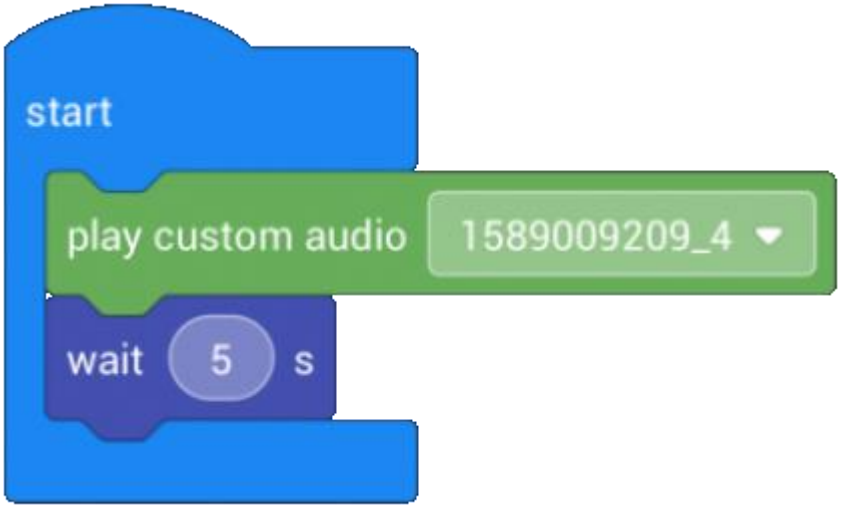
Python API:
Function: media_ctrl.play_sound(sound_enum, wait_complete_flag=True)
Parameters:

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

4. Play customized audio (select audio)



(1) Description: Play inserted customized audio
(2) Type: Execution, Non-blocking block
(3) Example: play music clip



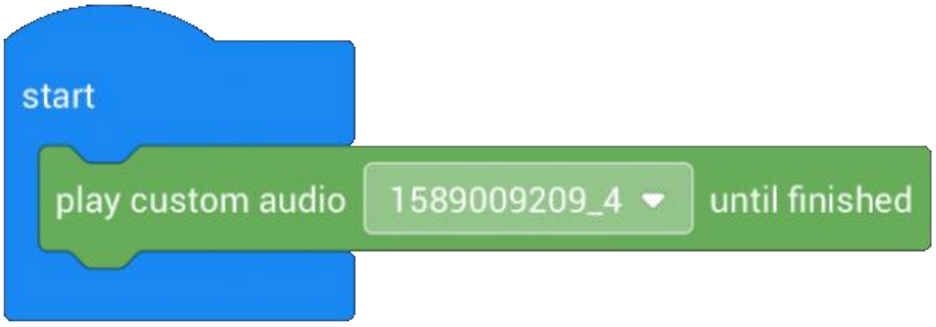
Note:
Please add a wait command after playing the customized audio, otherwise the audio will be not be played in full.

5. Play customized audio (select audio) until finished

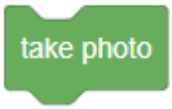
(1) Description: Play inserted customized audio, and the next command will not be executed until the audio is finished

(2) Type: Execution, Blocking block

(3) Example: play full music clip



6. Take photo

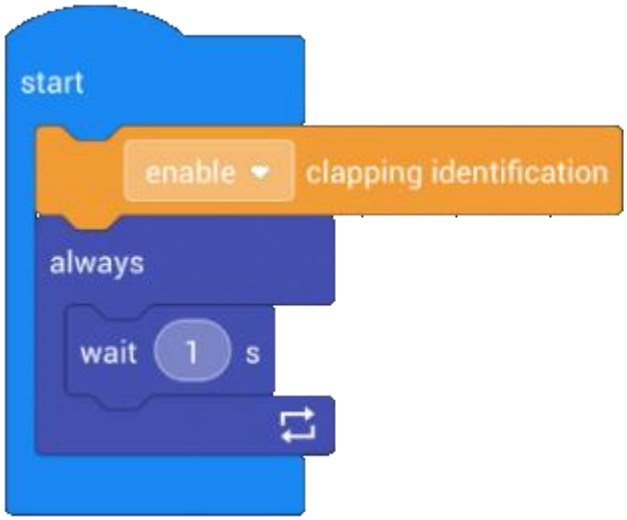


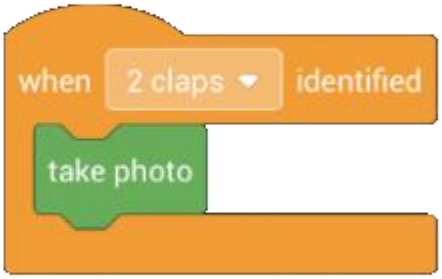
(1) Description: Set the robot to emit a specified shutter sound when it takes photos.

(2) Type: Execution block

(3) Example: Remote photo-shooting

When the robot recognizes two claps, it will take a photo.





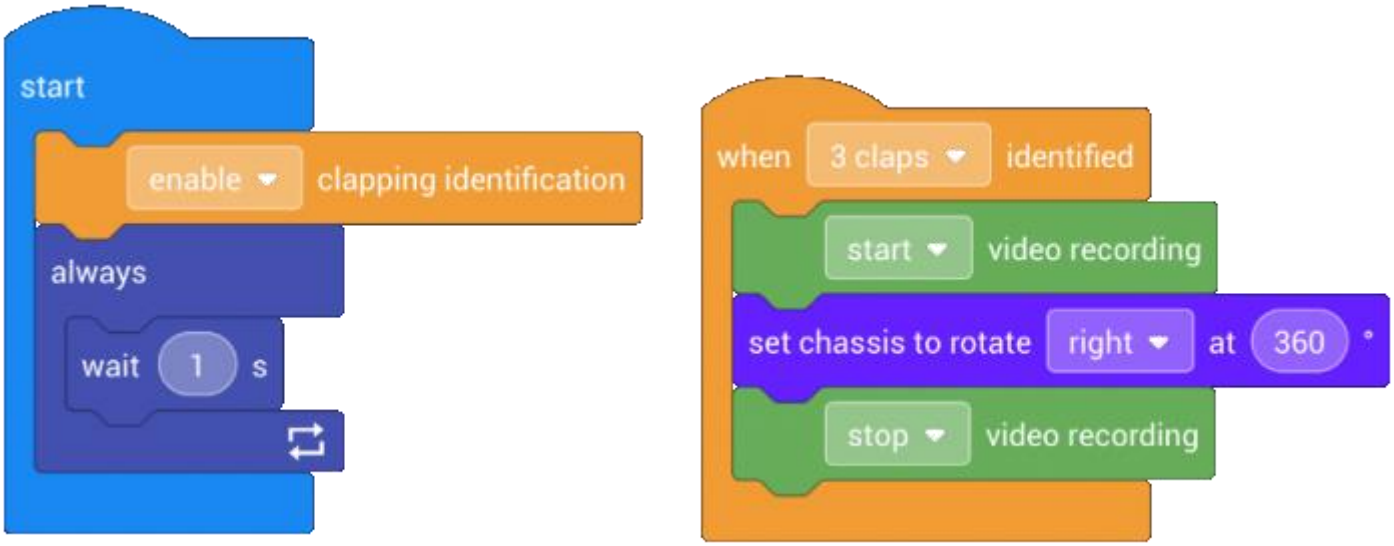
Note:

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( )

7. (Start) video recording

(1) Description: Start/stop recording; recorded videos will be saved in the APP album and SD card.
(2) Type: Execution block
(3) Example: Video recording
When the robot recognizes three claps, it rotates in a circle to record the surrounding environment video.



Note:
Users will need to insert an SD card with an available memory of more than 2GB.

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

- enable_enum(enum):
  - 1
  - 0

Commands

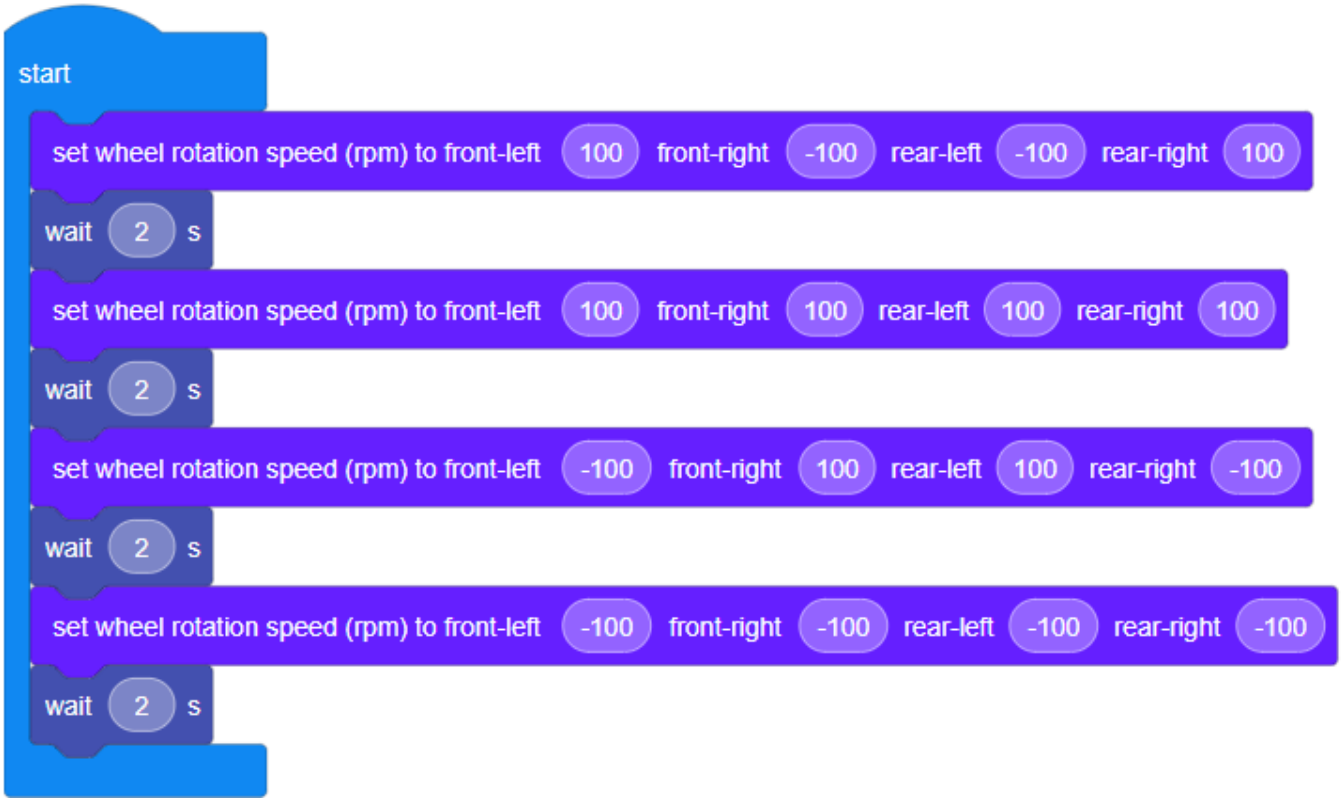
1. Wait (1) s

(1) Description: Execute the next command after waiting for a specified duration

(2) Type: Execution block

(3) Example: Complete a square

This will command the RoboMaster EP CORE to translate in the following order to complete a square every two seconds by moving as translating: “right→forward→left→backward.”



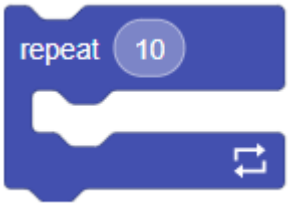
Python API

Function: time.sleep(t)

Parameters:

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

2. Repeat (10)

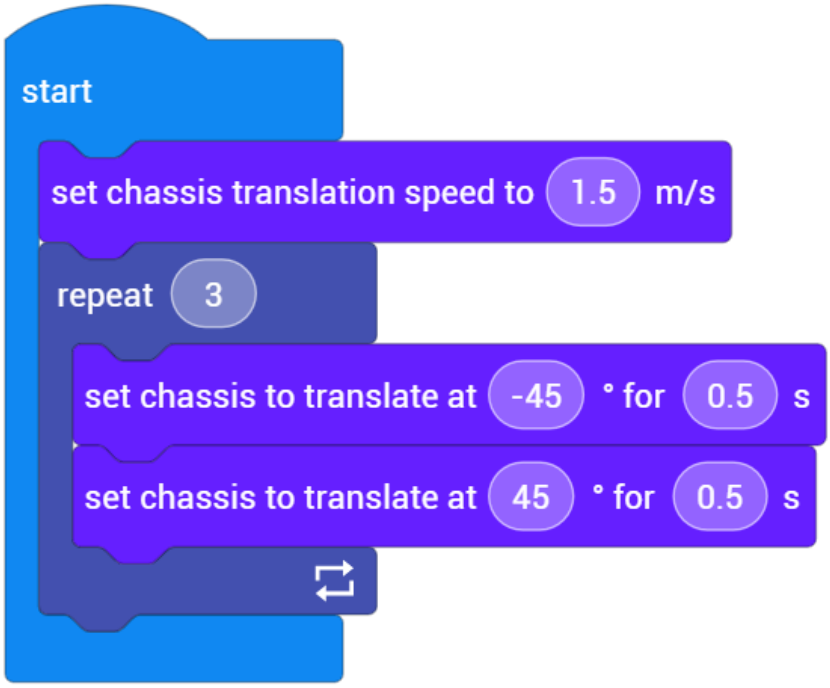


(1) Description: Configure a program to repeat the same command several times (finite loop block)

(2) Type: Execution block

(3) Example: Translate along a zigzag line

This will control the robot to translate along a zigzag line.



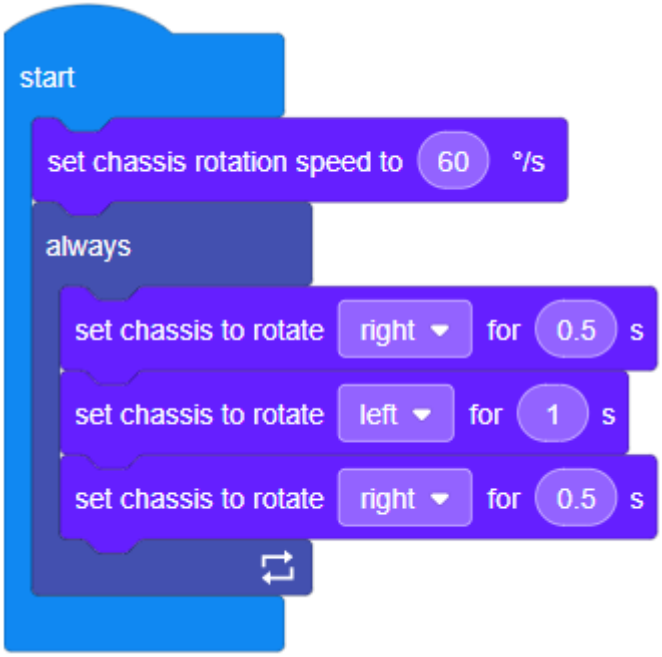
3. Always

(1) Description: Set an internal program to repeat continuously (infinite loop)

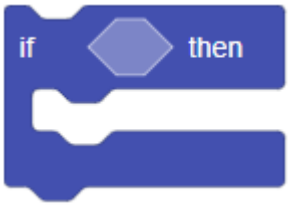
(2) Type: Execution block

(3) Example: Rotate chassis left and right

The chassis continues to rotate left and right.



4. If...then...

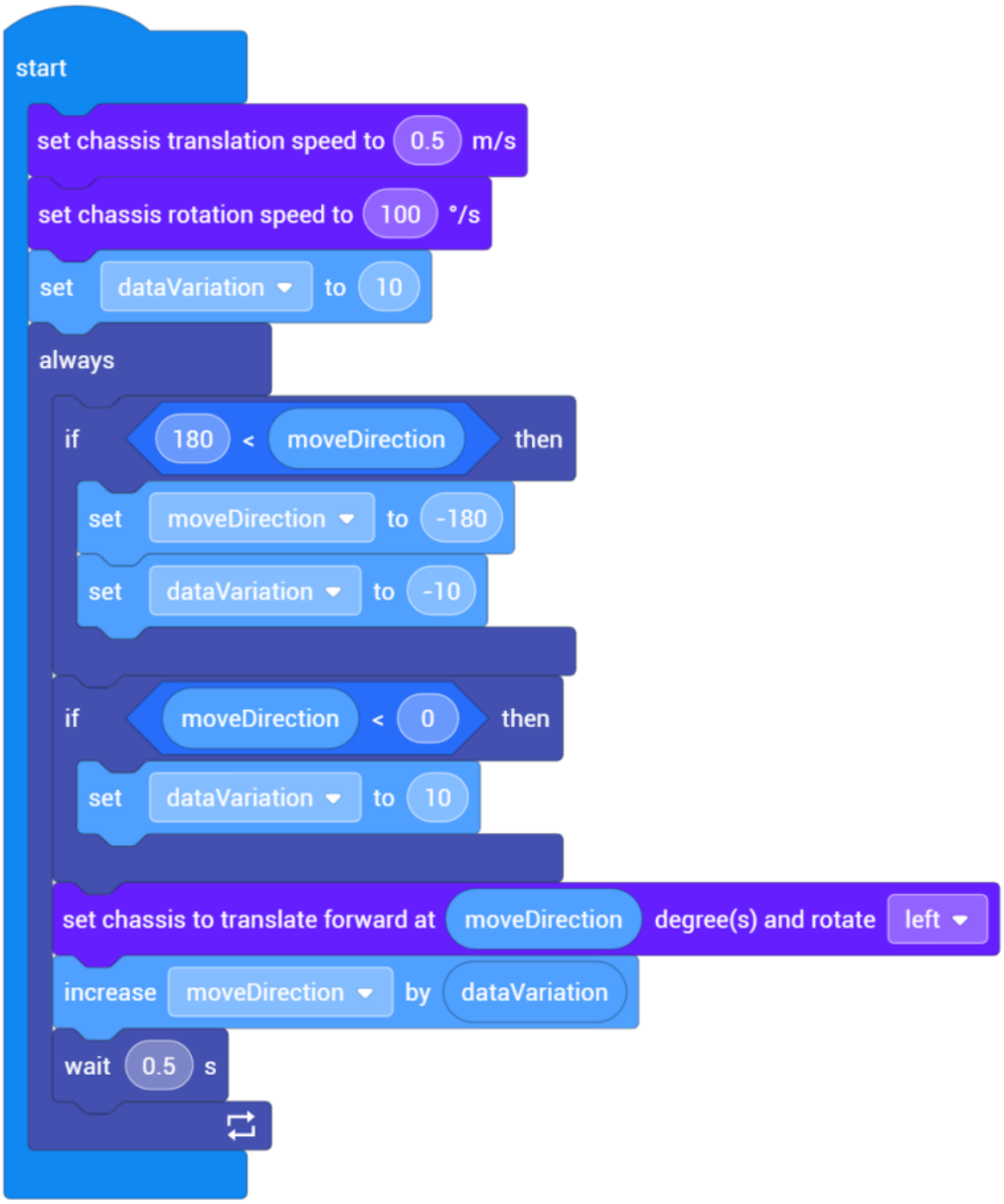


(1) Description: Run an internal program when the condition is “True”

(2) Type: Conditional statement block

(3) Example: Rotate and revolve

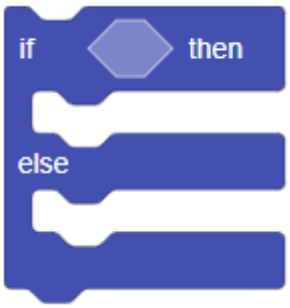
This will control the robot to revolve while rotating



Note:

The difference between conditional statement blocks and event blocks is that event blocks have higher priority when certain conditional statements are met. Regardless of which step the main function is at, the program within the event block will be finished first. Whereas conditional statement blocks will not run unless the main function preceding it and corresponding conditions are met.

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

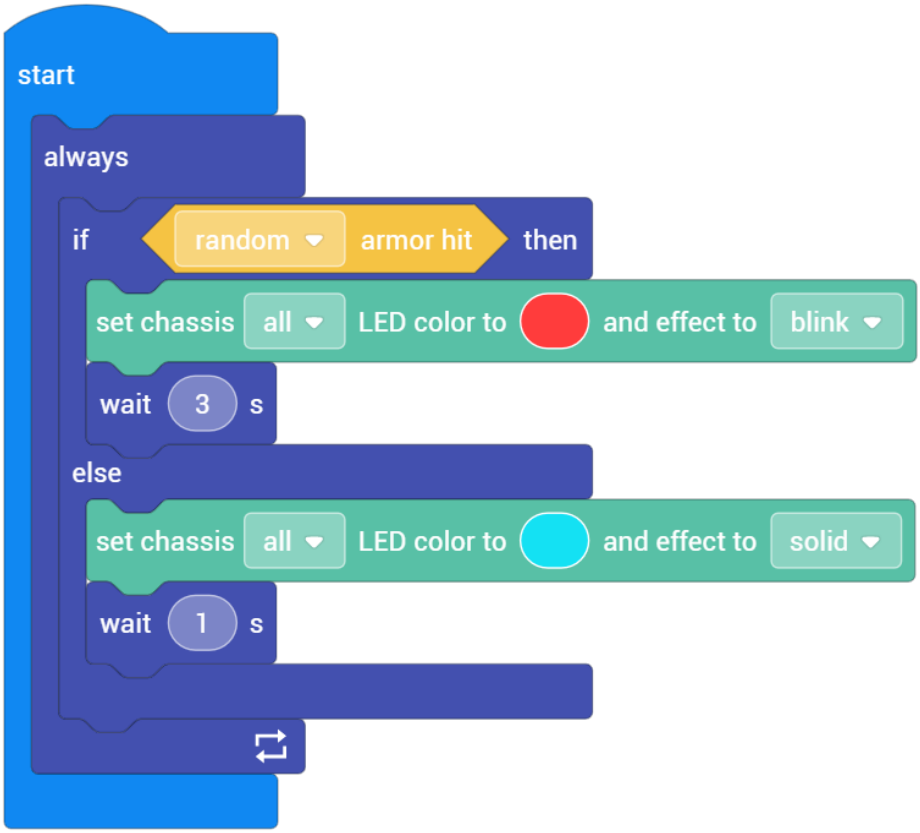


(1) Description: Run the “then” program when the condition is “True,” and runs the “else” program otherwise

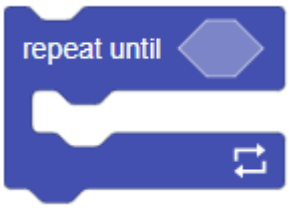
(2) Type: Conditional statement block

(3) Example: Hit indicator

If any armor of the robot is attacked, all the chassis LEDs will blink red; otherwise, the default LED effect is solid on.



6. Repeat until...

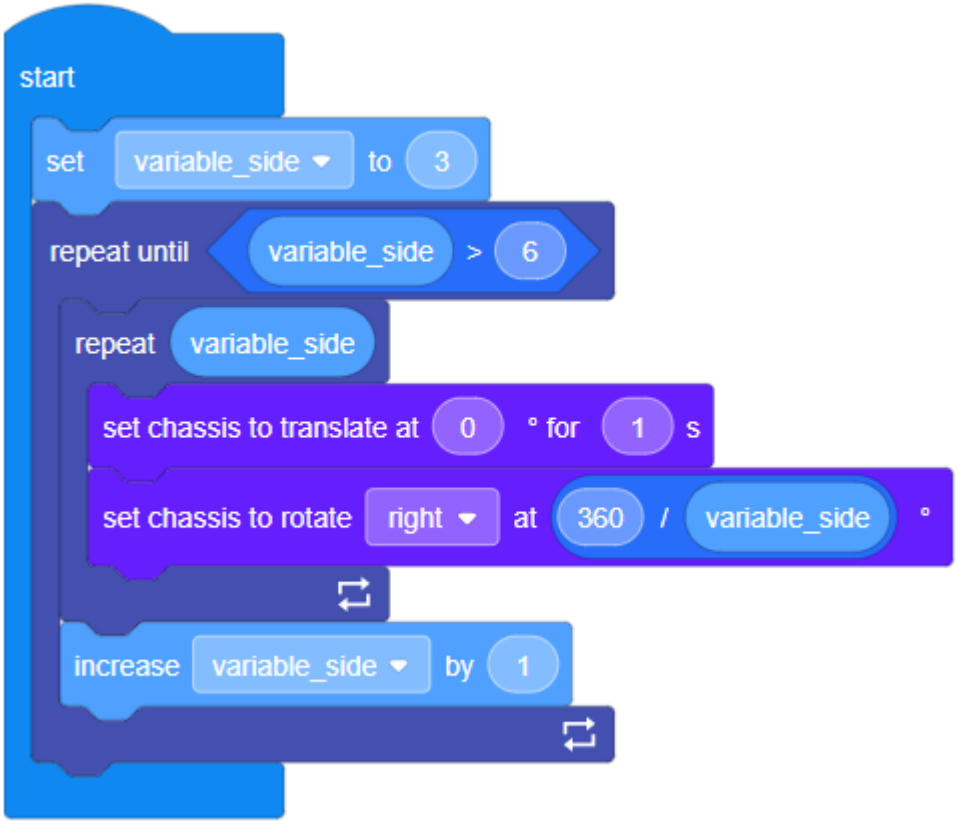


(1) Description: Repeat the internal program until the condition is met. That is: to run the internal program when the condition is not met; when the condition is met, it will jump out of the loop and execute the next command

(2) Type: Conditional statement block

(3) Example: Polygon positioning

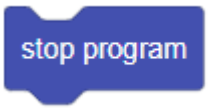
Set RoboMaster EP CORE to follow the tracks of regular triangle, square, regular pentagon and regular hexagon in sequence.



Note:

In order to make sure of the proper number of loop, consider it thoroughly when you are completing the judgment conditions.

7. Stop program

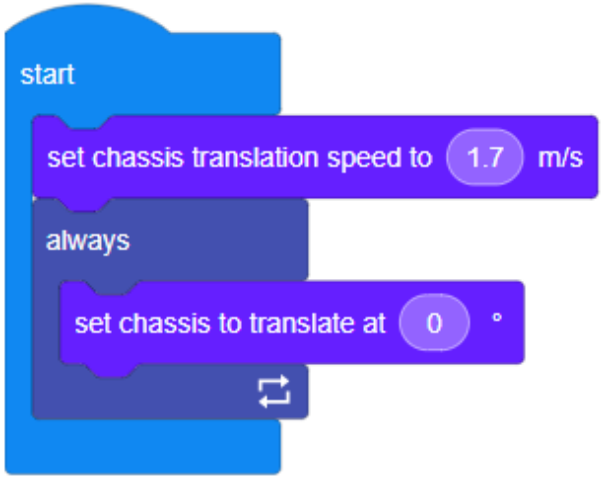


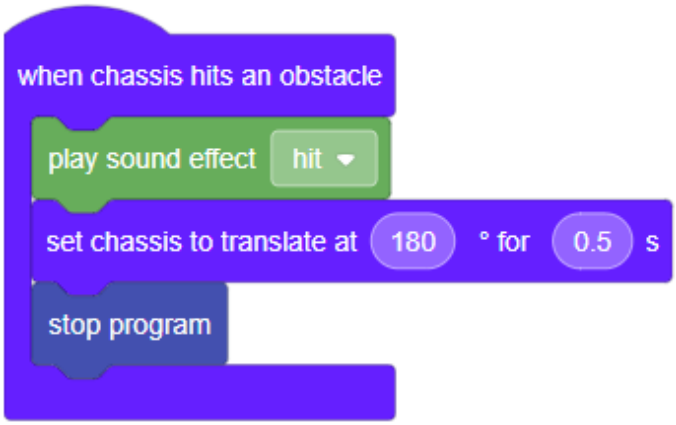
(1) Description: Stop a running program for the current block and exits

(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. (0)+(0)

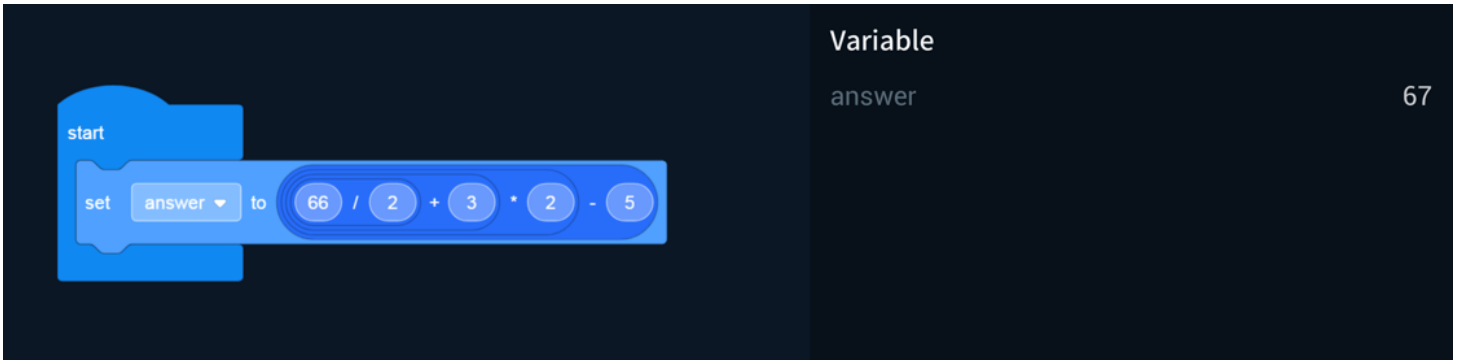
(1) Description: Add two numbers

(2) Return value: Variable-type data

(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.

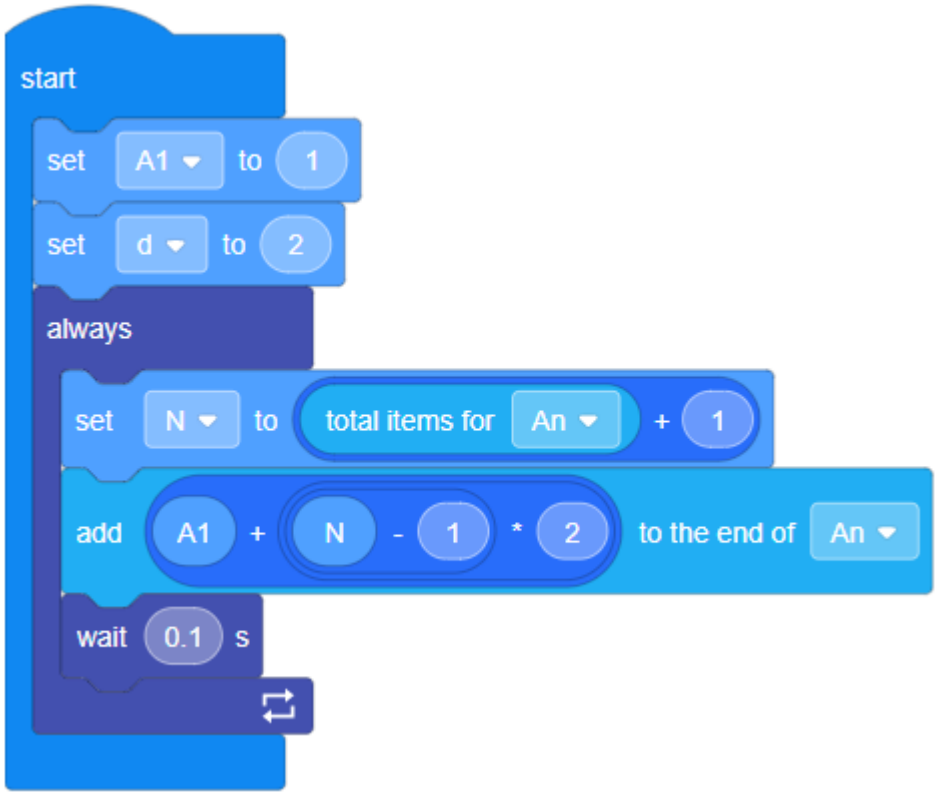
2. (0)-(0)

(1) Description: Subtract one number from another

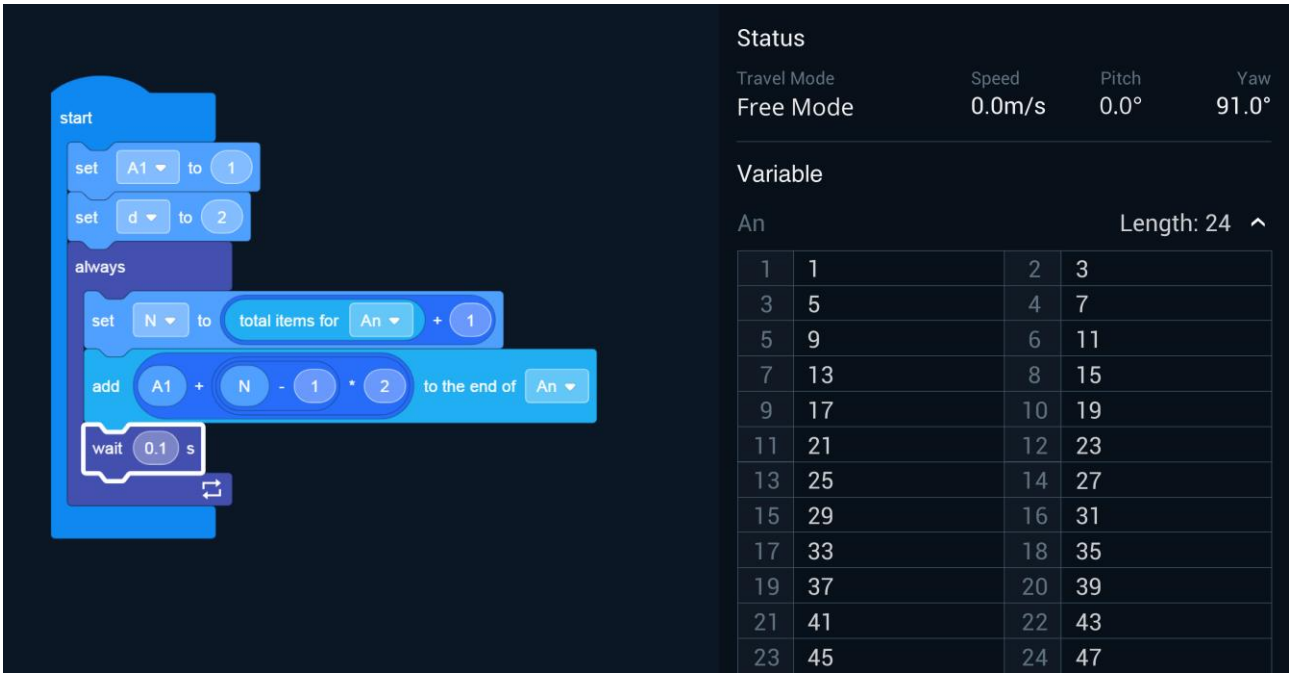
(2) Return value: Variable-type data

(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.



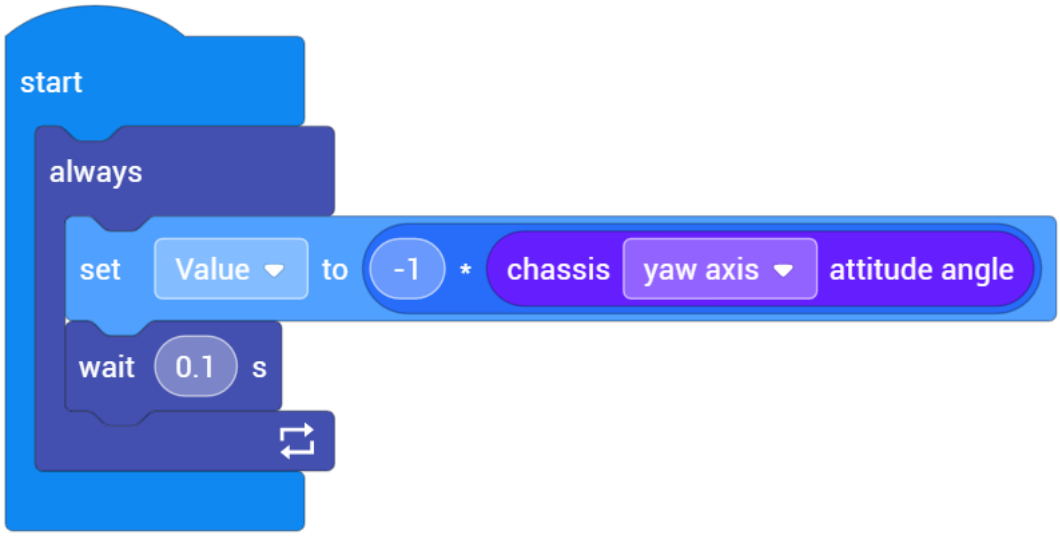
You can check the result using the FPV window:



Note:
Arithmetic calculations cannot be performed on lists.

3. (0)*(0)

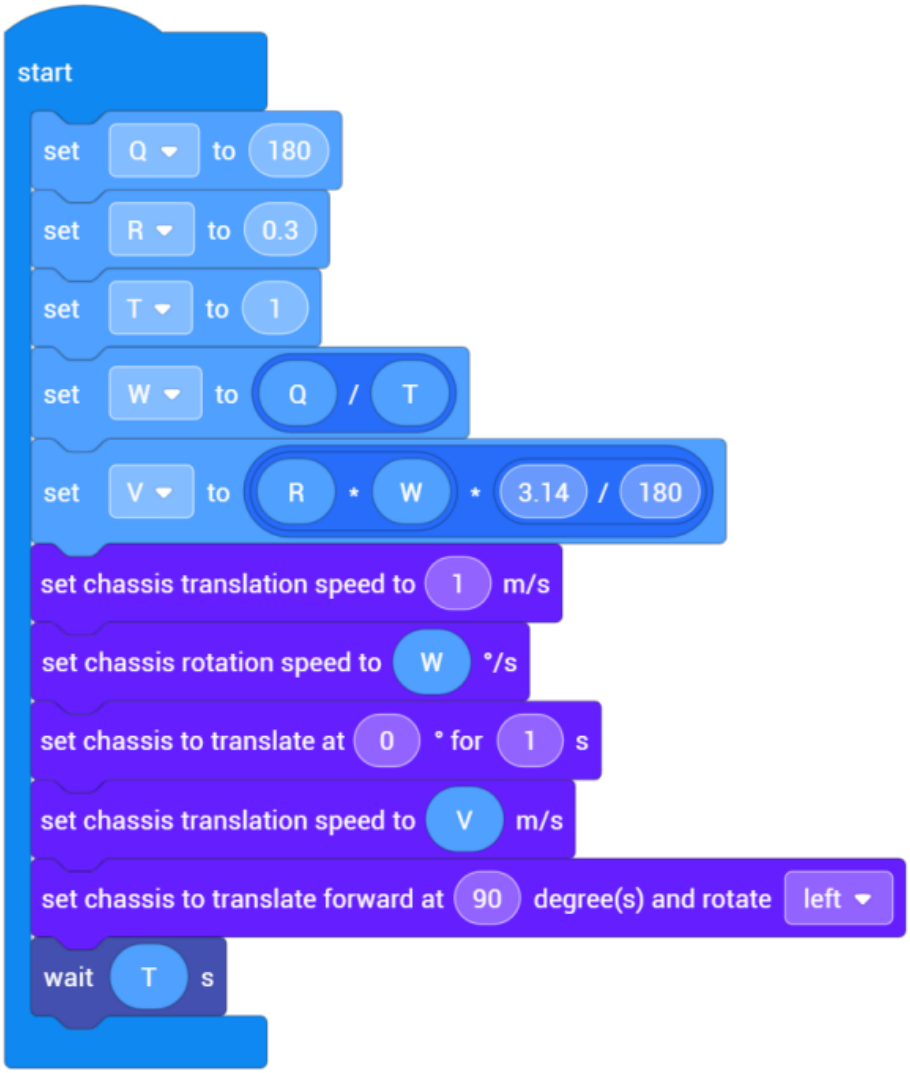
(1) Description: Multiply two numbers
(2) Return value: Variable-type data
(3) Example: Generate reverse display
This allows you to rotate the chassis manually and observe how the value changes in the FPV window.
Because this is the reverse display, when the gimbal is rotating on the right side of the 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.

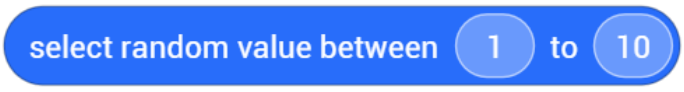
4. (0)/(0)

(1) Description: Divide one number by another
(2) Return value: Variable-type data
(3) Example: Drift
This will control the robot to drift and pull up.

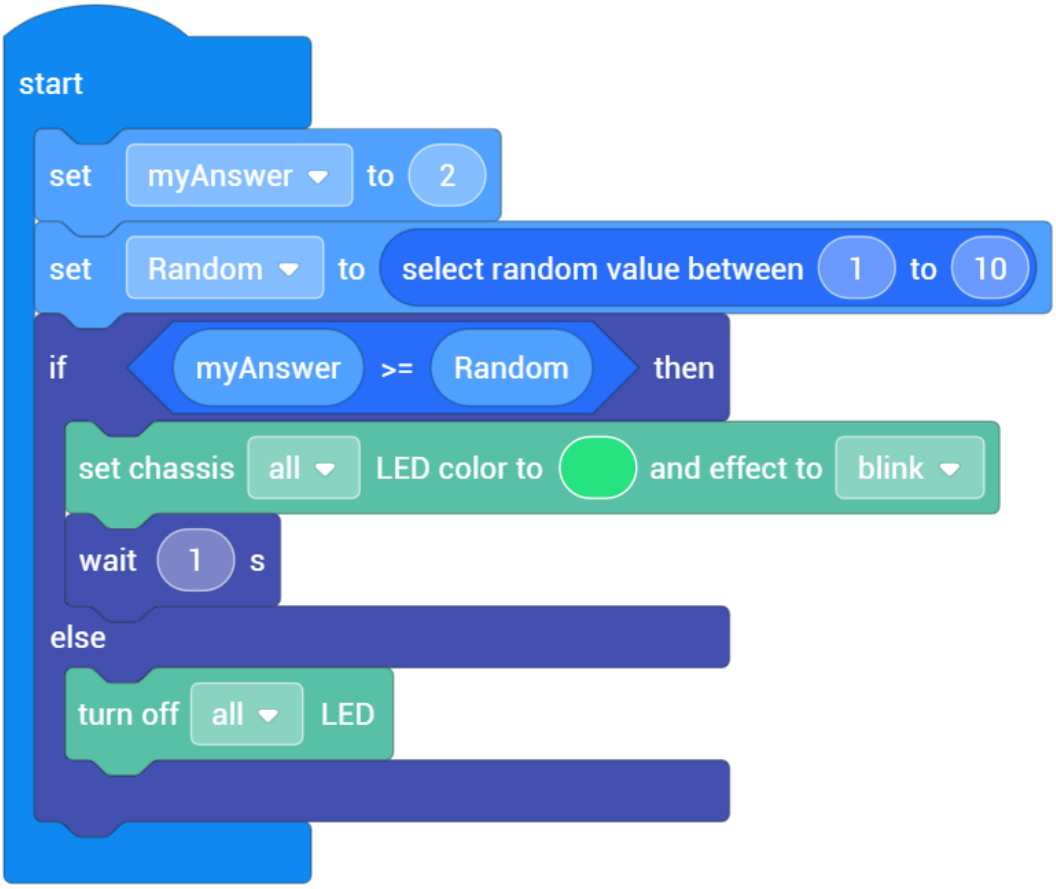


Note:
Arithmetic calculations cannot be performed on lists.

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



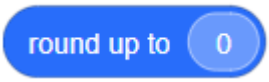
(1) Description: Select a random value from a specified range
(2) Type: Information block (variable-type data)
(3) Example: Play a number guessing game
Guess a number to enter and compare it to the random value generated by the robot. If your answer is greater than or equal to the robot's number, the chassis will flashing green light to indicate that the answer is correct; if your answer is less than the number, the chassis will turn off all LED lights.



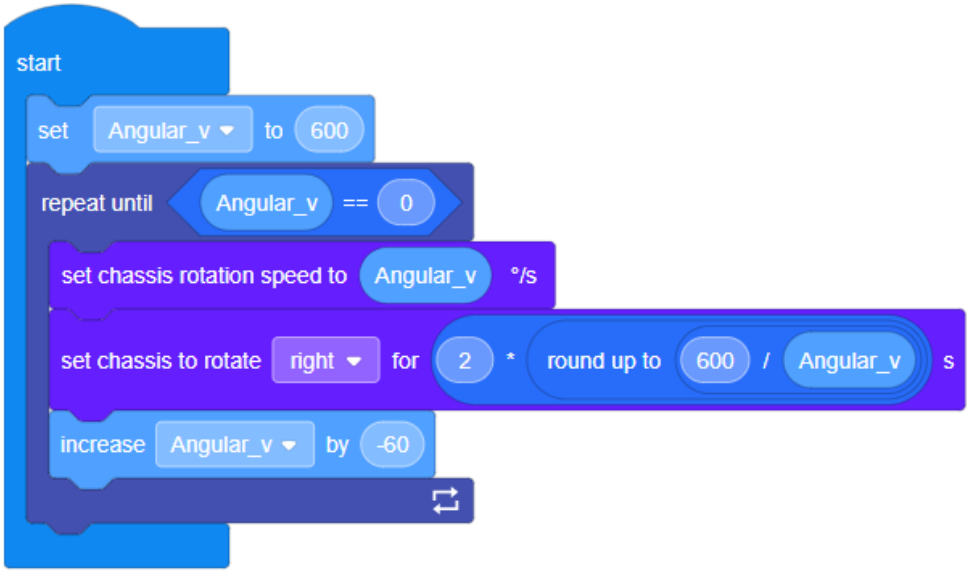
Note:

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



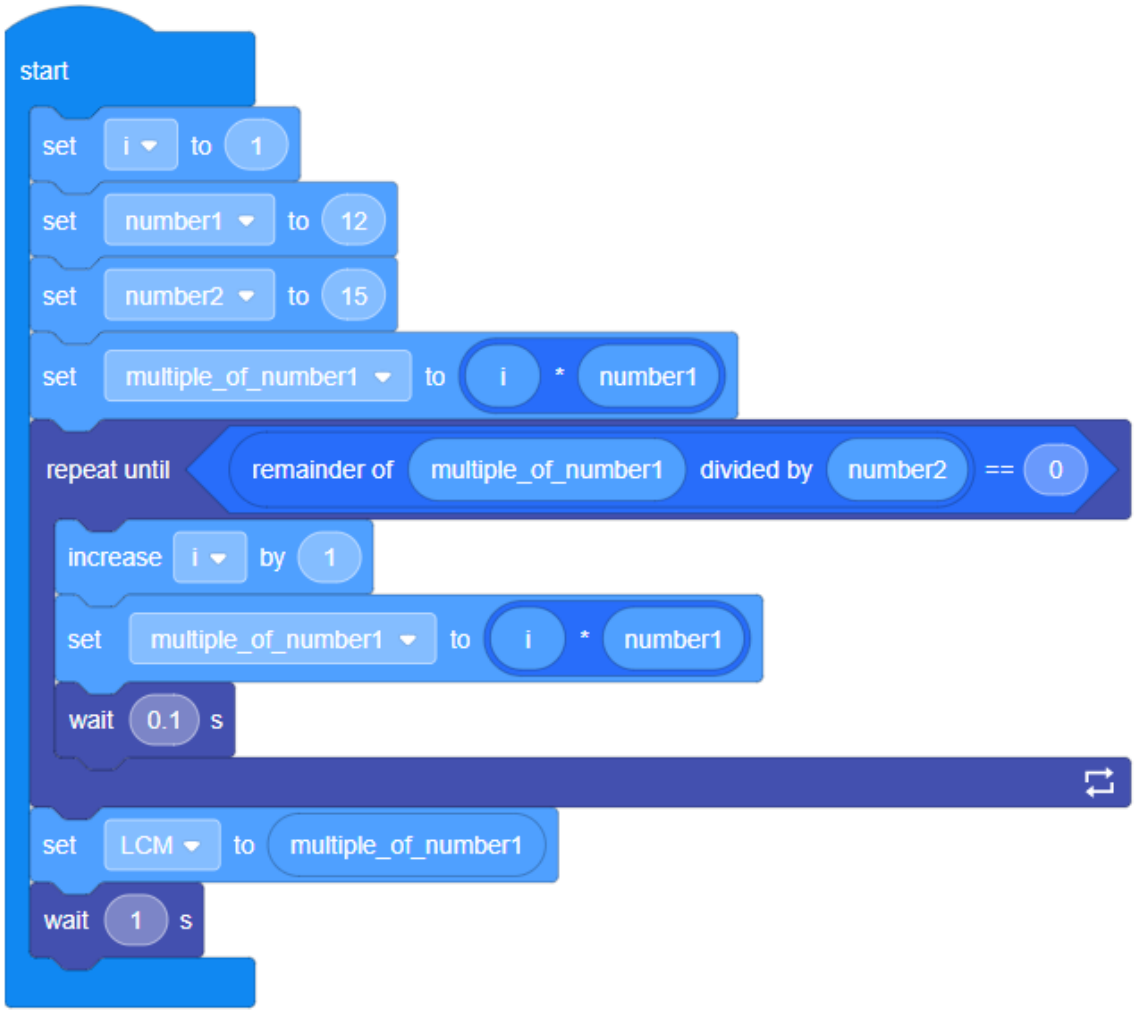
7. Remainder of (0) divided by (1)

(1) Description: Obtain the remainder of the first value divided by the second value

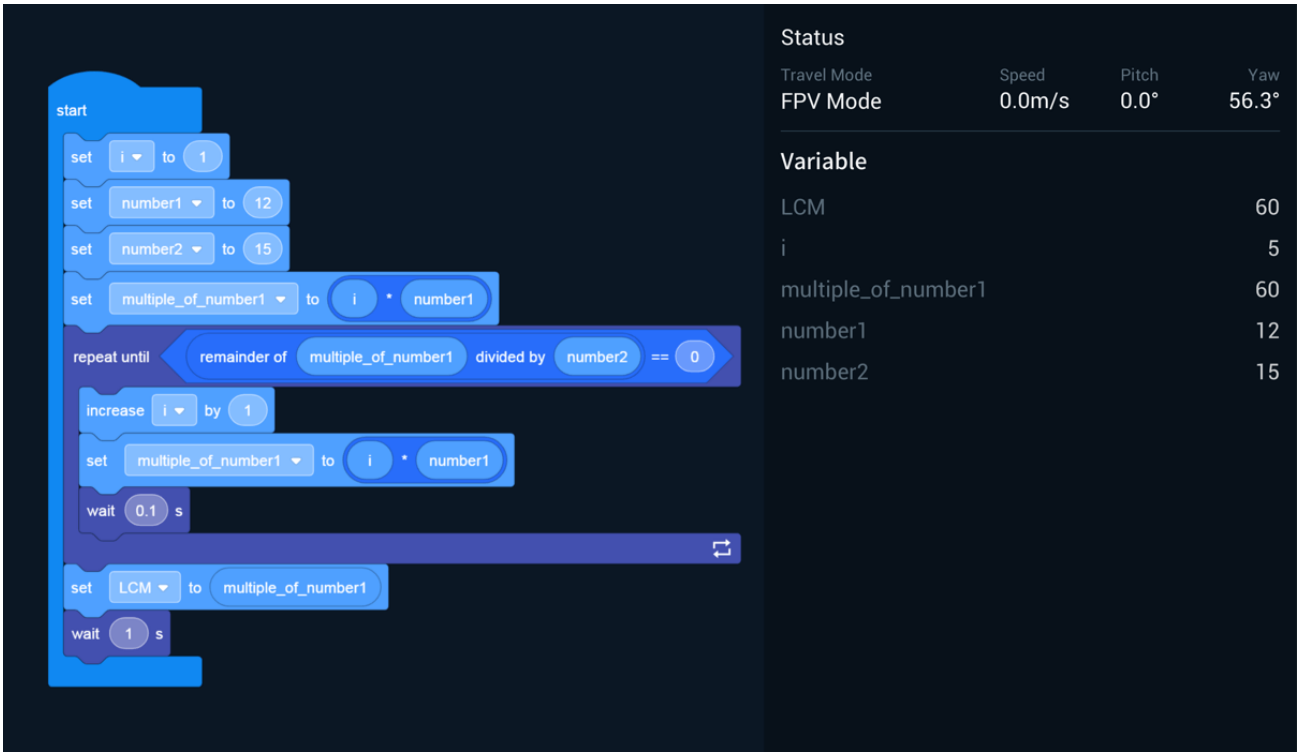
(2) Type: Information block (variable-type data)

(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:

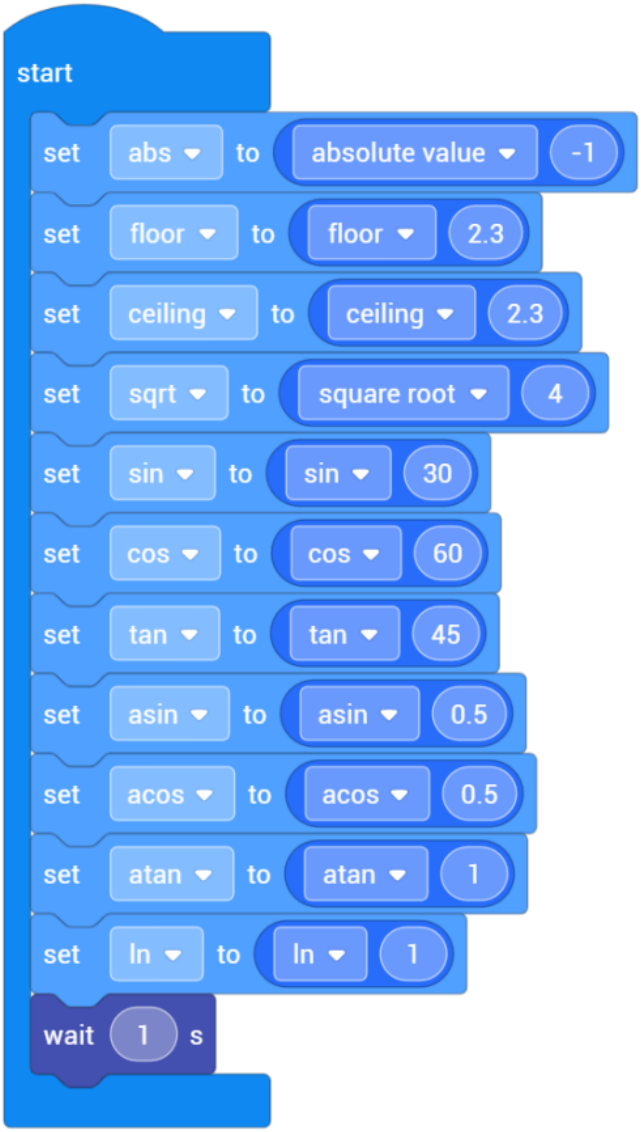
LCM is short for the lowest common multiple.

8. Absolute value (0)

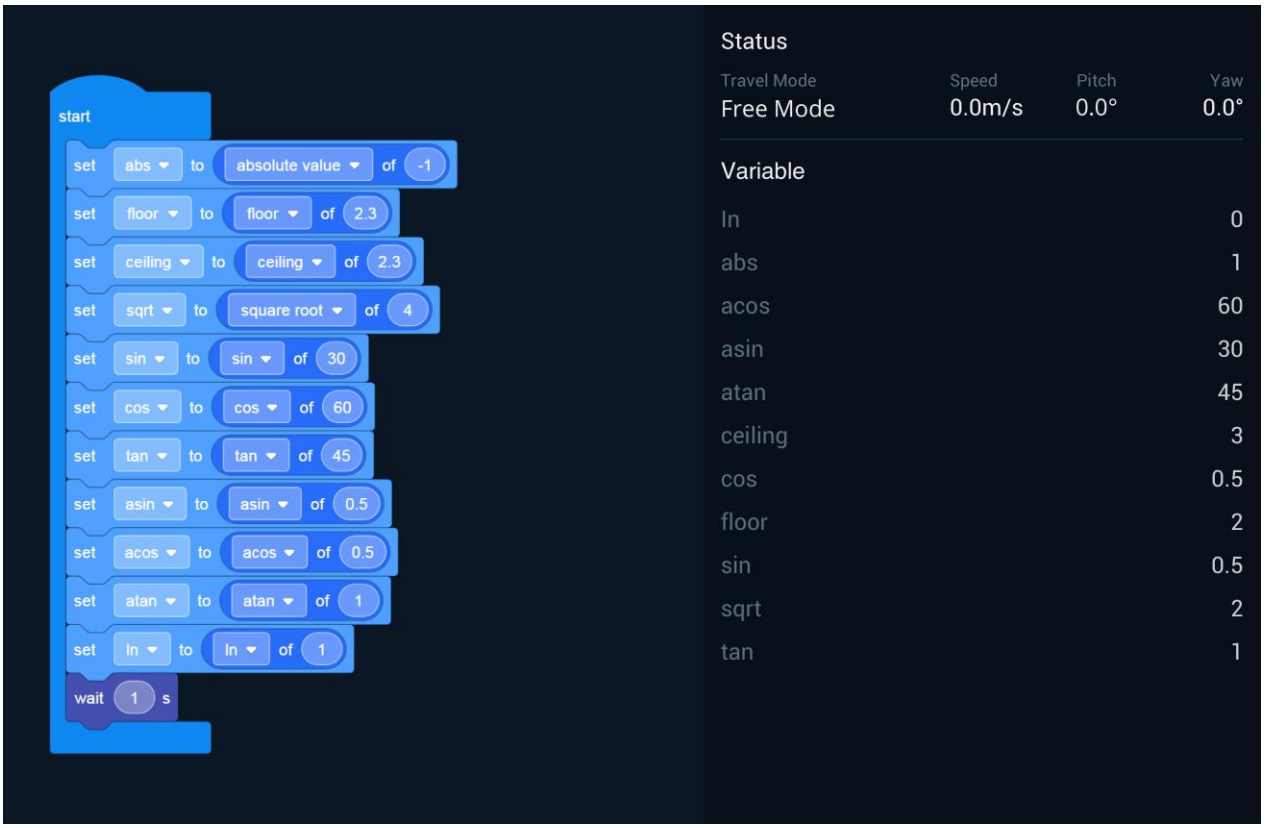
(1) Description: Perform mathematical operations for a variety of functions, such as calculating absolute value, floor and ceiling functions, square root, and sine angle

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

(3) Example: Mathematical calculation



You can check results using the FPV window.



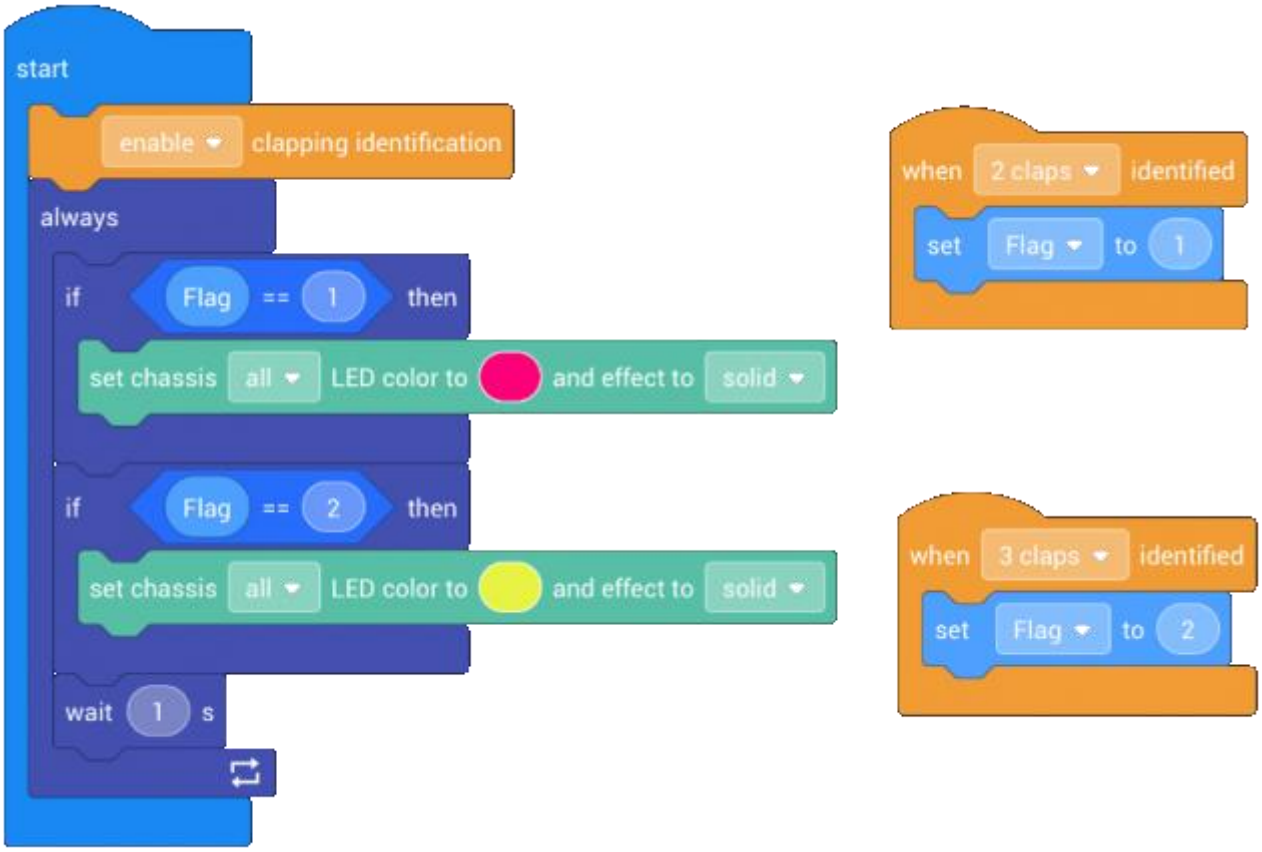
9. (0)==(0)

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

(2) Return value: Boolean

(3) Example: Set flag bits

Initially, the chassis displays a solid default color. All LEDs on the chassis 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.

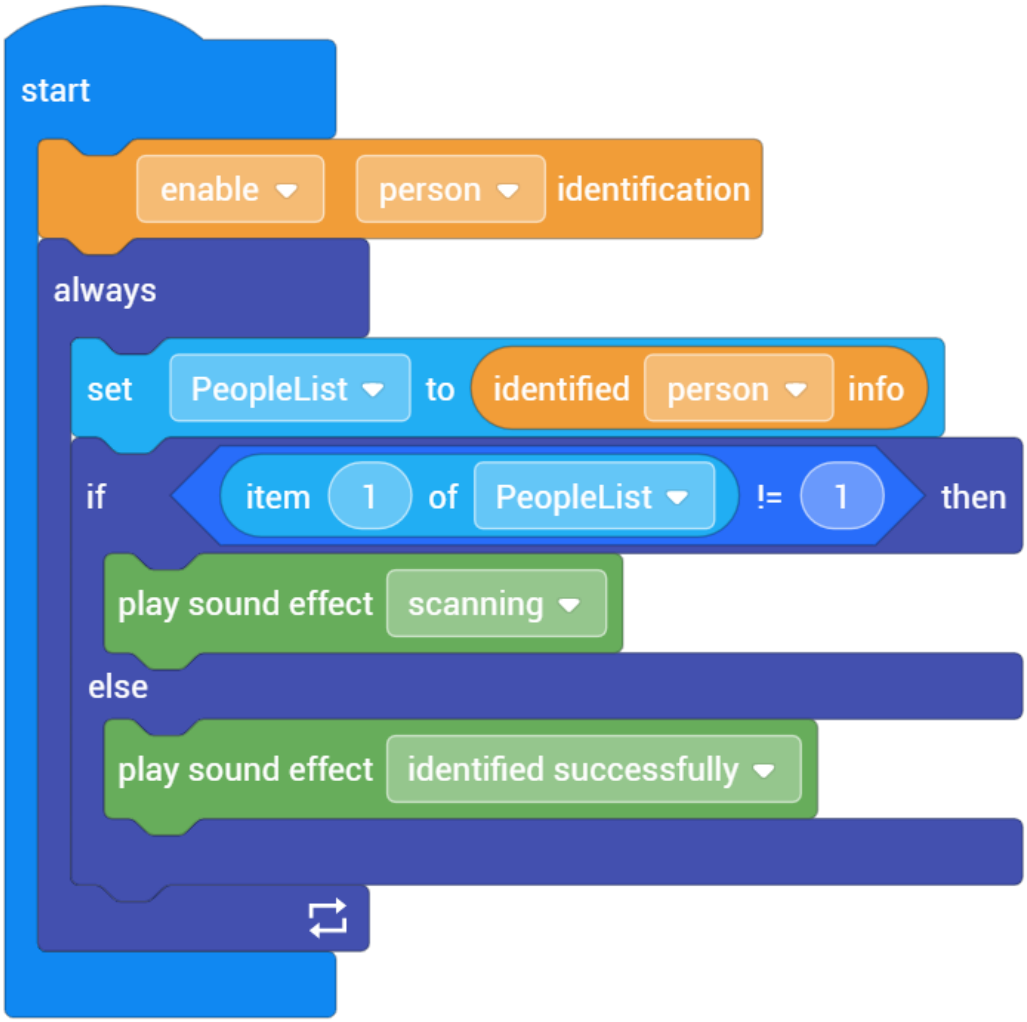
10. (0)!=(0)

(1) Description: Return “True” when two values are not equal; otherwise, “False” is returned

(2) Return 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.

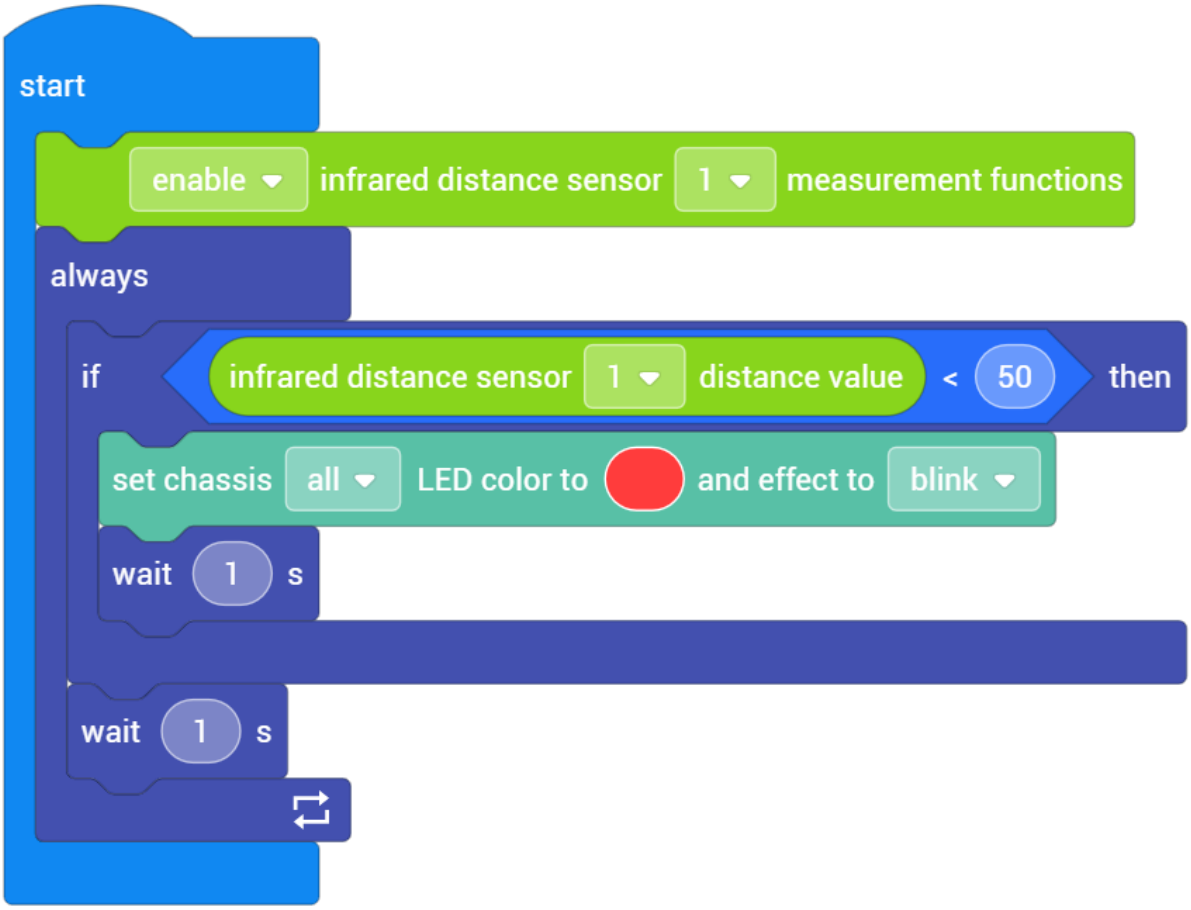
11. (0)<(0)

(1) Description: Return "True" if the first value is less than the second value; otherwise, "False" is returned.

(2) Return value: Boolean

(3) Example: Approaching warning

When an object enters the area within 50 cm around the robot, the robot chassis LED will blink red.



Note:

This is often used with conditional statements.

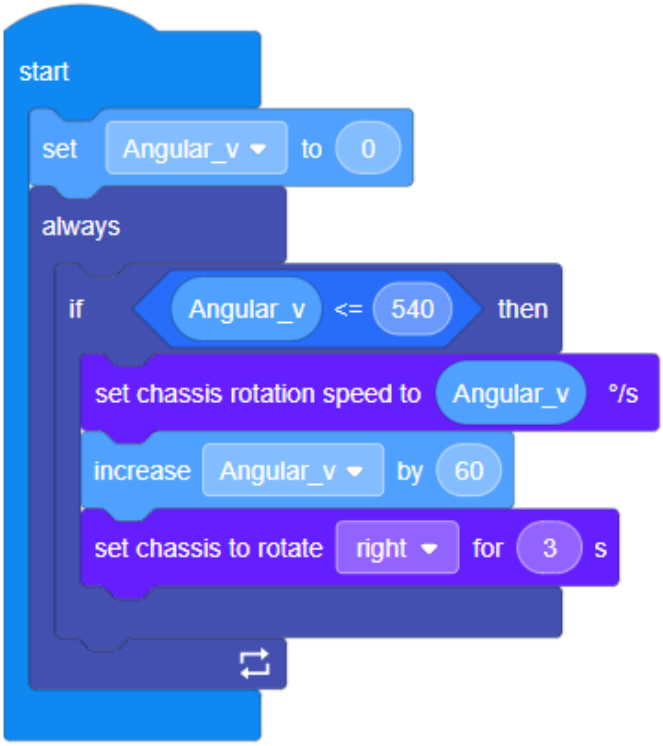
12. (0)<=(0)

(1) Description: Return the condition “True” if the first value is less than or equal to the second value; otherwise, “False” is returned.

(2) Return value: Boolean

(3) Example: Accelerate rotation

If the chassis is rotating at a rate less than or equal to 540 degrees/s, it will increase by 60 degrees/s per second while rotating to the right for 3 seconds at a time until the maximum value of 600 degrees/s is achieved and acceleration stops.



Note:

"<=" is the symbol for "less than or equal to" in programming language and has the same meaning as "≥" in mathematical language.

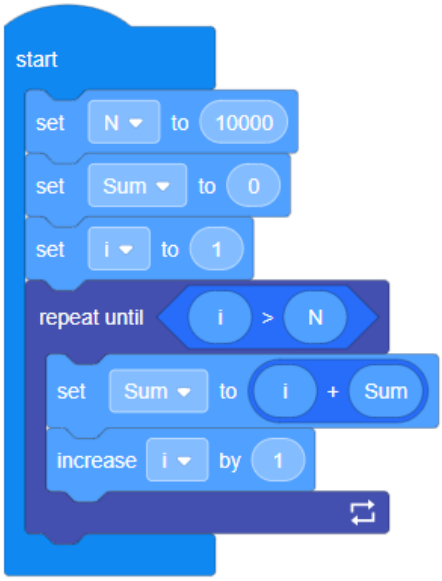
13. (0)>(0)

(1) Description: Return the condition "True" if the first value is larger than the second value; otherwise, "False" is returned.

(2) Return 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.)



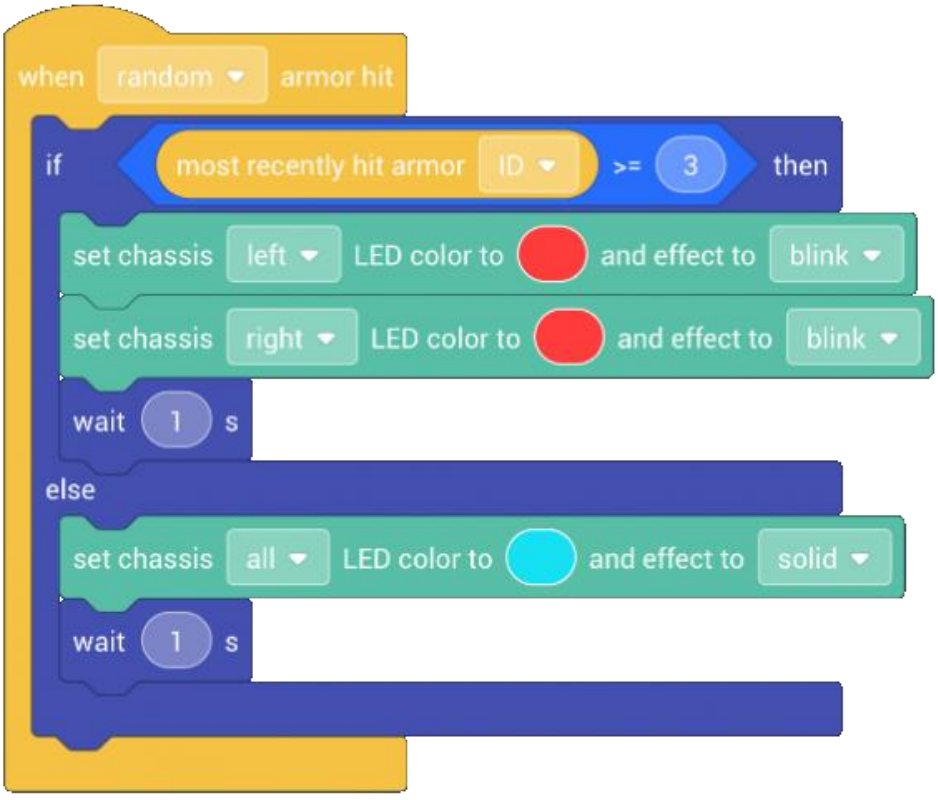
14. (0)>=(0)

(1) Description: Return the condition “True” if the first value is greater than or equal to the second value; otherwise, “False” is returned.

(2) Return value: Boolean

(3) Example: Partial response

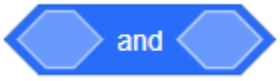
If the chassis front and rear armors were hit last, chassis front and rear LEDs will blink red; if the chassis armors were hit last on both sides, there is no response.



Note:

“>=” is the symbol for “greater than or equal to” in programming language and has the same meaning as “≤” in mathematical language.

15. ( ) and( )

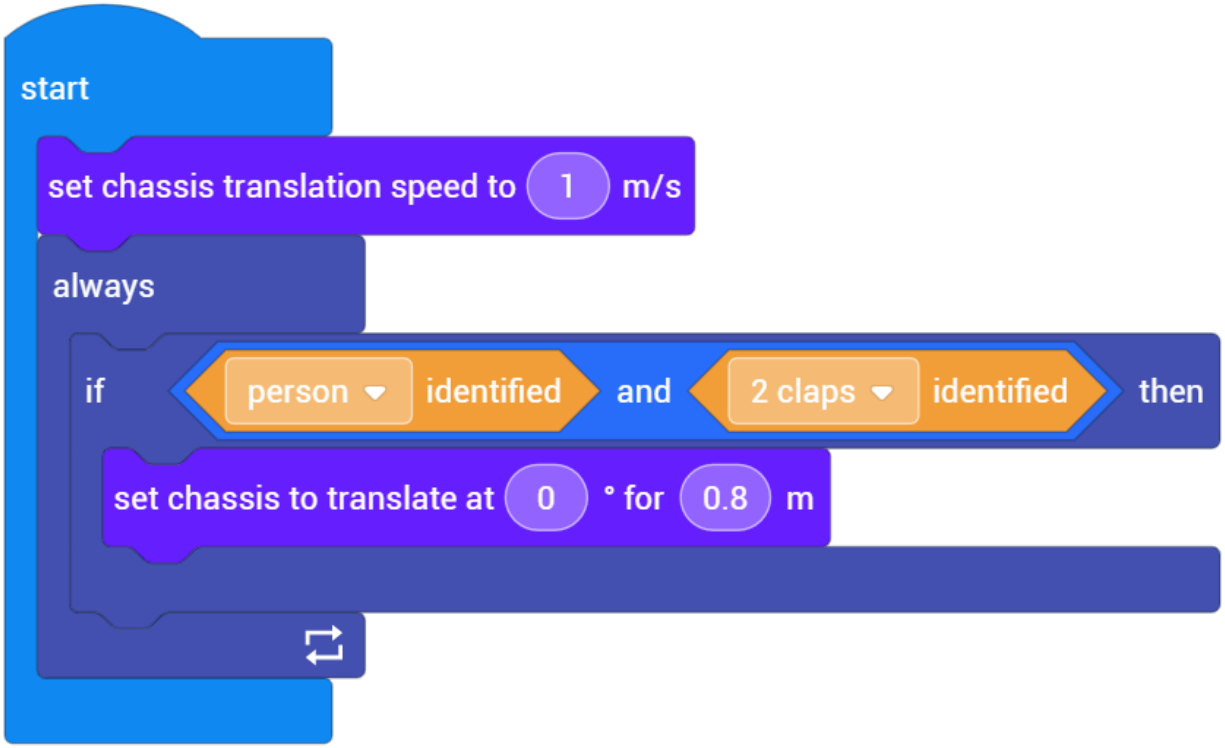


(1) Description: Return the condition “True” when two conditions are met; otherwise, the condition “False” is returned.

(2) Return 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.”

The “and” of Boolean truth list:

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

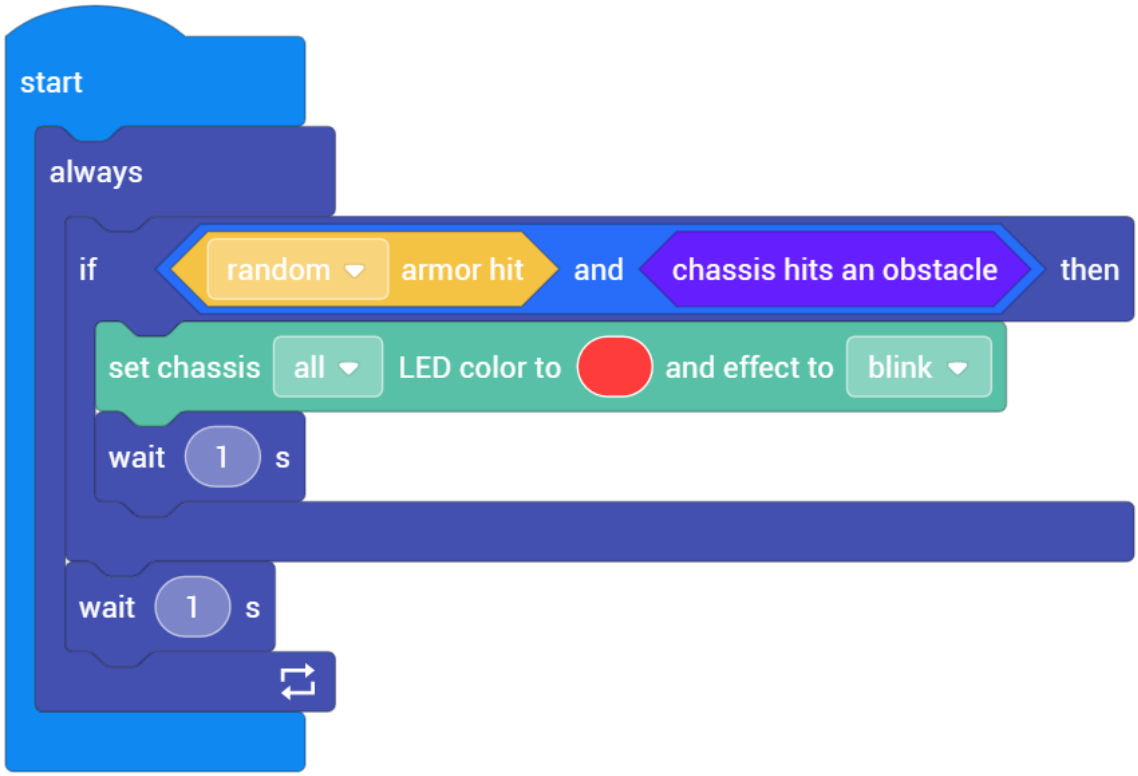
16. ( ) or ( )

(1) Description: Return “True” if either condition is met; or “False” if neither condition is met.

(2) Return value: Boolean

(3) Example: Emergency prompt

When the armor is attacked or the chassis hits an obstacle, all the LEDs on the chassis will blink red.



Note:

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

The “or” of Boolean truth list:

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

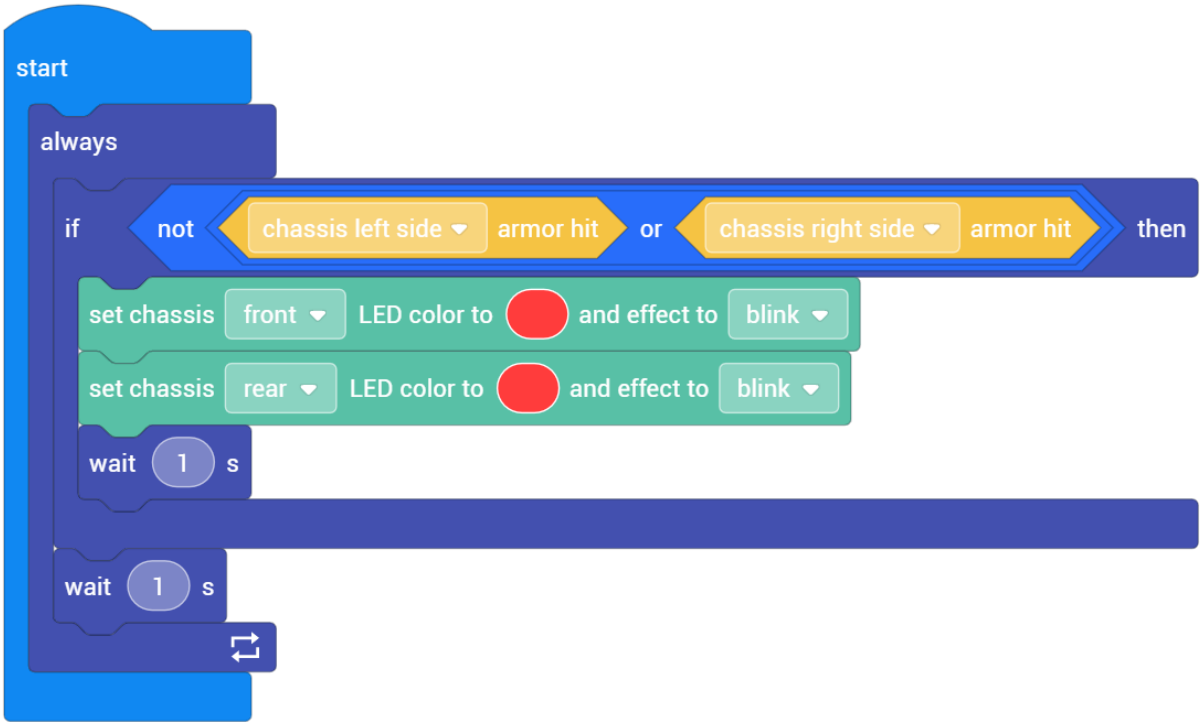
17. Not ( )

(1) Description: Take the opposite, i.e. return “False” if a condition is met; otherwise, “True” is returned.

(2) Return value: Boolean

(3) Example: Set either-or conditions

When the left and right armors are not attacked, it can be judged that the front and rear armors are attacked, and the front and rear armors blink red.



Note:

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

The “not” of Boolean truth list:

Not A		
A	Condition False	Condition True
Not A	TRUE	FALSE

18. Map (0) from low (0) high (1023) to low (0) high (4)

(1) Description: Scale the set value according to the proportional relationship.

(2) Type: Information block

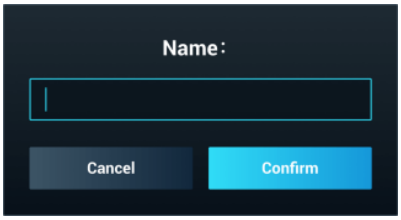
(3) Example: Ratio conversion

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.

Data Objects

Voltage

1. Create variable



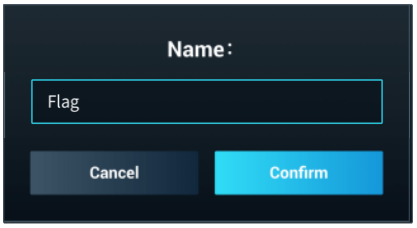
(1) Description: Create a name for a new variable.

(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.

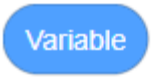


Variable names need to begin with an underscore or a letter, and can only contain numbers, uppercase and lowercase letters, and underscores.

Note:

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

2. #Variable#

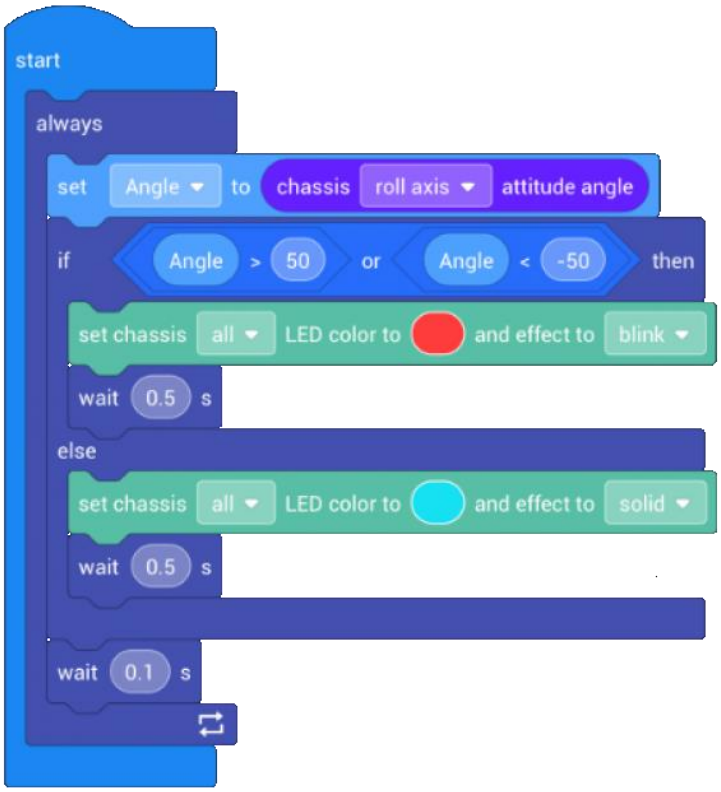


(1) Description: Obtain variable-type data

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

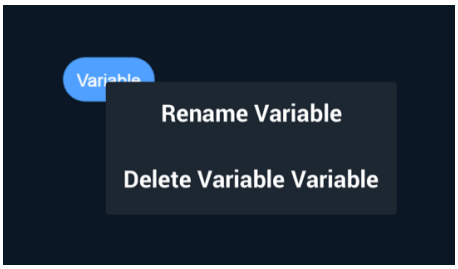
(3) Example: Risk warning

If the attitude angle of the roll axis of RoboMaster EP CORE chassis is greater than 50 or less than -50, the chassis LED will blink red to remind users that they should be careful of the risk of the robot rollover.

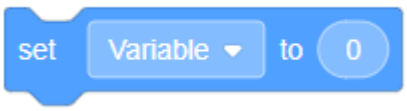


Note:

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



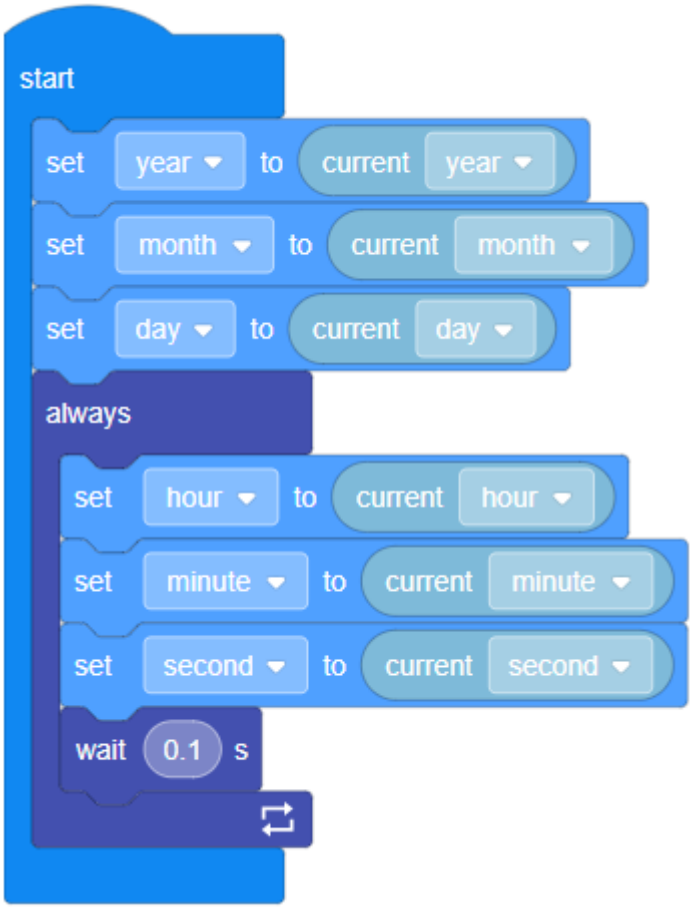
3. Set (# Variable#) to (0)



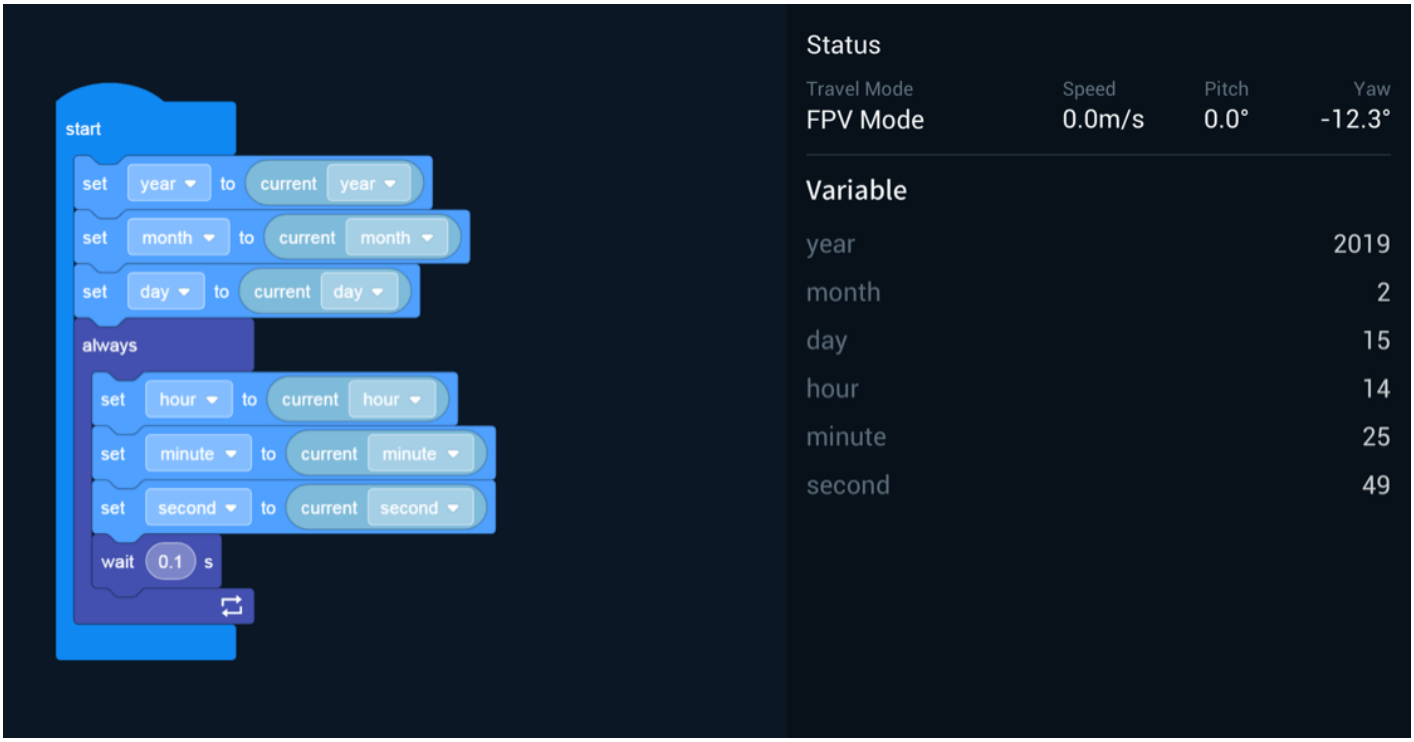
(1) Description: Assign value to a variable to save the input value for the variable

(2) Type: Execution block

(3) Example: Configure a digital clock

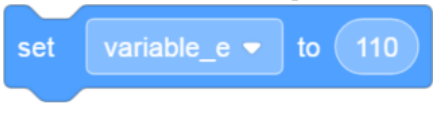


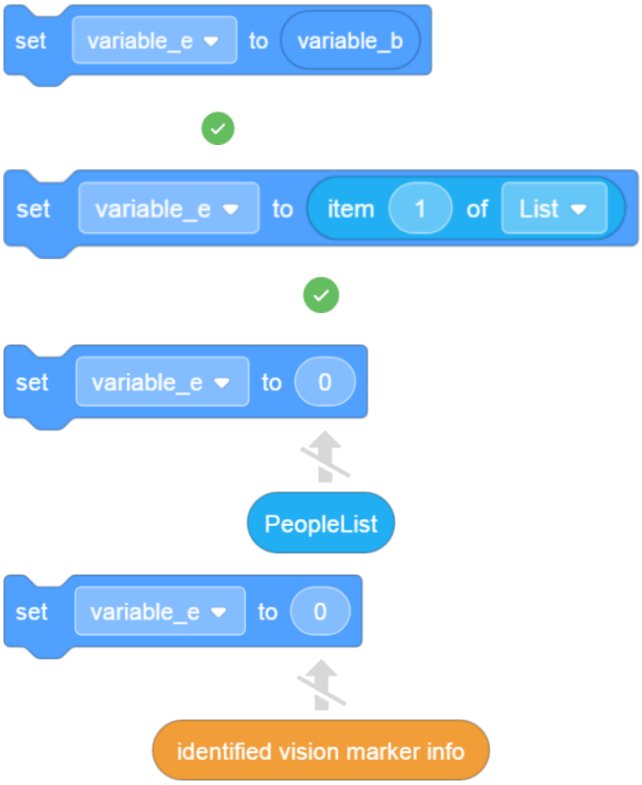
You can check details about the time using the FPV window. The hour, minute and second values will change continuously.



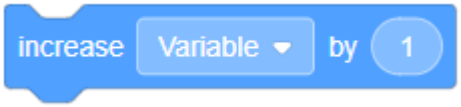
Notes:

1) Variables save one piece of data, and lists save a string of data, which cannot be mixed.
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)



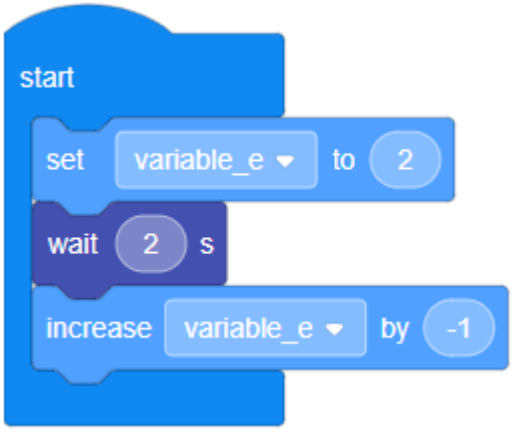
(1) Description: Change current variable value; positive values indicate an increase while negative values indicate a decrease

(2) Type: Execution block

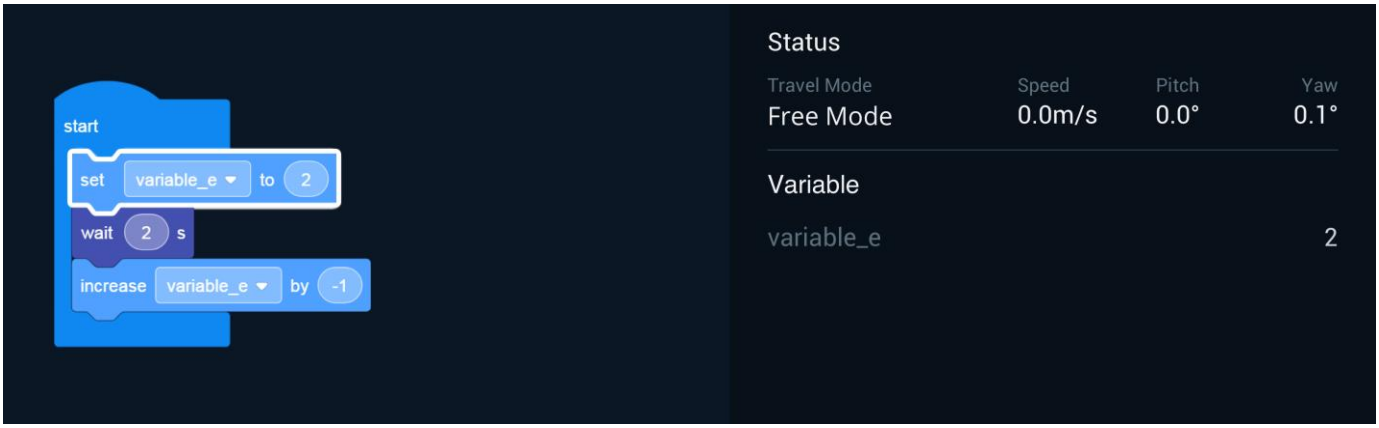
(3) Examples: Decrease variable value by 1, Translate in a figure-8 pattern

① Decrease variable value by 1

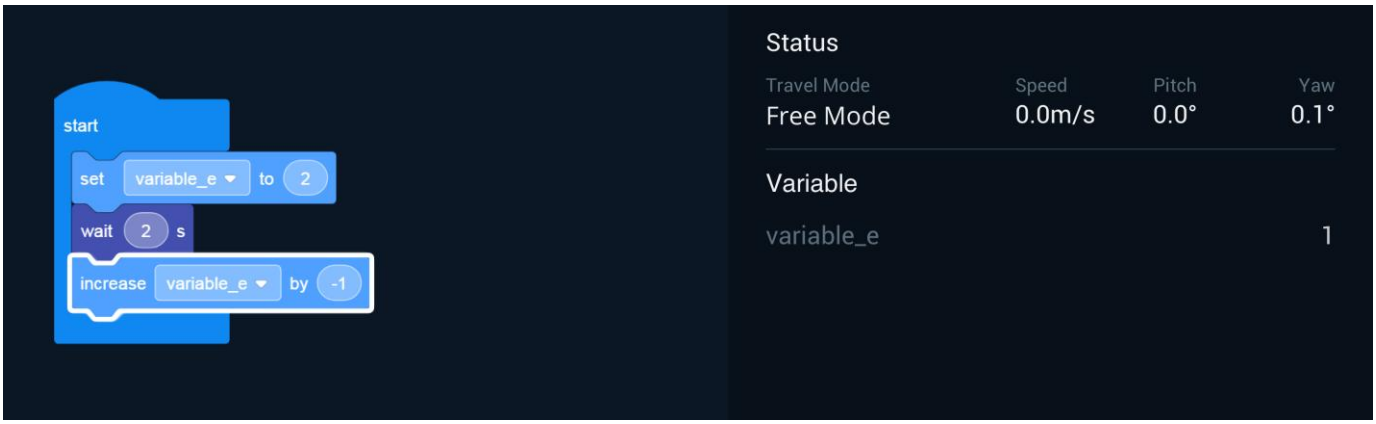
Assign a value to the variable, then configure the change in value:



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

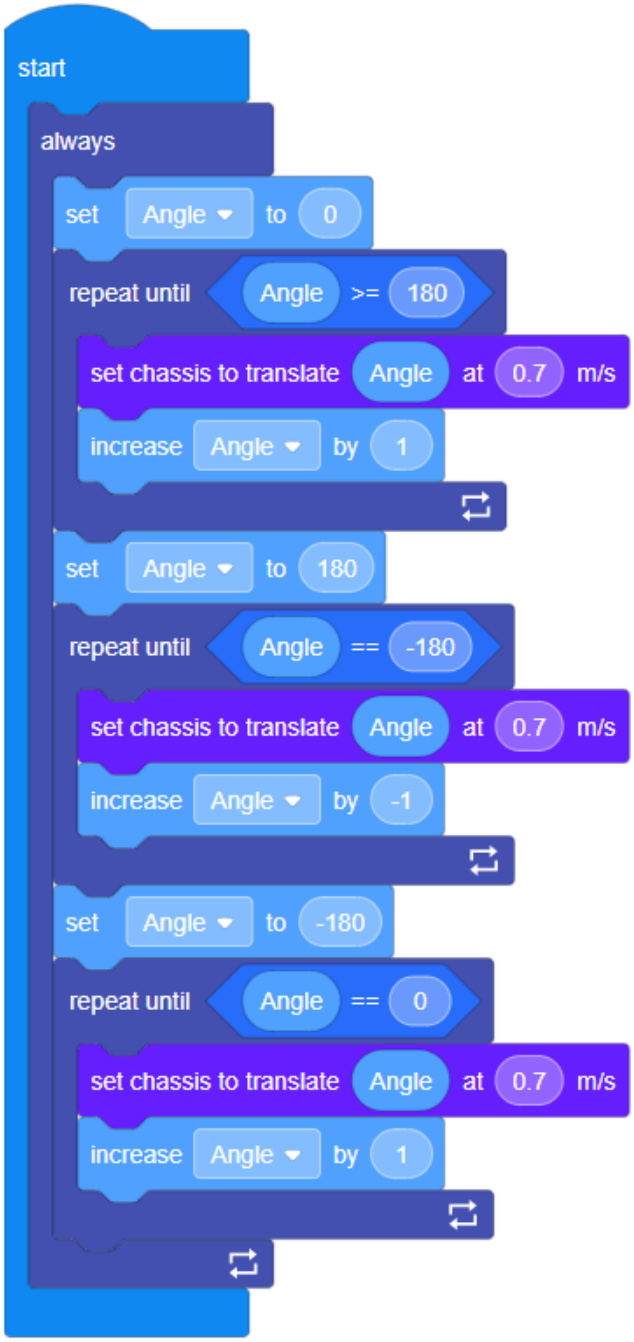


When you set the waiting time to 2 seconds and decrease the value by 1, the variable value of variable_e becomes 1.



② Translate in a figure-8 pattern

It is an alternative configuration for setting the robot to translate in figure-8 pattern.

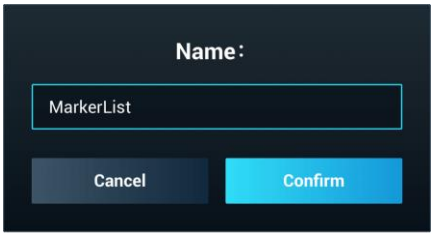


5. Create list

(1) Description: Create and names a list

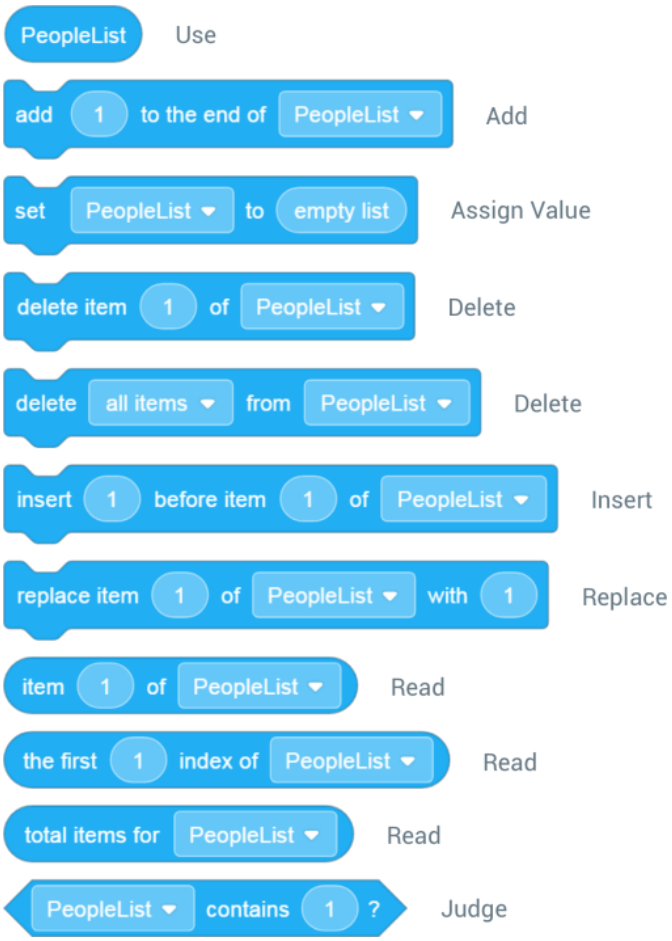
(2) Type: Settings block

(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 the list is created, several related blocks will appear, which are responsible for adjust, assign, adding and deleting the list etc.

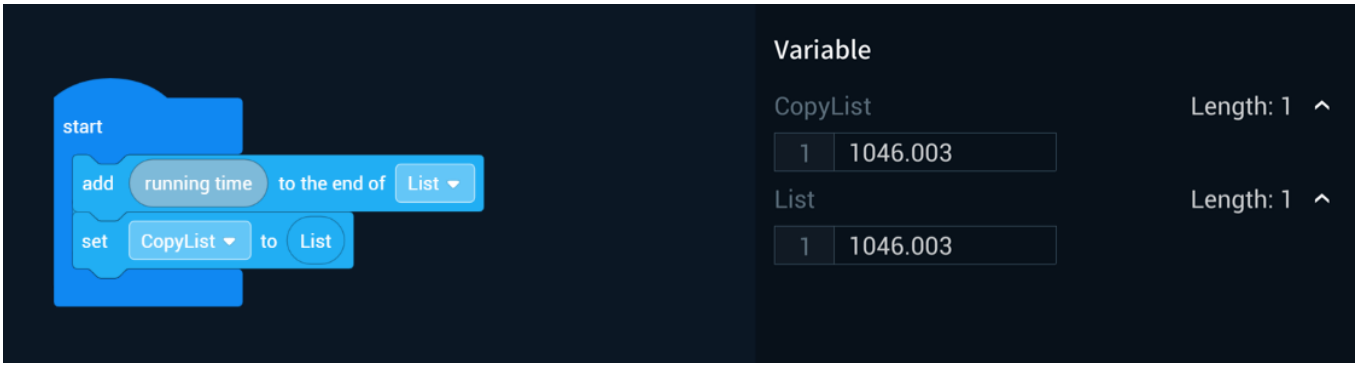


6. #List#

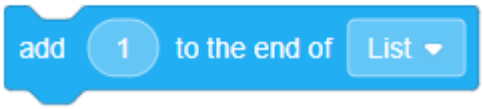
(1) Description: Obtain all items in a list
(2) Type: Information block (list)
(3) Example: Duplicate a list

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 details using the FPV window:



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

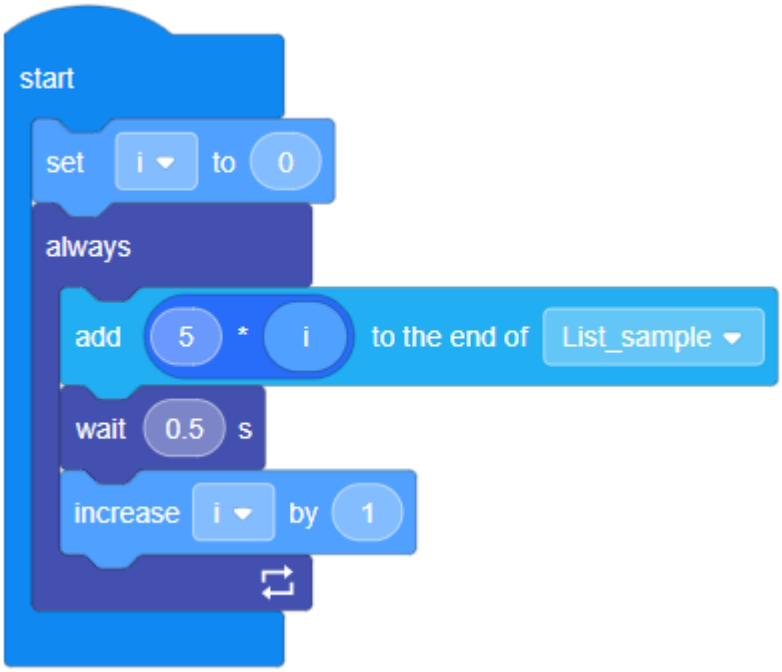


(1) Description: Add an item to the end of a list

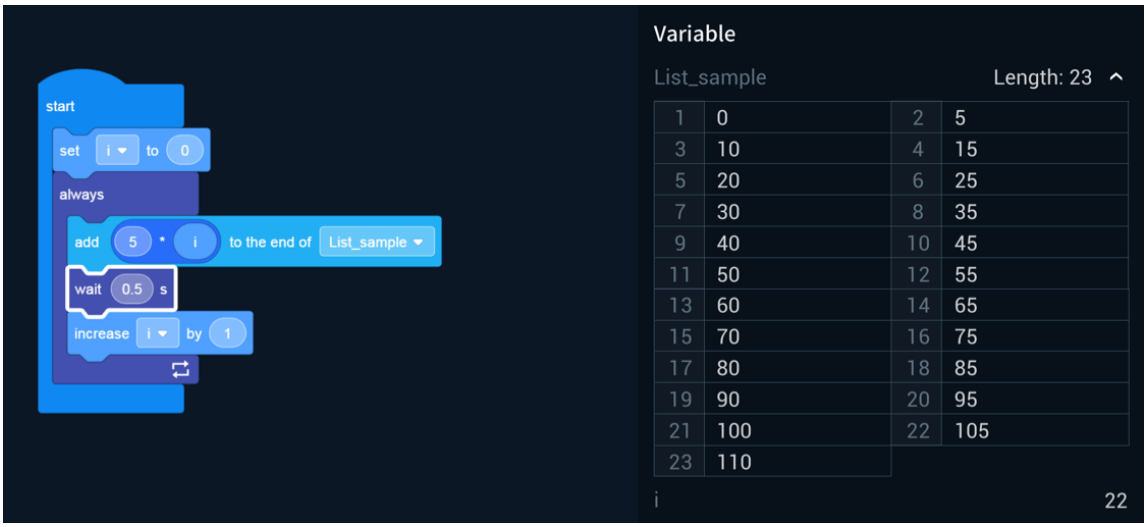
(2) Type: Execution block

(3) Example: Display multiples of 5

The configuration below allows you to obtain and display all multiples (≤0) of 5.



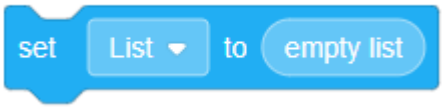
In the FPV window, a multiple of 5 is added to the list every 0.5 seconds.



Note:

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

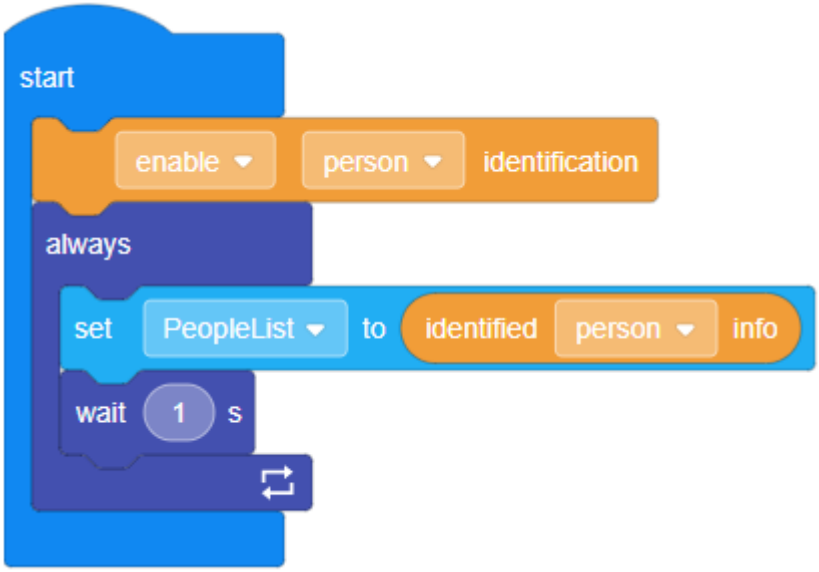
8. Set (#List#) to (empty list)



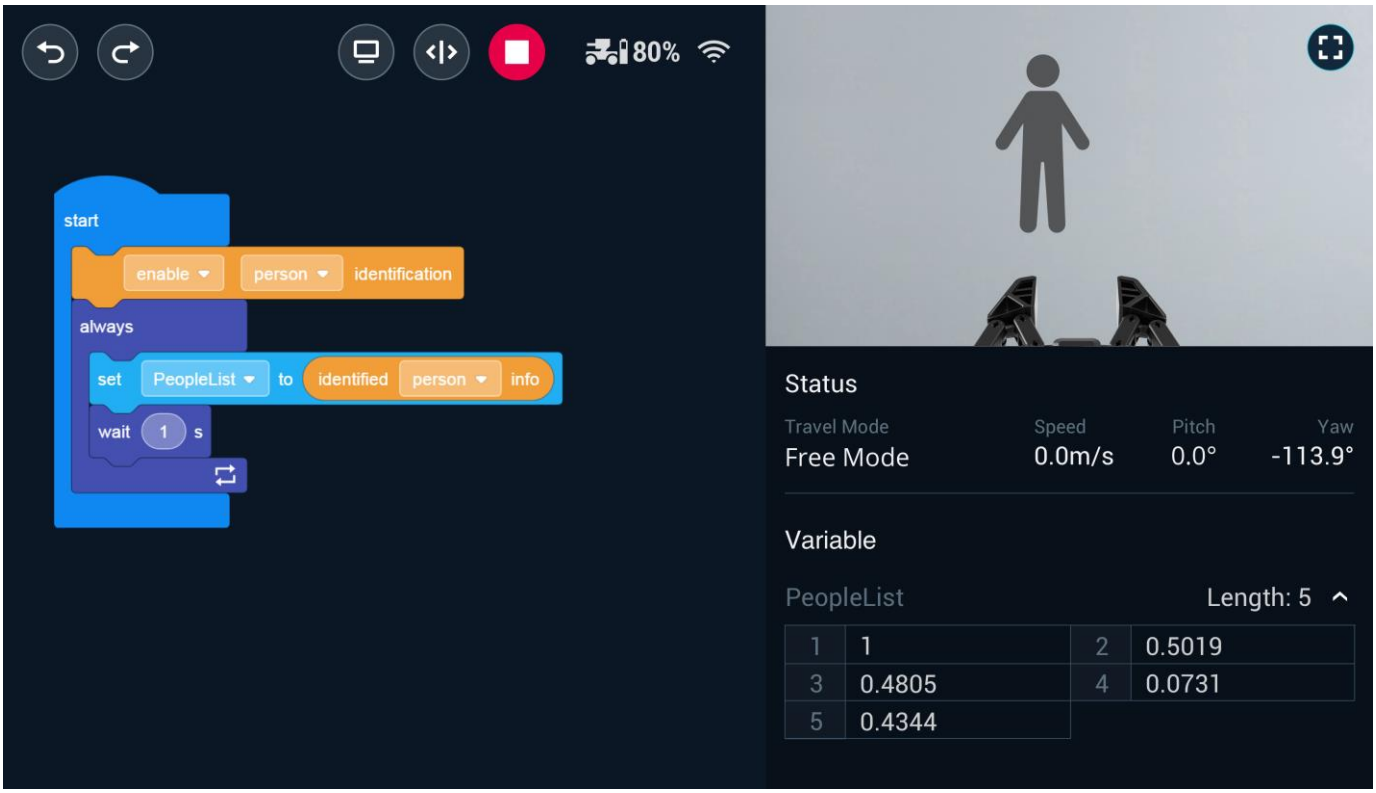
(1) Description: Assign values to a list

(2) Type: Execution block

(3) Example: Recognize a person



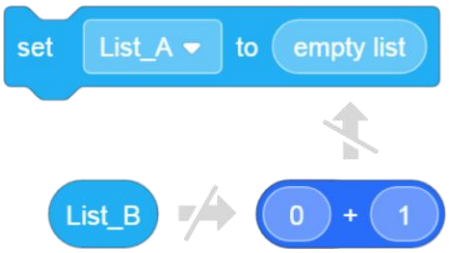
In the FPV window, the list is configured to recognize a person and generate relevant information:



Notes:

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.

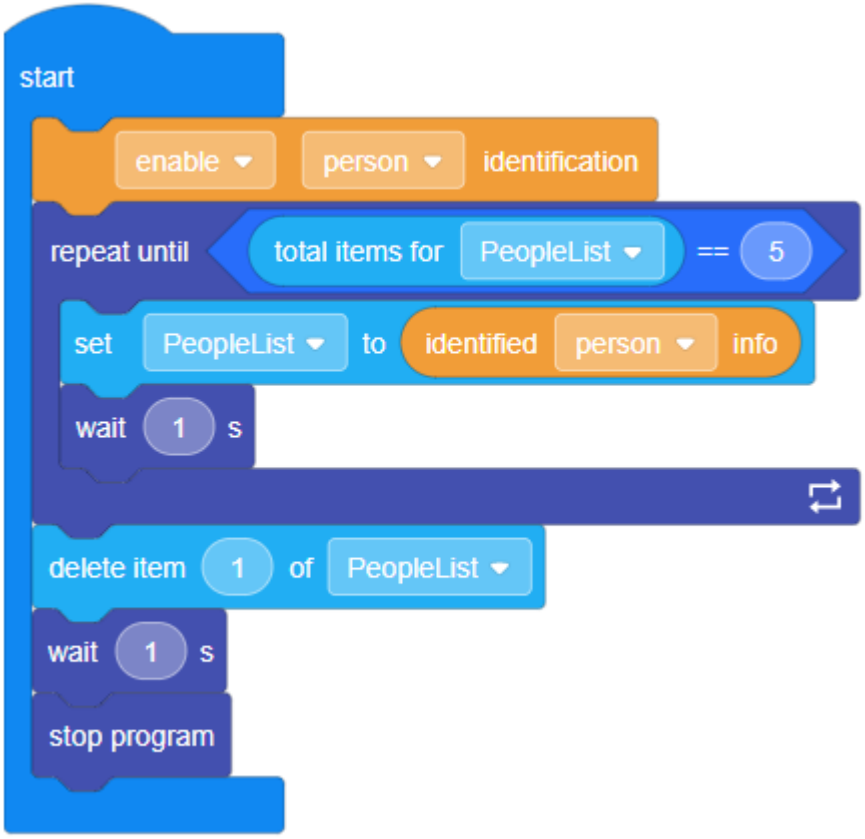


9. Delete item (1) of (#List#)

(1) Description: Delete an item from a list

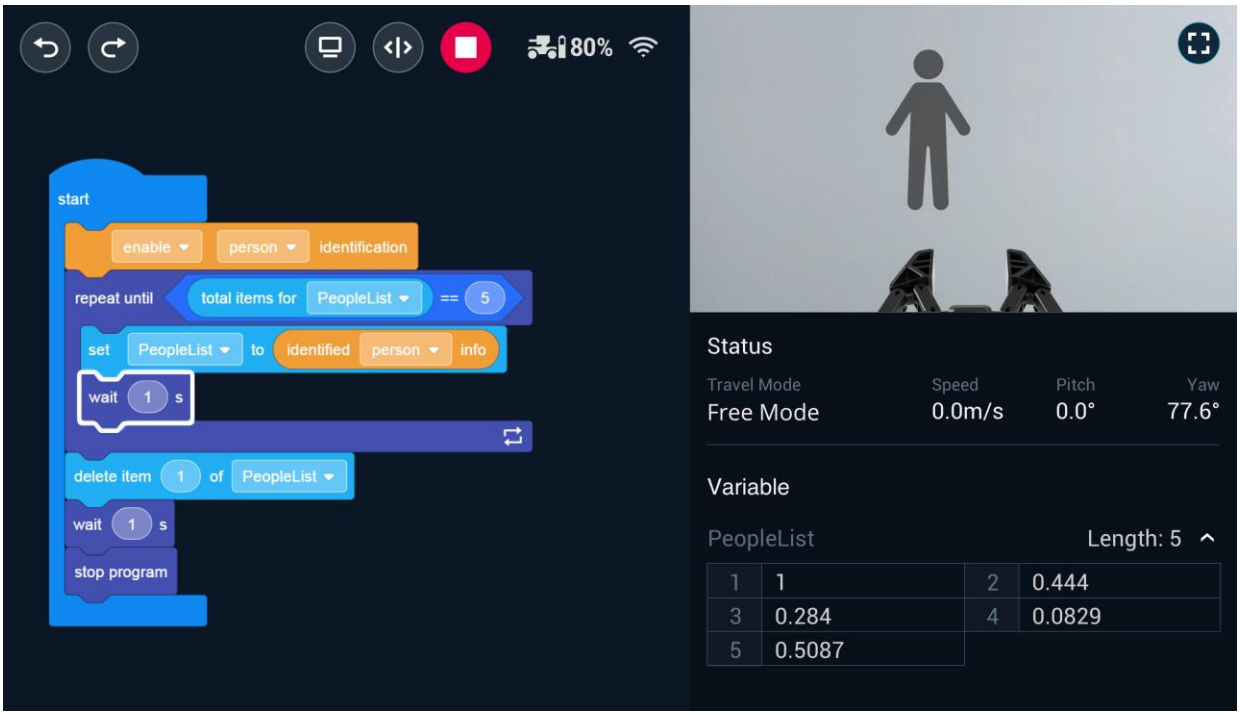
(2) Type: Execution block

(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:

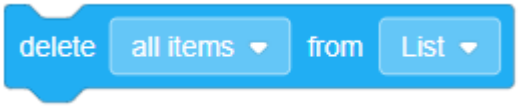


After:

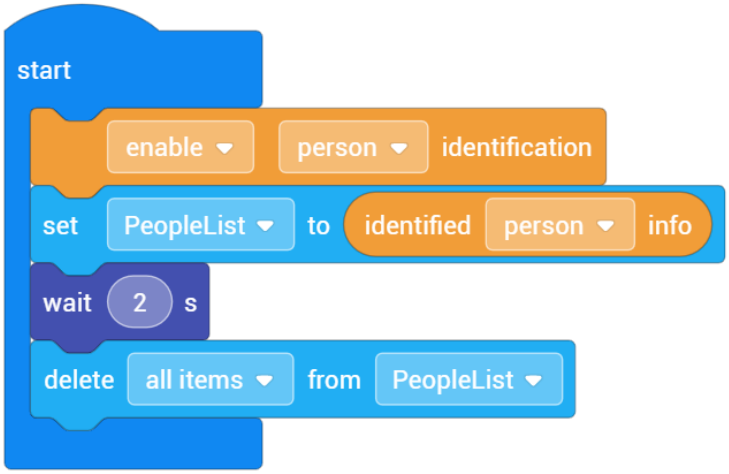
Notes:

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 by 1. All following items will come up by 1, of which the serial number reduces by 1 accordingly.

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

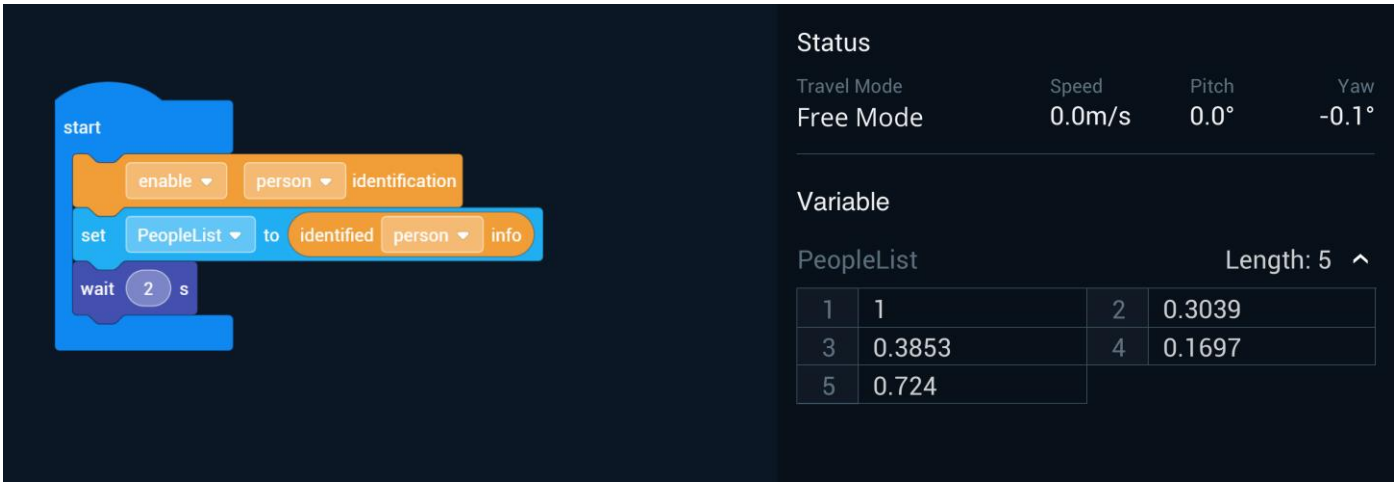


(1) Description: Delete all items or the last item from a list
(2) Type: Execution block
(3) Example: Clear list



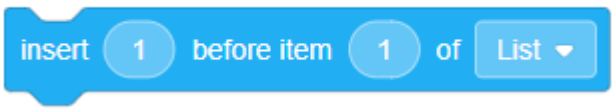
In the FPV window, the list length changes from 5 to 0.

Before:



After:

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

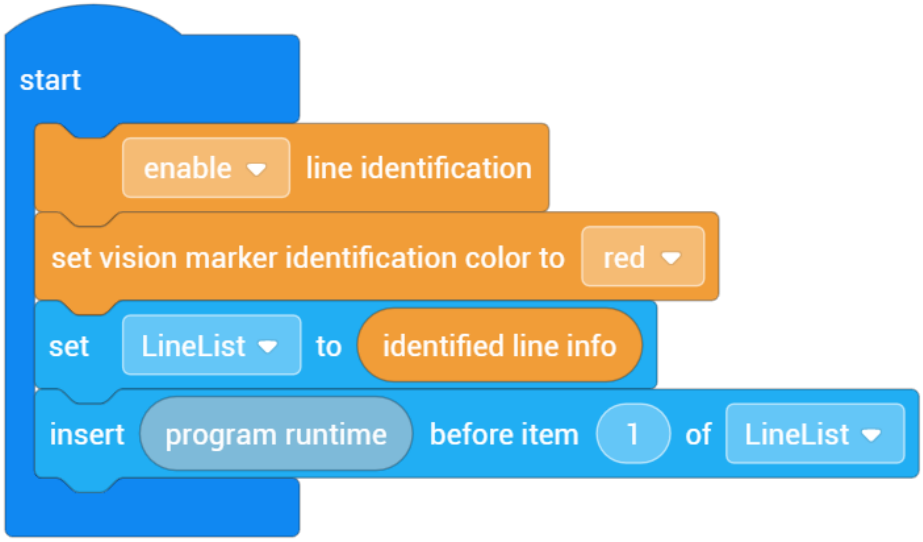


(1) Description: Insert an item at a specific location in a list and shifts subsequent items down the list

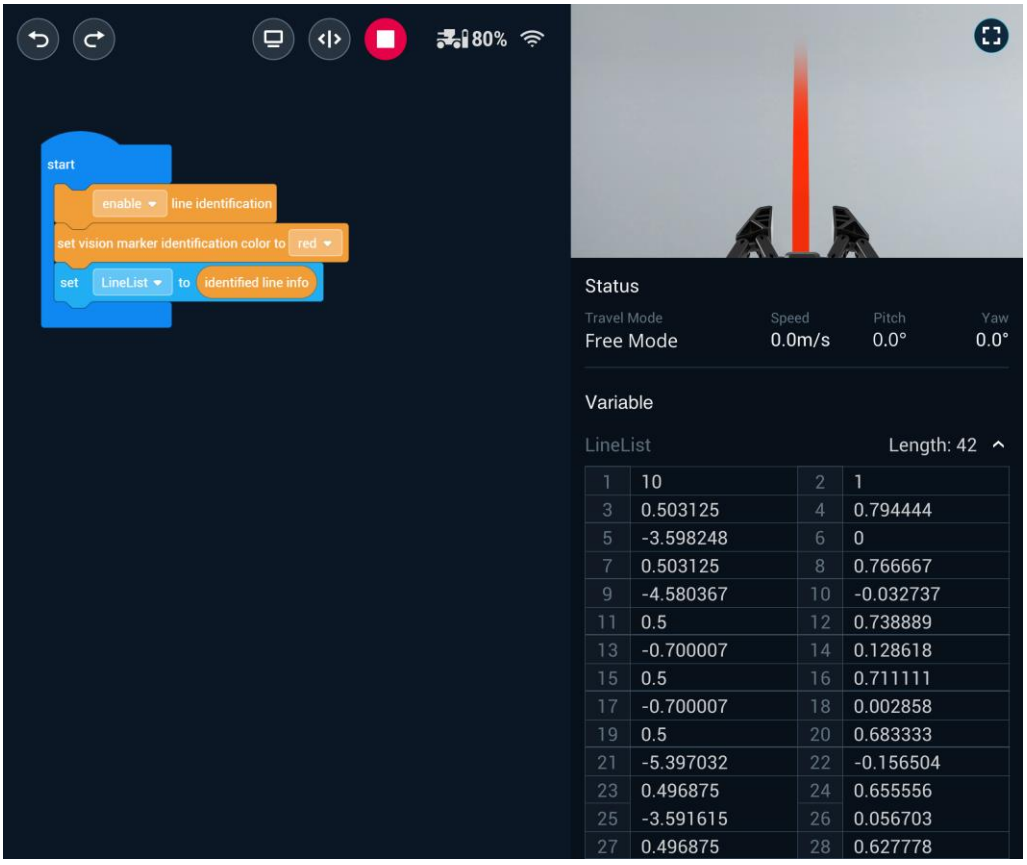
(2) Type: Execution block

(3) Example: Insert program runtime timer

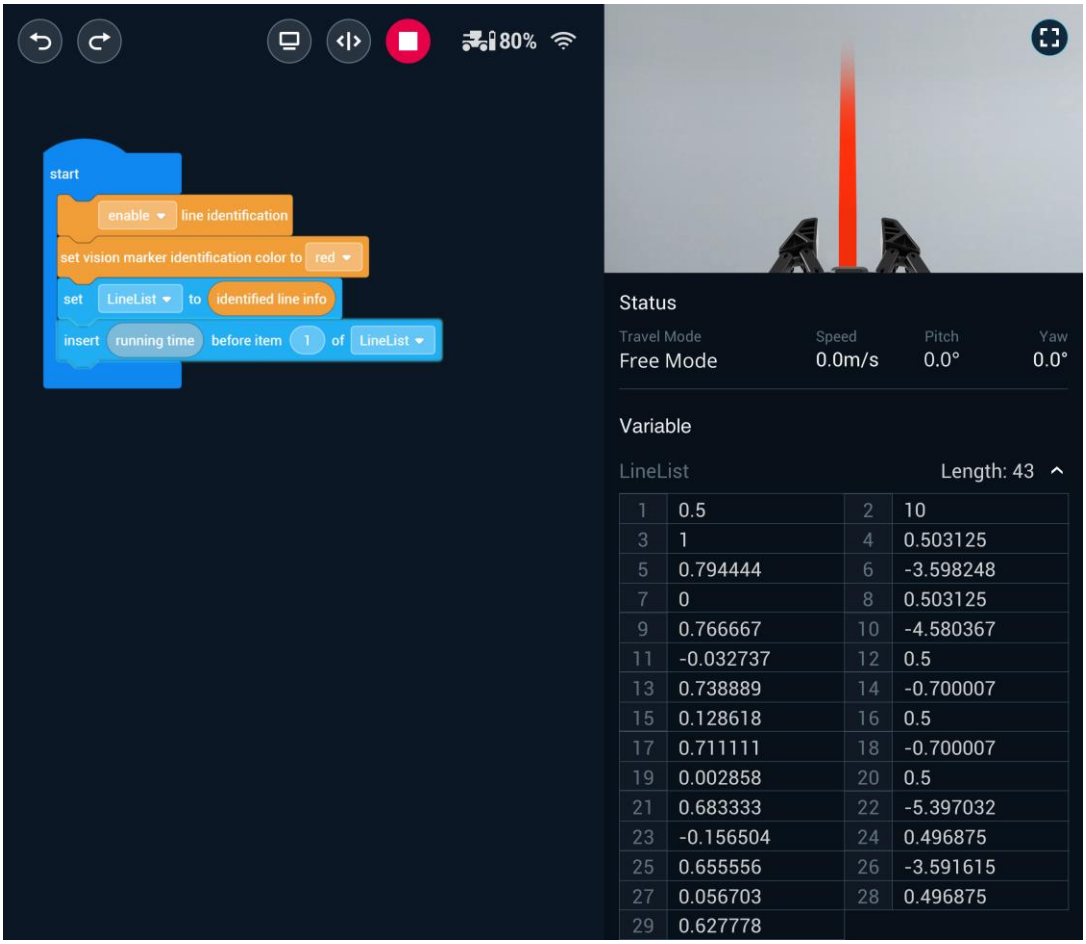
Apply the red tape and observe the FPV window. You can see that the “program runtime” is inserted as the first item of the list, the original items are shifted down the list, and the total number of items increases by 1.



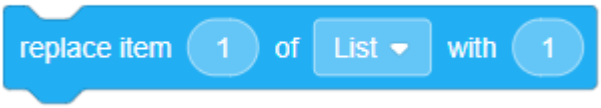
Before:



After:



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

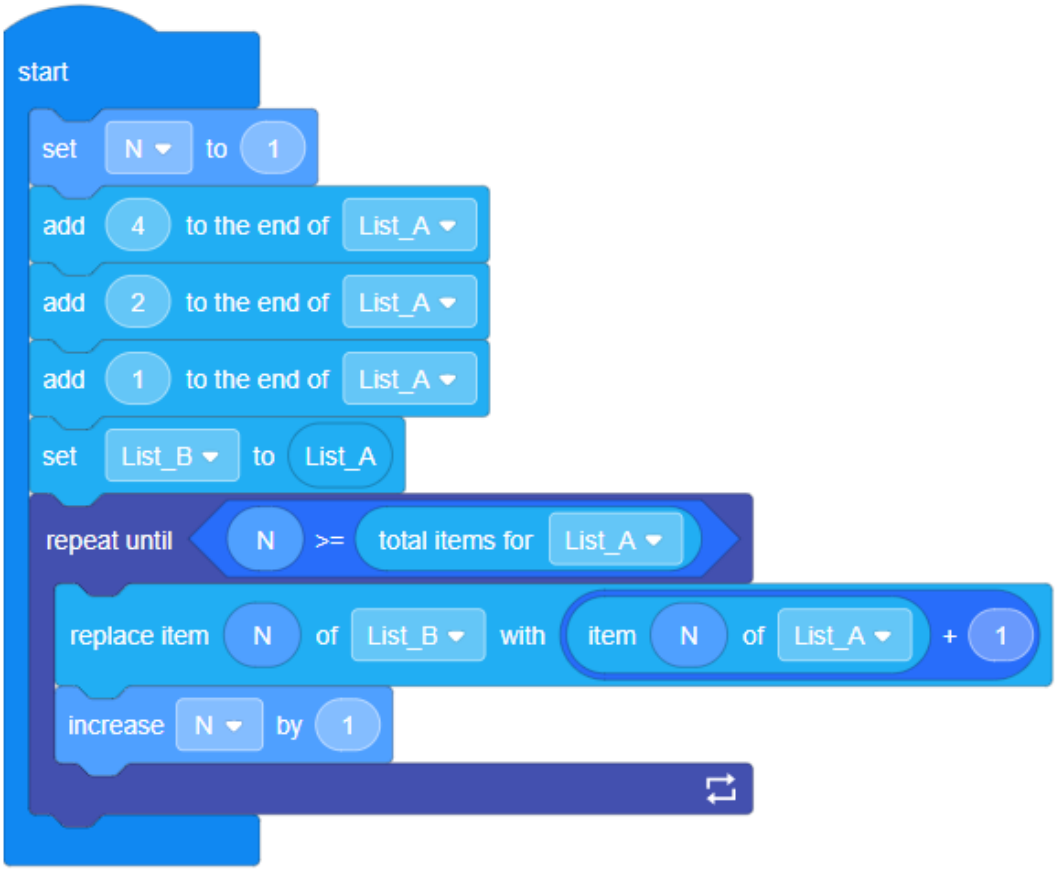


(1) Description: Replace an item on the list

(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.



Notes:

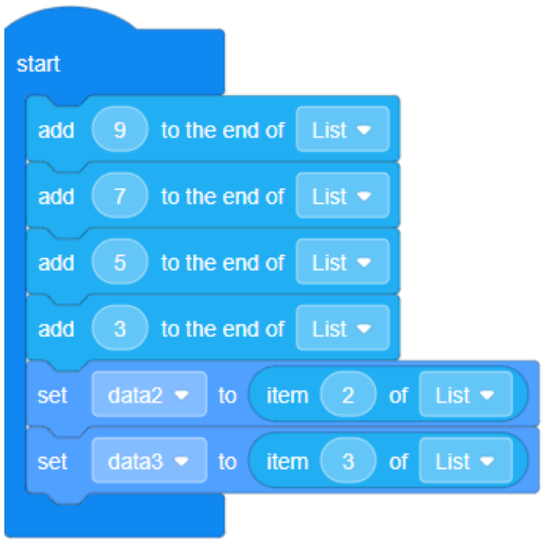
The item must contain a value before it can be replaced. For example, if List_B is empty, it is not useful by replacing one item in List_B.

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

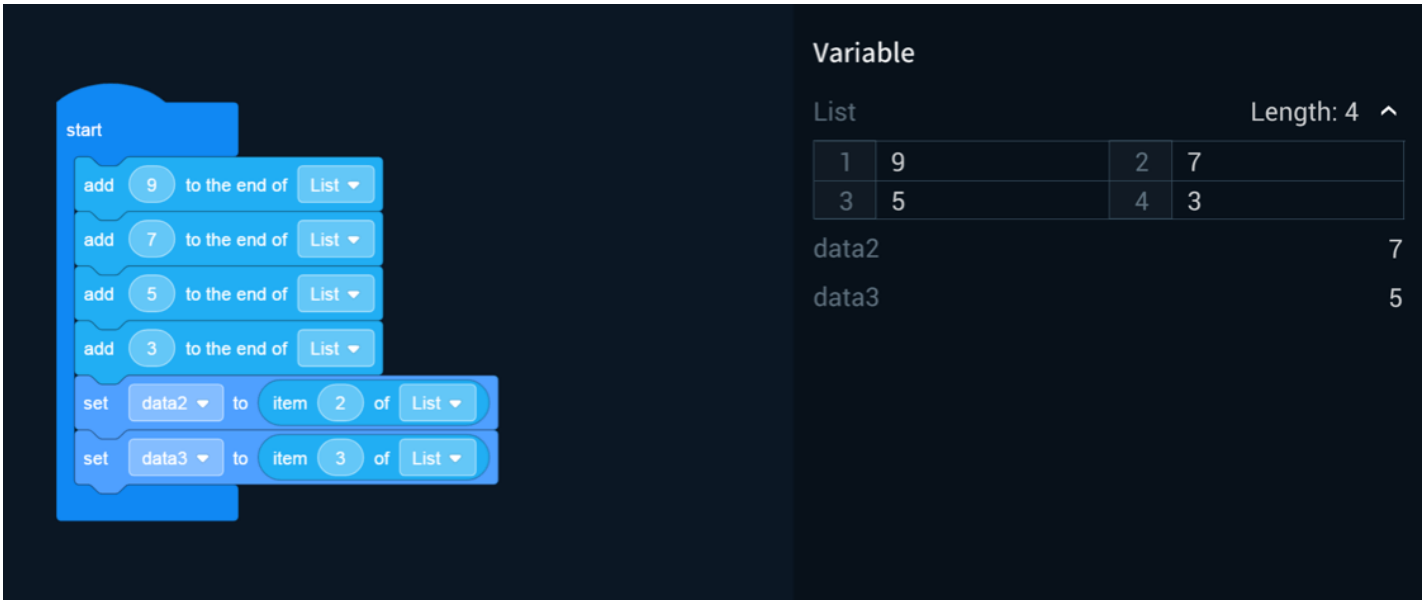
(1) Description: Obtain a specific item on a list

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

(3) Example: Specify an item on a list



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



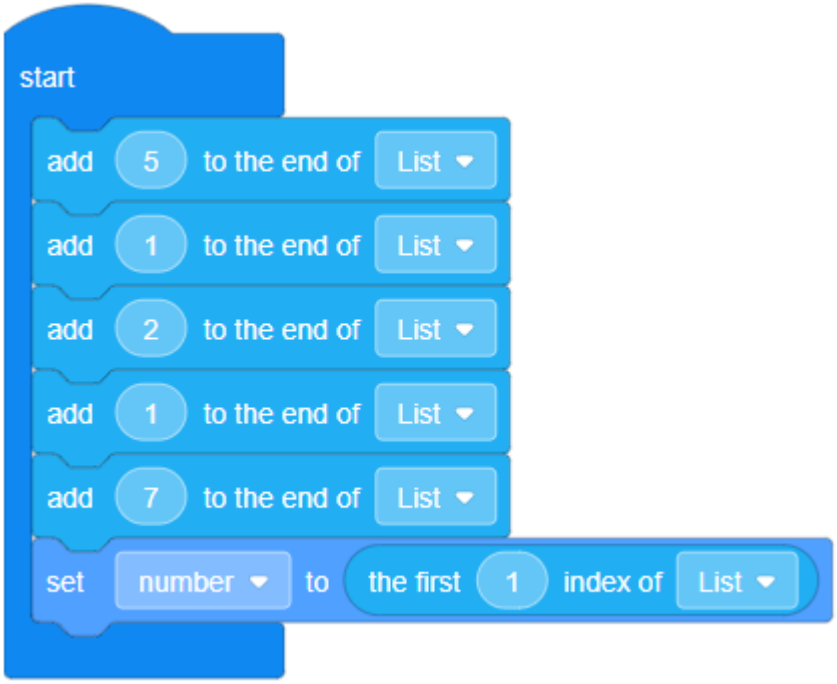
14. The first (1) index of (#List#)

（1）Description: Obtain the location of the data’s first occurrence in the current item (the number of the item)

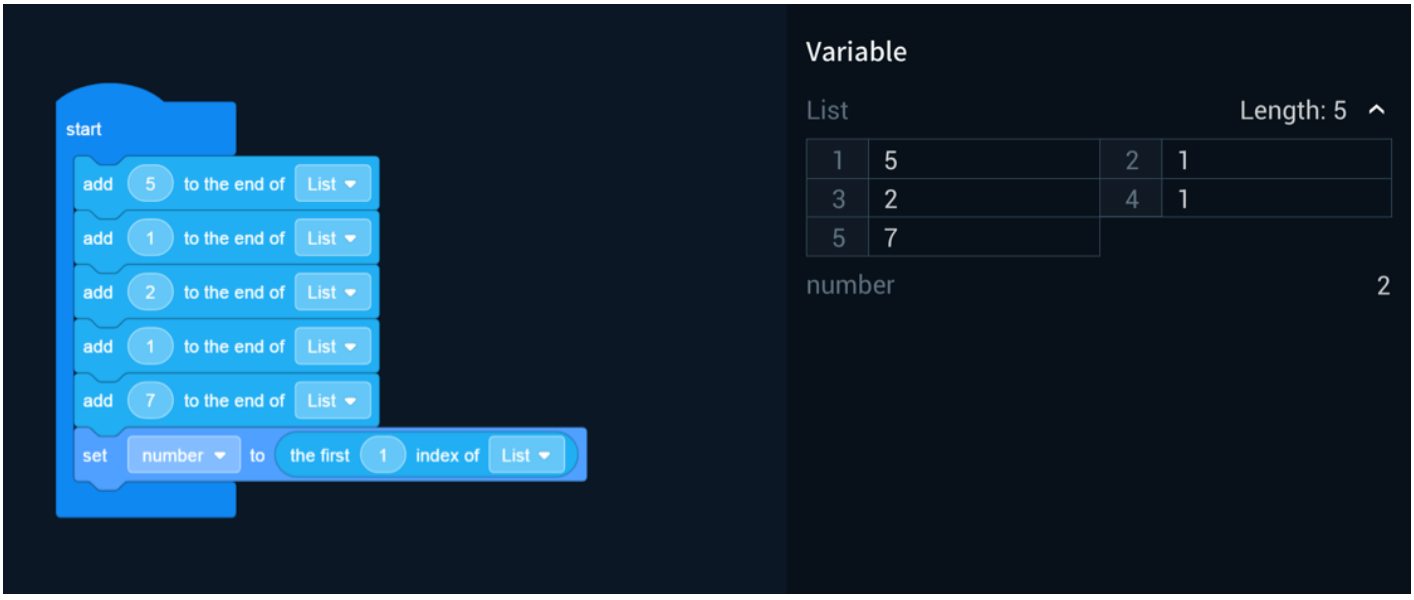
（2）Type: Information block (variable-type)

（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:

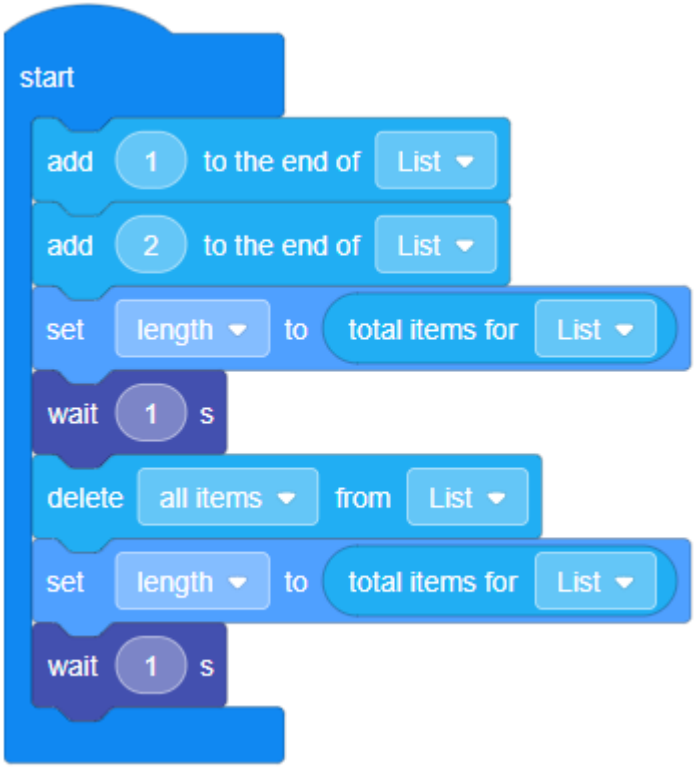


15. Total items for (#List#)

(1) Description: Obtain the total number of items for a list

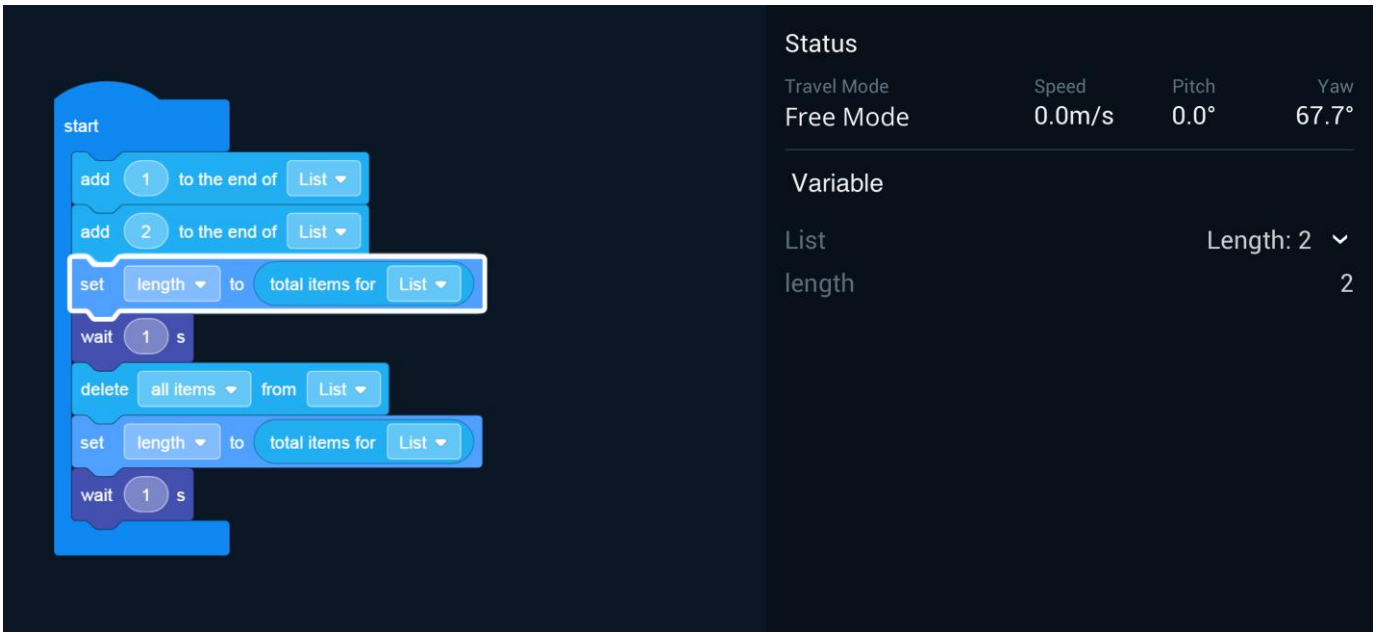
(2) Type: Information block (variable-type data)

(3) Example: Calculate the number of items

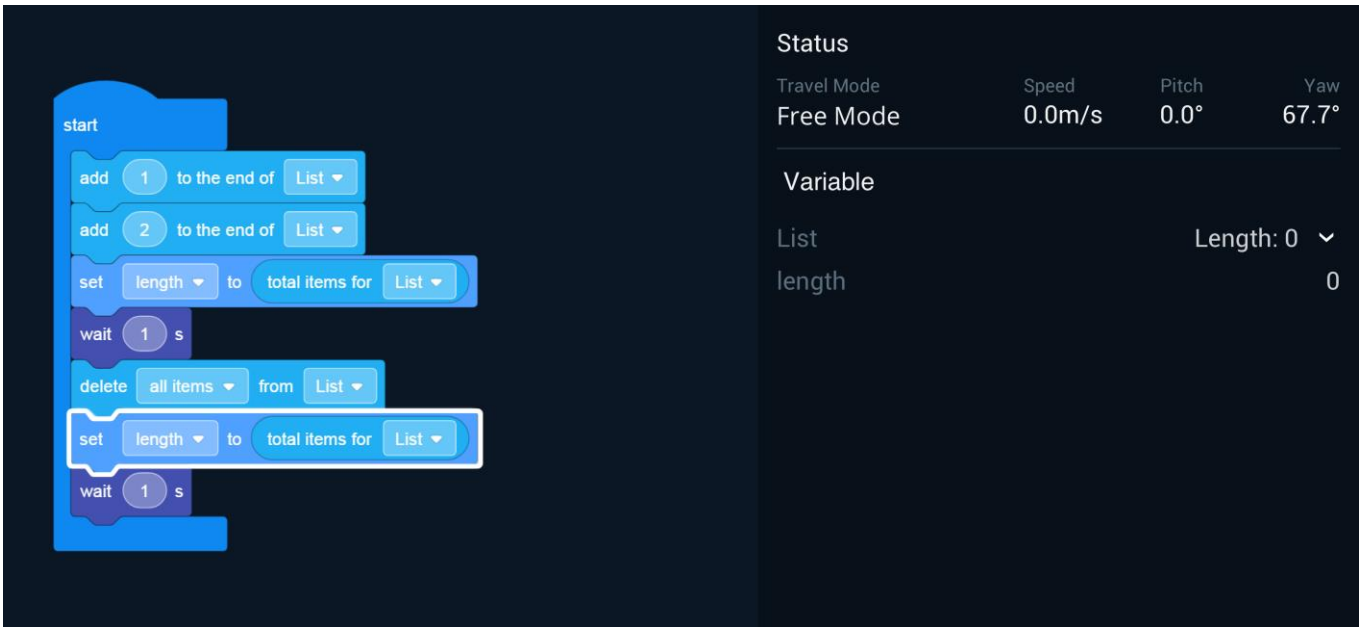


You can check value changes using the FPV window. The length value starts at 2 and becomes 0 after deleting all items.

Before:



After:



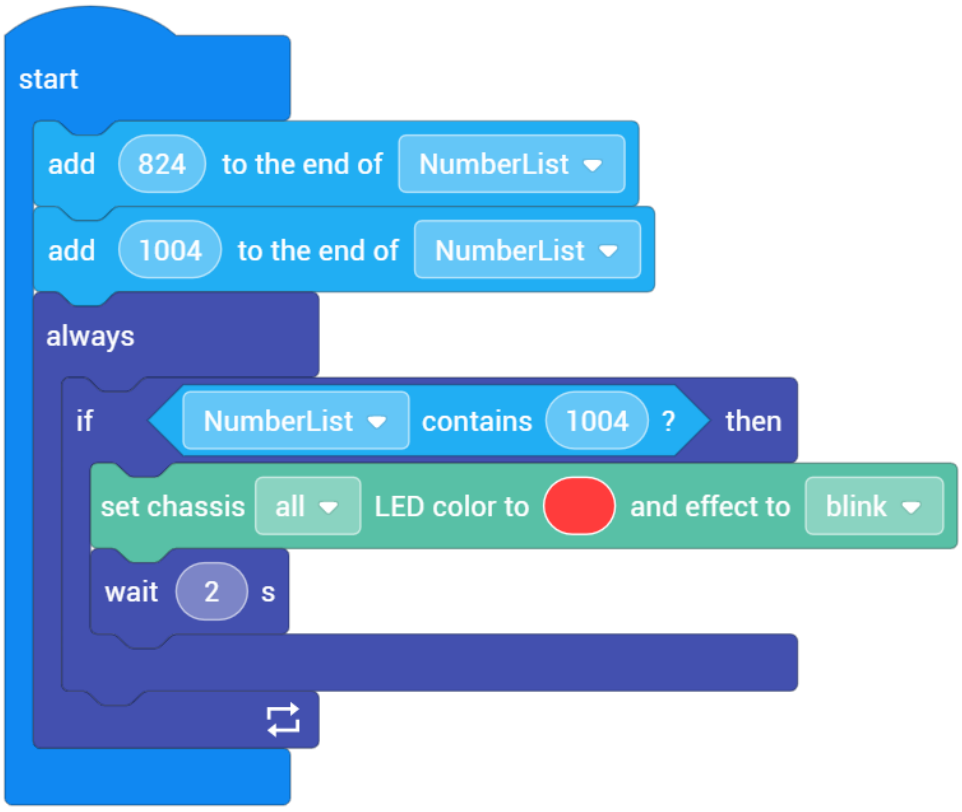
16. (#List#) contains (1)?

(1) Description: Return “True” when a list contains an input value; otherwise, “False” is returned

(2) Return value: Boolean

(3) Example: List data judgement

If a certain value is included in the list, all LEDs on the chassis blink red to indicate.



Note:

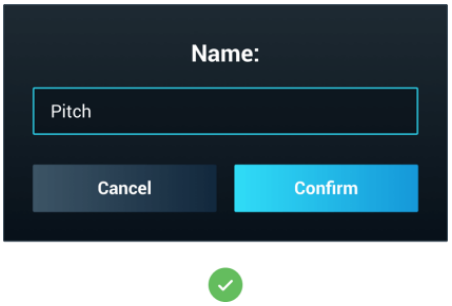
The judgment item must match exactly so the condition will be judged to be true.

17. Create PID controller

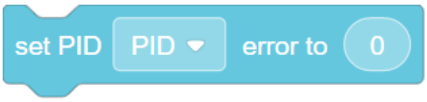
(1) Description: Create and names a PID controller

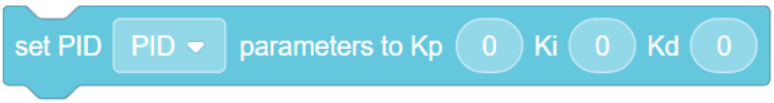
(2) Type: Settings block

(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:

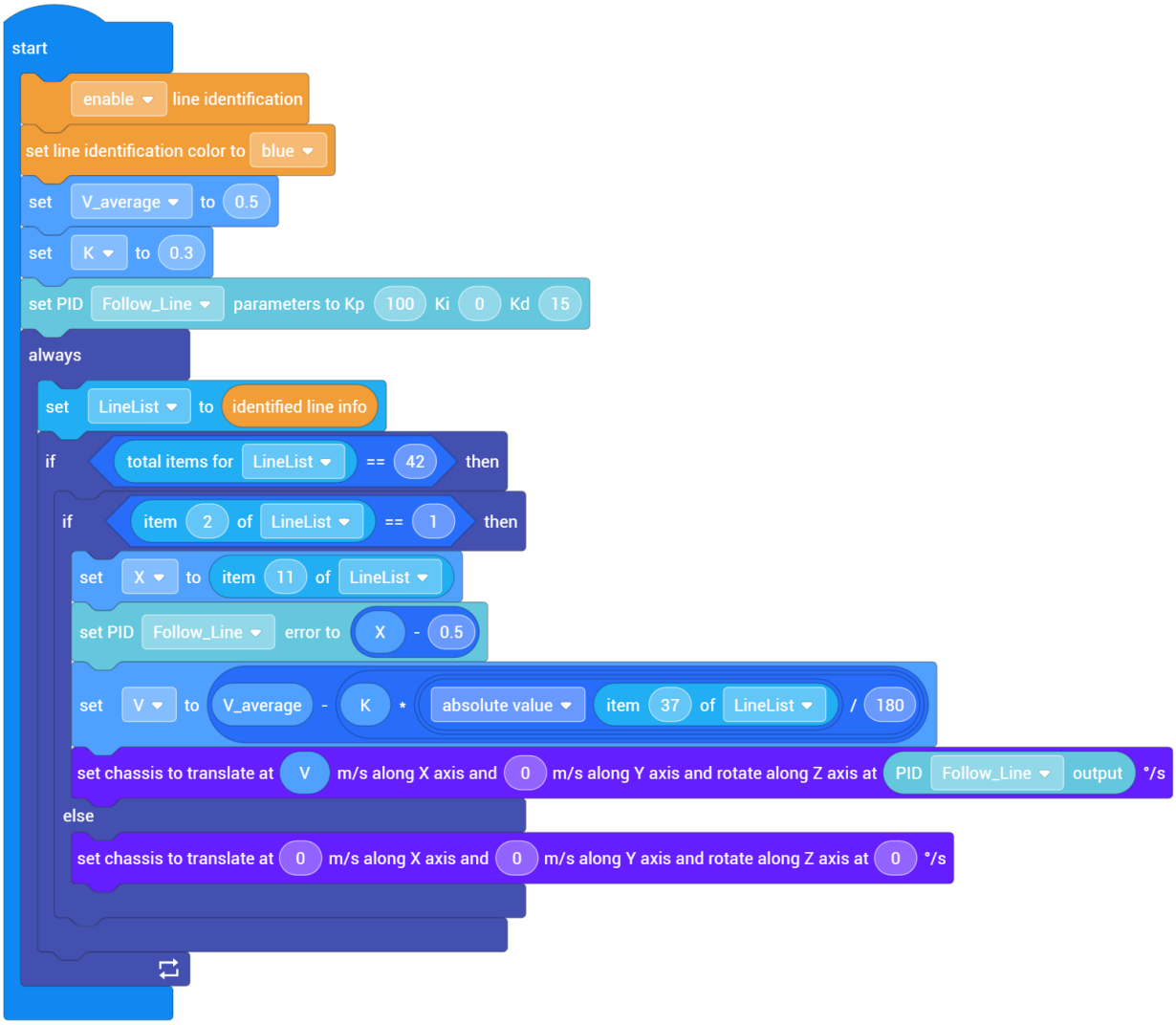
Class: rm_ctrl.PIDCtrl( )

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

(1) Description: Set the PID controller error, which is the difference between the target and returned values

(2) Type: Settings block

(3) Example: EP CORE’s line follow



Note:

Before running the program, make sure that the vision marker appears properly in the robot's FOV.

	Rotation	Upside Down	Perspective	Moving Too Fast	Insufficient sunlight	Obstructed
Case						
Impact	No Impact	No Impact	Greatly affected	Greatly affected	Greatly affected	Fail to identify

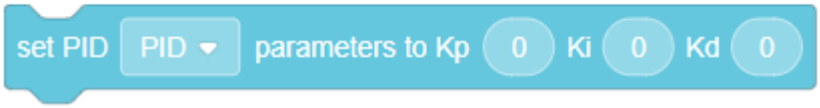
Python API:

Class: rm_ctrl.PIDCtrl( )

Function:

- set_error(error)

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

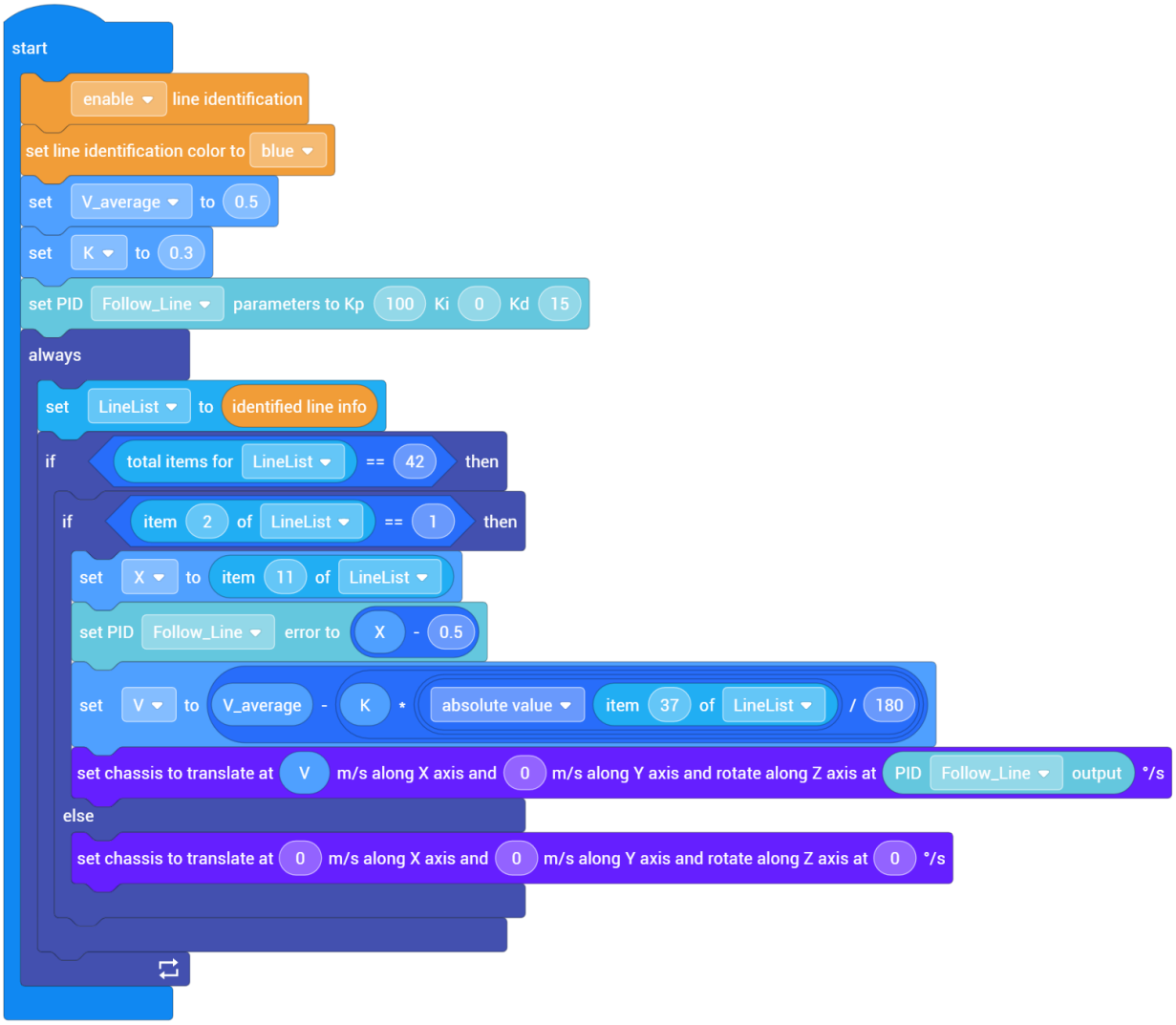


(1) Description: Adjust the PID parameters; Kp is the proportional coefficient, Ki is the integral coefficient, and Kd is the differential coefficient.

(2) Type: Settings block

(3) Example: EP CORE's line follow

Modify parameters of Kp, Ki, and Kd to optimize the closed-loop control system.



Note:

	Function	Features	Weakness
Proportional Control (P Control)	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 Control)	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 Control)	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 “Follow Vision Marker” project in the Road to Mastery section.

Python API:

Class: rm_ctrl.PIDCtrl( )

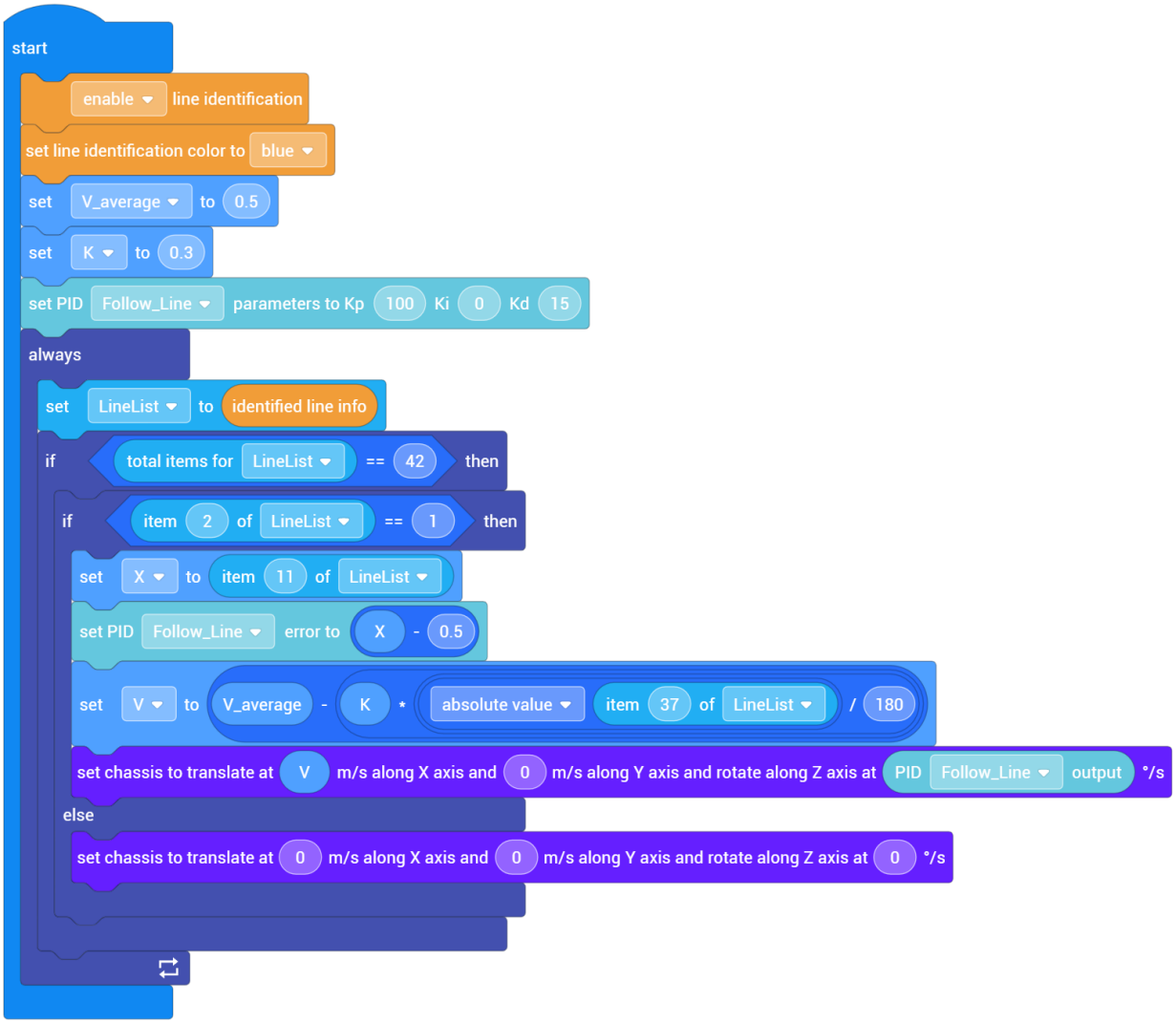
- Function:
  - set_ctrl_params(kp, ki, kd)
- Parameters:
  - kp(float)
  - ki(float)
  - kd(float)

20. PID (#PID#) output

(1) Description: Obtain the output value for a PID

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

(3) Example: EP CORE’s line follow



Python API:

Class: rm_ctrl.PIDCtrl( )

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

Functions

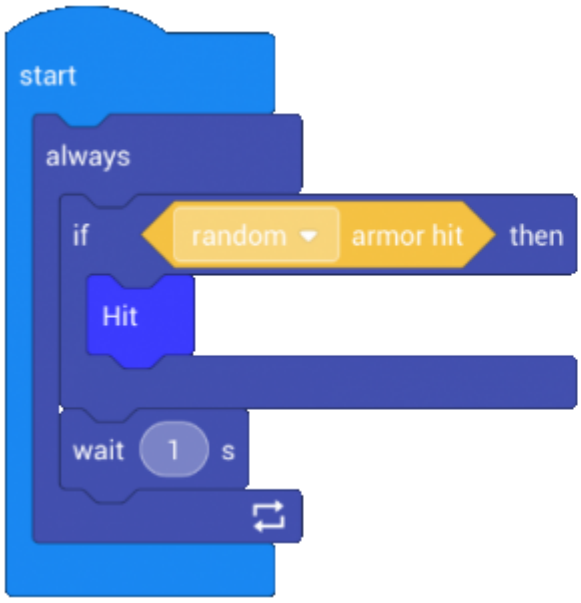
1. #Function#

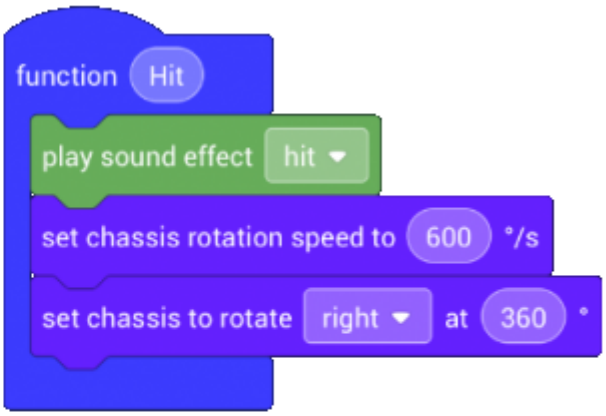
(1) Description: Package a program that needs to appear multiple times into a function, making it convenient to use

(2) Type: Function block

(3) Example: Hit rotation

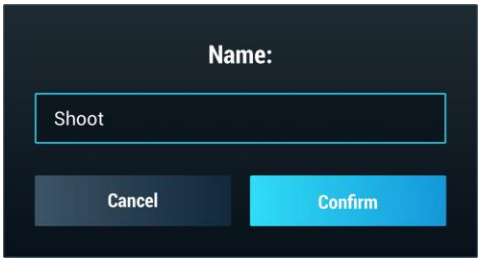
If the armor pieces on either side of the chassis are attacked, it will play the hit sound effect and the chassis will rotate right at 360°.

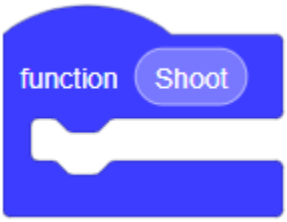




Notes:

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.