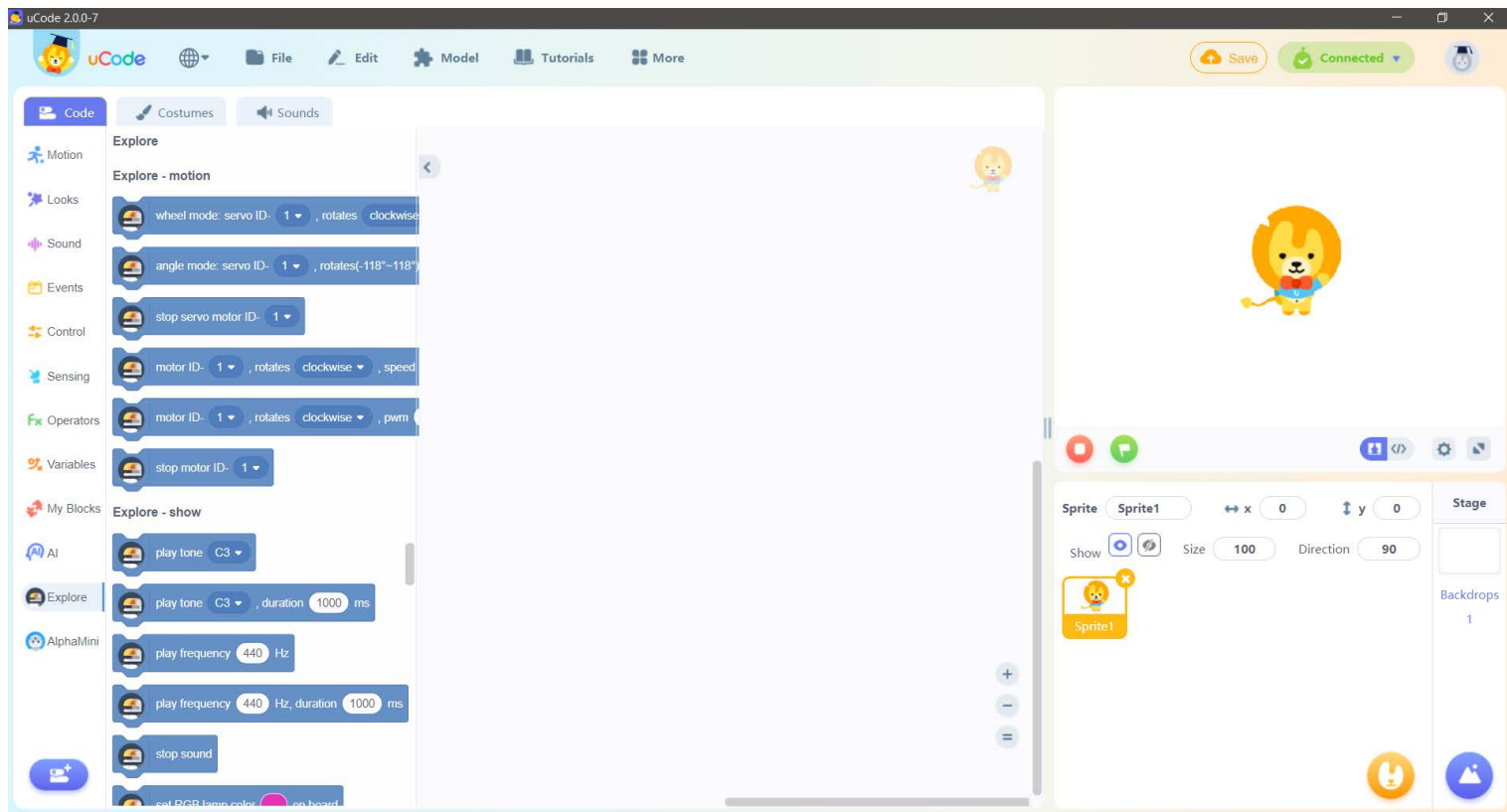




uCode

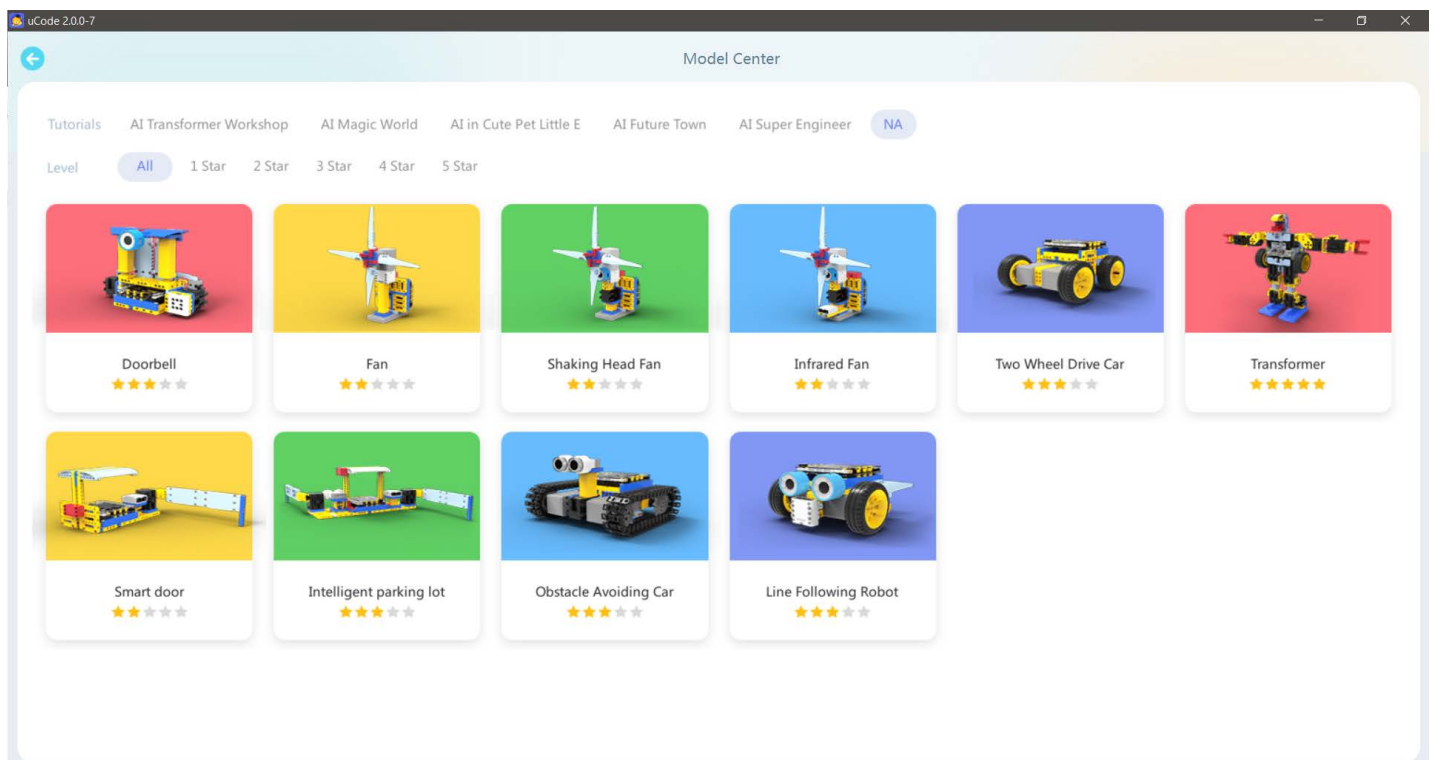
Block Guide

# What is uCode?



uCode is a software programming platform developed by UBTECH for students. Users can program robots, build games, program circuits and create animations by dragging and organizing blocks. The programming environment based on Scratch 3.0 makes it easy to jump in and start creating. Currently uCode is able to connect to a variety of hardware including UKIT Advanced microcontroller (Explore), Micro: Bit and Arduino microcontrollers.

This guide will introduce the blocks used in uCode and get to set up to start inventing!



# Sports Building Blocks

## 1. Move (10) steps



- Character moves to the right by the specified number of steps

## 2. Turn right (15) degrees



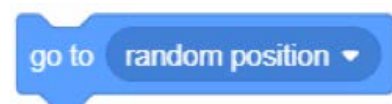
- Character turns right by specified angle

## 3. Turn left (15) degrees



- Character turns left by specified angle

## 4. Move to [random position]



- The character moves randomly or the position of the mouse pointer

## 5. Move x: (50) y: (0)



- The character moves to the specified coordinate point

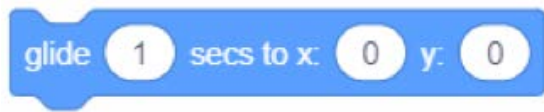
## 6. Taxi to [random position] within (1) seconds



- The character glides to a random position or the position of the mouse pointer within a specified time

#### 7. Taxi to x in (1) seconds: (50) y: (0)

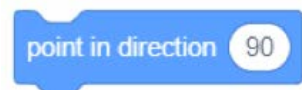
---



- The coordinate point specified by the character on the taxiway within the specified time

#### 8. Facing (90) direction

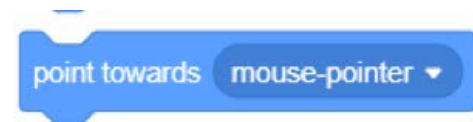
---



- The character faces the specified direction

#### 9. Facing the [mouse pointer]

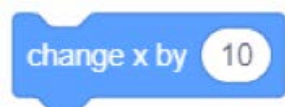
---



- The direction the character is facing the mouse pointer

#### 10. Increase the x coordinate (10)

---



- Increase the character's x coordinate by the specified value

#### 11. Set the x coordinate to (10)

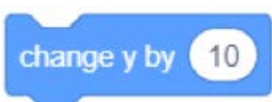
---



- Set the character's x coordinate to the specified value

#### 12. Increase the y coordinate (10)

---



- Increase the character's y coordinate by the specified value

### 13. Set the y coordinate to (10)

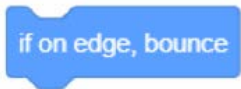
---



- Set the character's y coordinate to the specified value

### 14. Bounce at the edge

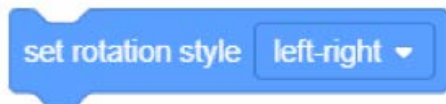
---



- When the character hits the edge of the stage, the character changes the direction of movement and moves in the opposite direction

### 15. Set the rotation method to [Flip left and right]

---



- Set the character's rotation method to flip left and right, non-rotatable or arbitrary rotation

### 16. x coordinates

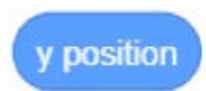
---



- Get the value of the x coordinate of the current character

### 17. y coordinates

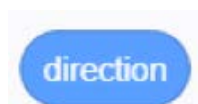
---



- Get the value of the y coordinate of the current character

### 18. Direction

---



- Get the direction of the current character

# Appearance building blocks

---

## 1. Say (), () seconds

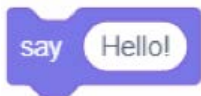
---



- A dialogue bubble pops up at the top right of the character, showing you the "entered text" and disappearing after displaying the number of seconds you set.

## 2. say ()

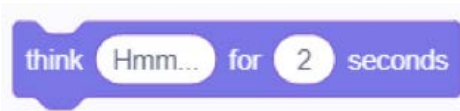
---



- A dialogue bubble pops up in the upper right of the character, showing "The text you entered".

## 3. Thinking (), () seconds

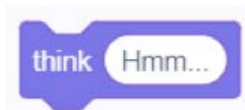
---



- A thought bubble pops up at the top right of the character, showing you "the text you entered", and disappears after continuously displaying the number of seconds you set.

## 4. Thinking ()

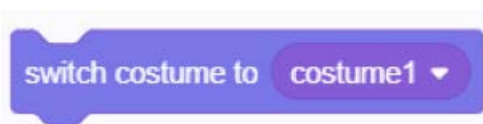
---



- A thought bubble pops up in the upper right of the character, showing "The text you entered."

## 5. Change to [Styling 1]

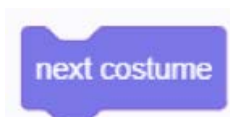
---



- The character switches to the specified look.

## 6. The next look

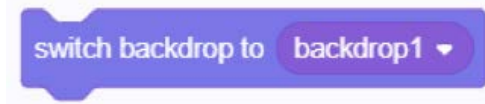
---



- Switch the character's look to the next look. If the current character is the last one in the list, it will loop to the first one.

## 7. Change to [Background 1] Background

---

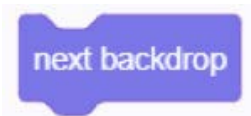


- The background switches to the specified background.

Press r to switch the background to Basketball 1

## 8. Next background

---



- Switch the background to the next background. If the current background is the last in the list, loop to the first.

## 9. Increase the size ()

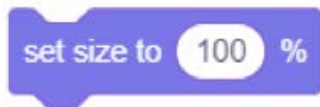
---



- Increase the character size by a percentage of "You enter a value", and negative numbers decrease.

## 10. Set the size to ()

---



- Set the character size as a percentage of the value you enter.

## 11. Increase the [Color] effect ()

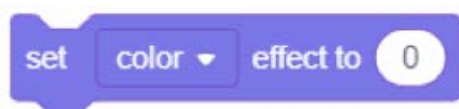
---



- Increase the character's [Color / Fisheye / Swirl / Pixelation / Mosaic / Brightness / Virtual Image] effects by a percentage of "the number you entered", and decrease the negative number.

## 12. Set the [Color] effect to ()

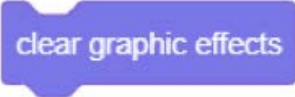
---



- Set the [Color / Fisheye / Swirl / Pixelation / Mosaic / Brightness / Dotted Line] effect to a percentage of "the value you entered".

### 13. Clear graphics effects

---

A blue block with a notch on the top-left and a bump on the bottom-right. It contains the text "clear graphic effects" in white.

- Clear all effects from the character.
- When the character is clicked, clear all the effects of the character

### 14. Display

---

A blue block with a notch on the top-left and a bump on the bottom-right. It contains the text "show" in white.

- Display the character in the running program so that the player can see the character.

### 15.Hide

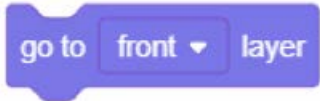
---

A blue block with a notch on the top-left and a bump on the bottom-right. It contains the text "hide" in white.

- Hide the character in the running program so that the player cannot see the character.

### 16. Move to the front [front]

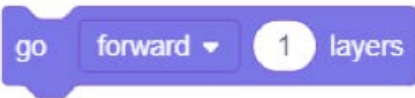
---

A blue block with a notch on the top-left and a bump on the bottom-right. It contains the text "go to" in white, followed by a dropdown menu showing "front" with a downward arrow, and the text "layer" in white.

- Move the character to the front [front / back] of the layer.

### 17. [forward] () layer

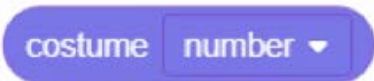
---

A blue block with a notch on the top-left and a bump on the bottom-right. It contains the text "go" in white, followed by a dropdown menu showing "forward" with a downward arrow, a white circle containing the number "1", and the text "layers" in white.

- [Front / Back] The number of layers where you enter the value.

### 18. Modeling [number]

---

A blue block with a notch on the top-left and a bump on the bottom-right. It contains the text "costume" in white, followed by a dropdown menu showing "number" with a downward arrow.

- [Number / Name] of the character model
- Example: When the green banner is clicked, the character will display the shape name through a dialog box.



## 19. Background [number]

---



- [Number / Name] of the stage background.

## 20. Size

---

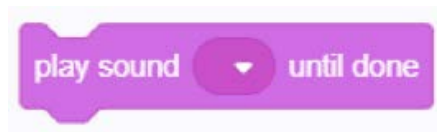


- Role size.

# Sound building blocks

1. Play the sound [recording] and wait for it to finish

---



- Play the sound of your choice and wait for it to finish executing the script below.

2. Play the sound [recording]

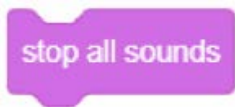
---



- Play the sound of your choice and execute the following script at the same time.

3. Stop all sounds

---



- Stop all sounds currently running

4. Increase the [tone] sound effect ()

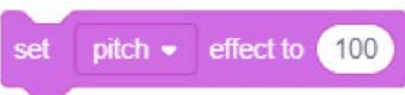
---



- Increase the [Tone / Left / Right Balance] sound effect by a percentage of "You enter the value", and negative numbers decrease.

5. Set the [Tone] sound effect to ()

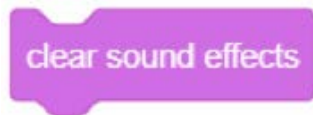
---



- Set the [Sound Effect / Left and Right Balance] sound effect as a percentage of "You enter the value".

## 6. Clear sound effects

---



- Clear all sound effects from the character.

## 7. Increase the volume ()

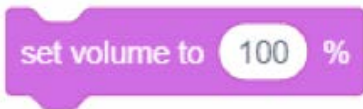
---



- Increase the volume by "you enter a value" by a percentage, and negative numbers decrease.

## 8. Set the volume (%)

---



- Set the volume as a percentage of "You enter a value."

## 9. Volume

---



- Monitor the volume of characters and the stage

# Event building blocks

---

## 1. When the green banner is clicked

---



- When the green banner is clicked, immediately execute the script example under this block: when the green

## 2. When the [Space] key is pressed

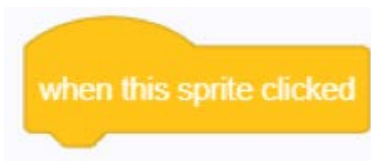
---



- When the keyboard presses the [Space /] key, the script under this block is immediately executed

## 3. When the character is clicked

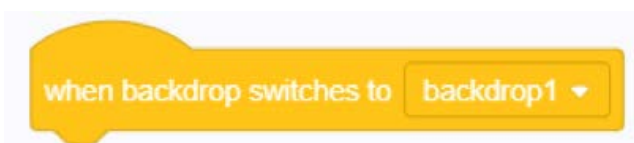
---



- When the character is clicked, immediately execute the script under this block

## 4. When the role is changed to [Background 1]

---



- When the character background is changed to "the background you specified", immediately execute the script under this block

## 5. When [loudness] is greater than ()

---



- When [loudness / timer] is greater than "you enter the value", the script under this block is executed immediately

## 6. When [message 1] is received

---



- When the specified message is received, the script under this block is immediately executed

## 7. Broadcasting [message]

---



- Send a broadcast to all characters (background) to inform the character (background) that received the broadcast content to perform some action

## 8. Broadcast [message 1] and wait

---



- Send a broadcast to all characters (background) to notify the character (background) that received the broadcast content to perform certain actions and wait for some actions initiated by the broadcast to complete.

# Controlling building blocks

## 1. Wait for () seconds



- For the separated blocks, wait for the "entered value" and execute the following script

## 2. Repeat () times



- Repeat the script blocks contained in the "Entered Value" blocks. The default value is 10 times.

## 3. Repeat



- Repeat an infinite number of times to execute the script blocks contained in the blocks

## 4. Use [My Variables] to step from (i) to (j) to (n)



- Define my variable with an initial value of i, an end value of j, and a step size of n. Each time i increases by n ( $i = i + n$ ), the script blocks included are executed when i is less than j, otherwise the loop blocks are skipped.

- Example:

```
for (i = 1; i < 10, i ++)
```

```
{
```

```
  Loop body statement;
```

```
}}
```

This loop will be executed 9 times (in the case of  $i = 0$  to  $i = 10$ )

Its execution process is this:

First assign an initial value of 1 to the variable  $i$ , and then judge, if the condition  $i < 10$  (yes) is satisfied, the statement of the loop body is executed, and then  $i++$  is completed, that is,  $i = i + 1$ ;  $i$  becomes 2 and then judges If the condition  $i < 10$  is satisfied (still), the statement of the loop body is executed again, and  $i++$  is continued after completion. When  $i$  becomes 10, it is judged whether the condition  $i < 10$  is satisfied, and it is found that it is no longer satisfied, and then exits the loop.

## 5. If <condition> then ()

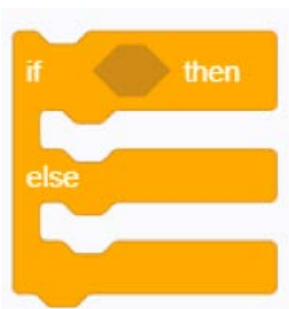
---



- If the embedded <condition> is true, the script block contained in the "if" block is executed, otherwise the block is skipped.

## 6. If <condition> then () otherwise ()

---



- If the embedded <condition> is true, execute the script blocks contained in the "if" block, otherwise execute the script blocks contained in the "other" block.

## 7. wait ()

---



- Use to separate building blocks, wait for embedding <condition> to run, run the following building block script, otherwise wait

## 8. Repeat until ()

---



- Repeat the included script blocks until the embedded <condition> is satisfied, then run the script blocks under this block

## 9. Stop [All Scripts]

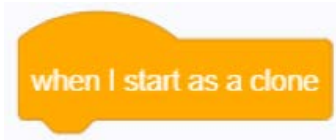
---



- Stop execution of [All scripts / This script / Other scripts of this role]

## 10. When started as a clone

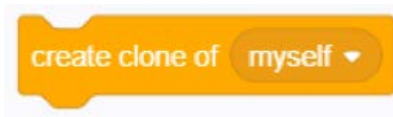
---



- When the clone is generated, execute the script block under this block.

## 11. Clone [self]

---



- Clone yourself or another character's clone

## 12. Delete this clone

---



- Delete the clone



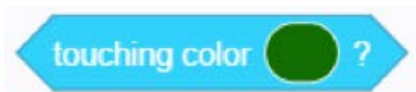
# Detect building blocks

## 1. Encountered the [mouse pointer]?



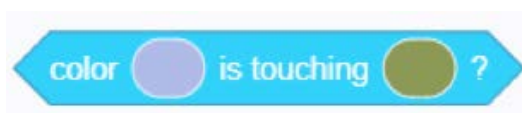
- When the character touches the [mouse pointer / edge of the stage], this building block condition is satisfied.

## 2. Encounter the color [color block]?



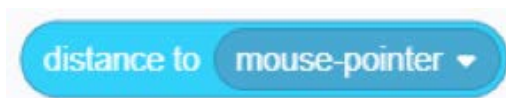
- When the color specified by the user is met, the conditions of this building block are satisfied

## 3. Color [color block] meets [color block]?



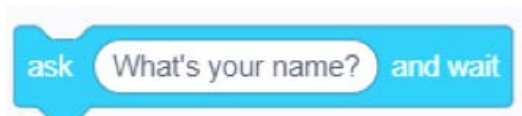
- If the specified color meets another specified color, the condition of this building block is true

## 4. Distance to [mouse pointer]



- Distance from current role to "choice role"

## 5. Ask () and wait



- The role asks the specified question and waits for the user to enter an answer.

## 6. Answer



- Save the answer to the query block, save only the most recent input value. If there is no input, a null value is saved

## 7. Press the [Space] key?



- If the specified button is pressed, the condition of this building block is satisfied

## 8. Press the mouse?

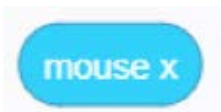
---



- If you press the mouse, the conditions for this building block are true

## 9. The x coordinate of the mouse

---



- Get the value of the mouse x coordinate

## 10. The y coordinate of the mouse

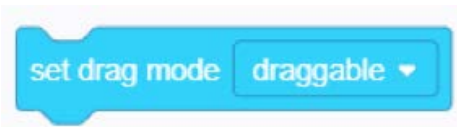
---



- Get the value of the mouse y coordinate

## 11. Set the drag mode to [Dragable]

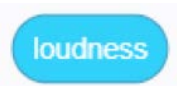
---



- Set the character's drag mode to draggable or non-draggable

## 12. Loudness

---



- Get the current loudness

## 13. Timer

---



- Get the value of the timer

## 14. Reset the timer

---



- Reset timer

## 15. [Background Number] of [Stage]

---



- Get the background number of the stage [background name / volume]

## 16. [year] at current time

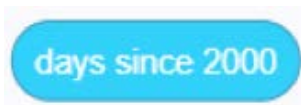
---



- Get the current year [month / date / week / hour / minute / second]

## 17. Days Since 2000

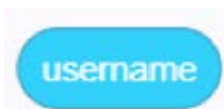
---



- Get the number of days since 2000

## 18. Username

---



- Get username

# Operational building blocks

1. ( ) + ( )



- Get parameter A and parameter B and perform addition to get the value

2. ( ) - ( )



- Get the value obtained by performing the subtraction of parameter A and parameter B

3. ( ) \* ( )



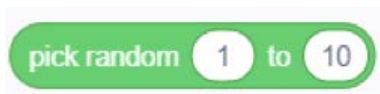
- Get the value obtained by performing the multiplication of parameter A and parameter B

4. ( ) / ( )



- Get the value obtained by performing the division of parameter A and parameter B

5. Take a random number between (1) and (10)



- Randomly draw numbers in a specified range

6. ( ) > ( 50 )



- When the parameter A is greater than the parameter B, the building block condition is satisfied

7. ( ) < ( 50 )



- When the parameter A is smaller than the parameter B, the building block condition is satisfied

8. () = (50)

---



- When the parameter A is equal to the parameter B, the building block condition is satisfied

9. () and ()

---



- When condition A and condition B are satisfied at the same time, the condition of this building block is satisfied

10. () or ()

---



- When one of the conditions A and B is true, the condition of this building block is true

11. () does not hold

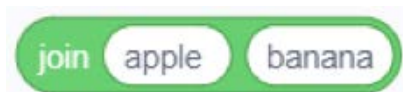
---



- When the specified conditions are not met, the building block conditions are met.

12. Connect (Apple) and (Banana)

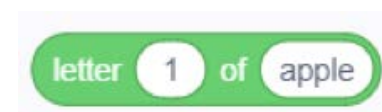
---



- Get the result of combining specified text or numbers

13. (Apple) Character (1)

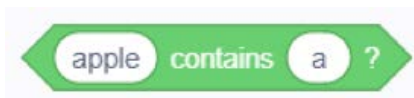
---



- Get the character at the specified position of the specified text or number

#### 14. (Apple) contains (fruit)?

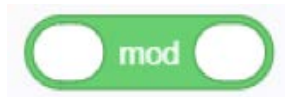
---



- When the specified text or number A contains the content of the specified text or number B, the condition of this building block is satisfied

#### 15. () divided by () remainder

---



- Get the value of the remainder of parameter A divided by parameter B

#### 16. Rounded ()

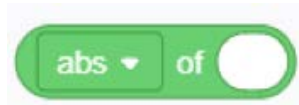
---



- Get the rounded value of the specified value

#### 17. [Absolute value] ()

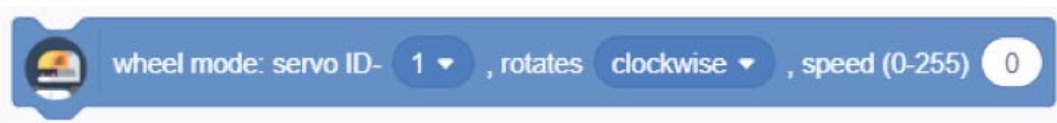
---



- Get the value of the result of the specified operation
- Algorithm: absolute value, round down, round up, square root, sin, cos, tan, asin, acos, atan, ln, log, e ^, 10

# Sports building blocks

## 1. Servo wheel mode ID- [01], [clockwise], speed (0-255) (0)



- In the steering wheel mode, the ID of the designated servo rotates clockwise [counterclockwise] at the speed of "you enter the value"

## 2. Servo angle mode ID- [01], angle (-118 ° ~ 118 ° ) (0), running time (1000) milliseconds



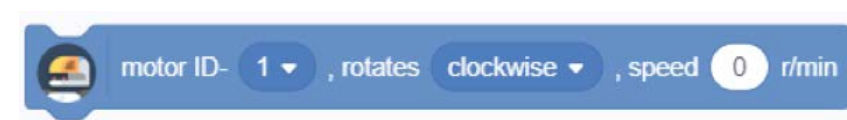
- In the servo angle mode, the ID of the designated servo goes to the next state (the angle of "you enter the value"), and the duration of "you enter the value"

## 3. Stop the servo ID- [01]



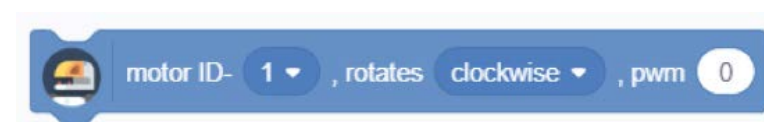
- Stop specified servo operation

## 4. Motor ID- [01], [clockwise] rotation, constant speed (0 ~ 140) (0) revolutions / minute



- The ID of the specified motor rotates clockwise [counterclockwise] at a constant speed of "You enter the value"

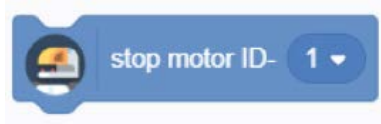
## 5. Motor ID- [01], [clockwise] rotation, PWM (100)



- The ID of the specified motor rotates clockwise [counterclockwise] according to the pulse adjustment (PWM) of "You enter the value"

## 6. Stop motor ID- [01]

---



- Stop specified motor operation



# Acousto-optic building blocks

---

## 1. Play the tone [C3]

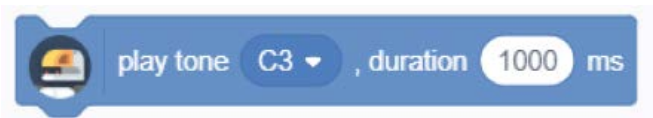
---



- Explore's buzzer plays the specified tone while executing the script below

## 2. Play tone C3, duration (1000) milliseconds

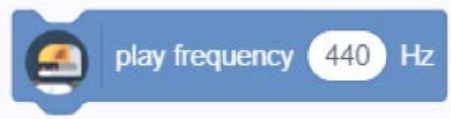
---



- Explore's buzzer plays the specified tone and runs the specified time, while executing the script below

## 3. Play frequency (400) Hz

---



- Explore the buzzer to play the sound of the specified frequency while executing the following script

## 4. Play frequency (400) Hz, duration (1000) milliseconds

---



- Explore's buzzer plays the sound of the specified frequency and runs the specified time, while executing the following script

## 5. End sound

---



- End all sounds of the current script

## 6. Onboard RGB light color is ()



- Explore's onboard RGB lights light up in specified colors

## 7. Onboard RGB light color is R (255) G (0) B (0)



- The onboard light of Explore lights up the specified
- color parameter range:

R: 0 ~ 255

G: 0-255

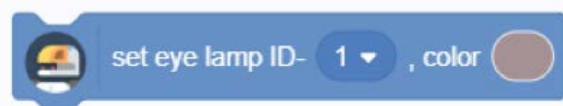
B: 0-255

## 8. Light up the eye lamp ID- [01], color ()



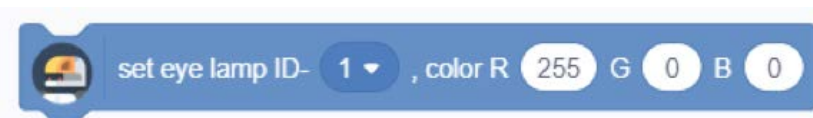
- The ID of the specified eye lamp lights up in the specified color, and execution will not stop at the same time

## 9. Light up the eye lamp ID- [01], color R (255) G (0) B (0)



- The specified eye lamp ID lights up the specified color, and execution will not stop at the same time

## 10. Eye light ID- [01], scene light [colorful marquee light], (1) times



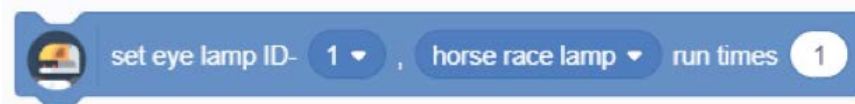
- The specified eyelight ID displays the unspecified scene light [Colorful Marquee / Disco / Three Primary Colors / Color Stacking], and runs the specified number of times while executing the following script

## 11. Eye light ID- [01], scene light [colorful marquee light], (1) times until end



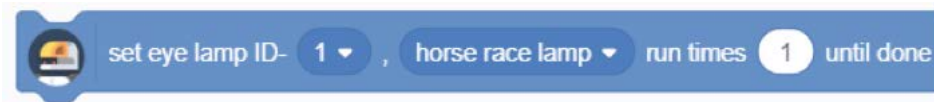
- The specified eye light ID is displayed as the specified scene light [Colorful Marquee / Disco / Three Primary Colors / Color Stacking]. Until the specified number of times are completed, the following script will be executed.

## 12. Eye lamp ID- [01], custom light flap [color] duration (100) milliseconds



- The specified eye lamp ID lights up the color of the specified light petal, and runs the specified time, while executing the following script

## 13. Eye lamp ID- [01], custom light flap [color] duration (100) milliseconds until the end



- The specified eye lamp ID lights up the color of the specified light petal. Until the specified time is completed, the following script will be executed.

## 14. Turn off the eye light ID- [01]



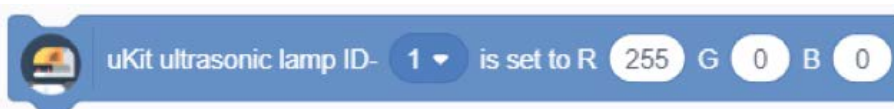
- Turn off the effect of the specified eye light

## 15. uKit time-out light ID- [1] is set to (color)



- The specified ultrasonic light ID lights up the specified color, and execution will not stop at the same time

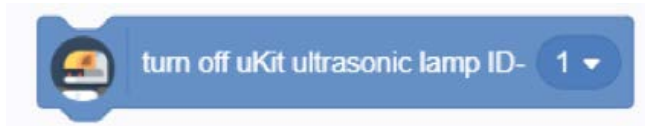
## 16. UKit ultrasonic light ID- [1] is set to R (255) G (0) B (0)



- The specified ultrasonic light ID lights up the specified color, and execution will not stop at the same time

## 17. Turn off uKit time-out light ID- [01]

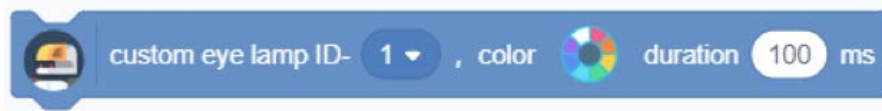
---



- Turn off the effect of the specified ultrasound

## 18. Eyclight ID- [01], custom light flap [color] duration (100) milliseconds

---



The specified eye lamp ID lights up the color of the specified light petal, and runs the specified time, while executing the following script

## 19. Eye lamp ID- [01], custom light flap [color] duration (100) milliseconds until the end

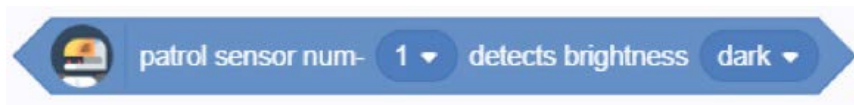
---



The specified eye lamp ID lights up the color of the specified light petal, and the execution of the following script is started until the specified time is completed

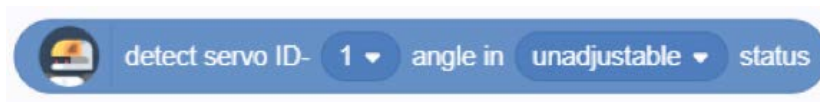
# Perception building blocks

1. Line inspection sensor num- [01] detects [deep]



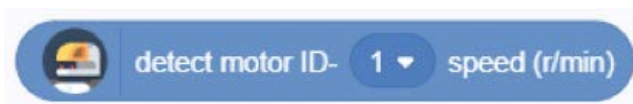
- When the color detected by the line-tracking sensor at the specified detection port is [deep or light], the condition of this building block is true

2. Read the angle of the servo ID- [01] when the status is [non-movable]



- Get the angle of the specified servo at [non-automatic / automatic]

3. Read the speed of motor ID- [01] (r / min)



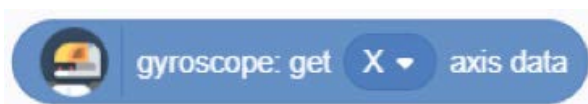
- Get the speed of the specified motor

4. Get the battery voltage

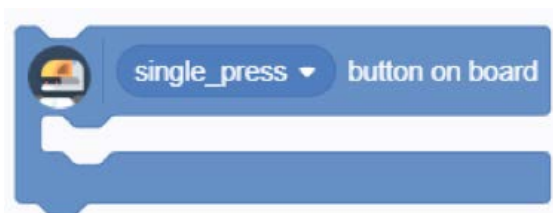


- Get the voltage value of the battery connected to the current Explore

5. Gyroscope [Acceleration (m / s<sup>2</sup>)] Select [X axis]



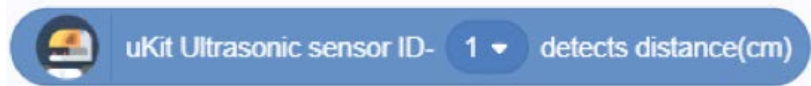
6. The on-board status of the button is [click]



- If the on-board state of the button is [click / double-click / long press], the script building block included in the building block is executed, otherwise the building block is skipped

## 7. uKit ultrasonic sensor ID- [01] Get distance (cm)

---



- Get the distance of the specified uKit ultrasonic sensor using ultrasonic testing

## 8. uKit infrared sensor ID- [01] get distance

---



- Get the distance tested by the specified uKit infrared sensor

## 9. Press sensor ID- [01] status is [click]

---



- If the status of the specified installation sensor is [click / double-click / long press], the script building block included in the building block is executed, otherwise the building block is skipped

## 10. Press the sensor ID- [01] status is [click]

---



When the status of the specified compression sensor is [click / double-click / long press], the condition of this building block is satisfied

## 11. Brightness sensor ID- [01] obtains the brightness value (lux)

---



- Get the brightness value of the specified brightness sensor

## 12. Sound sensor ID- [01] obtains the sound intensity value

---



- Get the sound intensity value of the specified sound sensor

### 13. Humidity sensor ID- [01] Get humidity%

---



- Gets the humidity percentage of the specified humidity sensor

### 14. Color sensor ID- [01] detects [R] value

---



- Get the [R / G / B] value of the specified color sensor

### 15. Color sensor ID- [01] detects [R] value

---



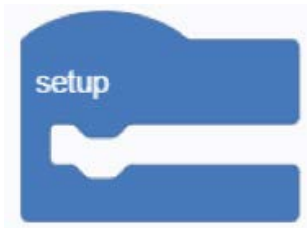
- Get the [R / G / B] value of the specified color sensor

# Start building blocks

---

## 1. Initialization

---



- The content of the script contained in the initialization block will not be repeatedly executed.
- Note: The initialization module is not allowed to be copied.

## 2. Stop the program

---



- Stop the current script



# Sports building blocks

## 1. Servo wheel mode ID- [01], [clockwise], speed (0-255) (1)

wheel mode: servoID- 01 ▾ , rotates clockwise ▾ , speed(0~255) 1

- In the steering wheel mode, the ID of the designated servo rotates clockwise [counterclockwise] at the speed of "you enter the value"

## 2. Servo angle mode ID- [01], angle ( $-118^{\circ} \sim 118^{\circ}$ ) (0), running time (1000) milliseconds

angle mode: servo ID- 01 ▾ , rotates ( $-118^{\circ} \sim 118^{\circ}$ ) 0 , duration 1000 ms

- In the servo angle mode, the ID of the designated servo goes to the next state (the angle of "you enter the value"), and the duration of "you enter the value"

## 3. Servo angle mode ID- [01], angle ( $-118^{\circ} \sim 118^{\circ}$ ) (0), running time (1000) ms until end

angle mode: servo ID- 01 ▾ , rotates ( $-118^{\circ} \sim 118^{\circ}$ ) 0 , duration 1000 ms until done

- In the servo angle mode, specify the ID of the servo to the next state (the angle of "you enter the value"), and run the time of "you enter the value" until the operation is completed

## 4. Stop the servo ID- [01]

stop servo 01 ▾

- Stop specified servo operation

## 5. Line following task speed (0-255) (100)

patrol mode: servo speed(0~255) 100

- In line patrol mode,

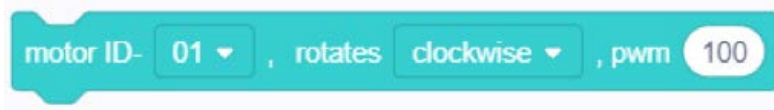
## 6. Motor ID- [01], [clockwise] rotation, constant speed (100) revolutions / minute

motor ID- 01 ▾ , rotates clockwise ▾ , speed 100 r/min

- The ID of the specified motor rotates clockwise [counterclockwise] at a constant speed of "You enter the value"

#### 7. Motor ID- [01], [clockwise] rotation, PWM (100)

---



- The ID of the specified motor rotates clockwise [counterclockwise] according to the pulse adjustment (PWM) of "You enter the value"

#### 8. Stop motor ID- [01]

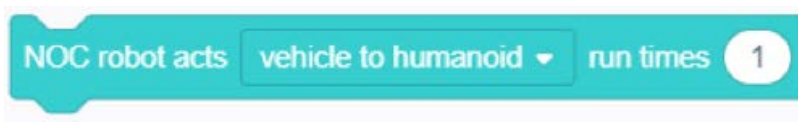
---



- Stop specified motor operation

#### 9. NOC robot action [car becomes humanoid] running times (1)

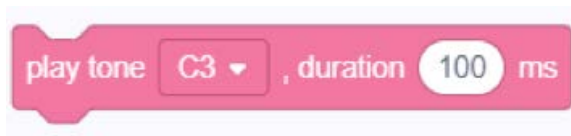
---



- NOC robot action list, only available for NOC robot models, where "vehicle stop" ignores the number of runs

# Acousto-optic building blocks

## 1. Play tone [C3], duration (100) milliseconds



- Explore's buzzer plays the specified tone and stops after running the specified time

## 2. Play frequency (400) Hz, duration (100) milliseconds



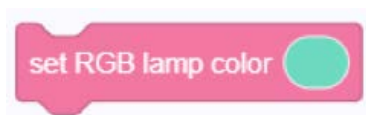
- Explore's buzzer plays a sound of a specified frequency and stops after running for a specified time

## 3. End sound



- End all sounds of the current script

## 4. Onboard RGB light color is ()



- Explore's onboard RGB lights light up in specified colors

## 5. Onboard RGB light color is R (0) G (0) B (0)



- The onboard light of Explore lights
- up the specified color parameter

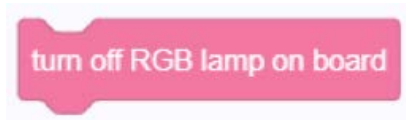
range:

R : 0~255

G : 0~255

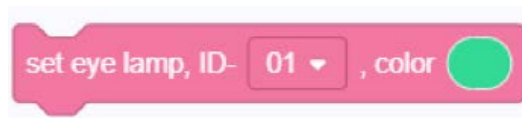
B : 0~255

## 6. Turn off the onboard RGB lights



- Turn off Explore's onboard RGB lights for the current script

## 7. Light up the eye lamp ID- [01], color ()



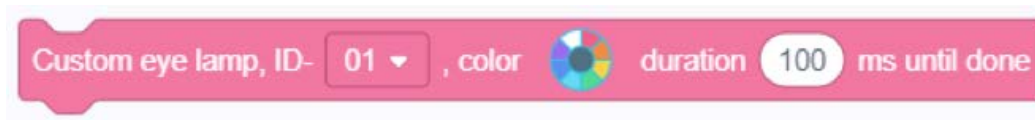
- The ID of the specified eye lamp lights up in the specified color, and the script execution will not stop

## 8. Light up the eye lamp ID- [01], color R (0) G (0) B (0)



- The specified eye lamp ID lights up in the specified color, and the script execution will not stop

## 9. Eye lamp ID- [01], custom light flap [color] duration (100) milliseconds until end



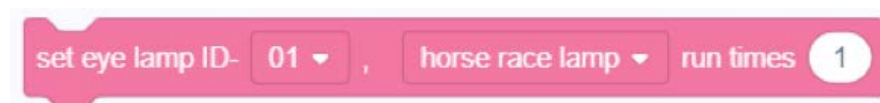
- The specified eye lamp ID lights up the color of the specified light petal and runs for the specified time

## 10. Eyclight expression ID- [01], display expression [blink], color (), (1) times



- The specified eye lamp ID displays the expression of your specified color, and runs the specified number of times. At the same time, the following script

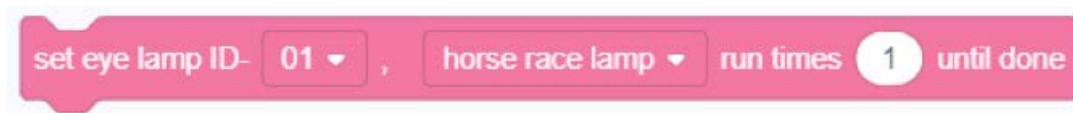
## 11. Eye light ID- [01], scene light [colorful marquee light] (1) times



- The specified eyclight ID displays the unspecified scene light [Colorful Marquee / Disco / Three Primary Colors / Color Stacking], and runs the specified number of times while executing the following script

## 12. Eyelight ID- [01], scene light [Colorful Marquee] (1) times until the end

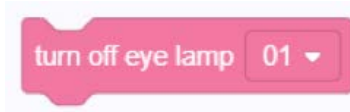
---



- The specified eye light ID is displayed as the specified scene light [Colorful Marquee / Disco / Three Primary Colors / Color Stacking]. Until the specified number of times are completed, the following script will be executed.

## 13. Turn off the eye light [01]

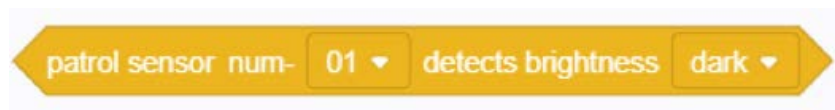
---



- Turn off the effect of the specified eye light

# Perceptual building blocks

1. Line inspection sensor num- [01] detects [deep]



2. Read the angle of the servo ID- [01] when the status is [non-movable]



- Get the angle of the specified servo at [non-automatic / automatic]

3. Read the speed of motor ID- [01] (r / min)



- Get the speed of the specified motor

4. uKit ultrasonic sensor ID- [01] ultrasonic ranging (cm)



- Get the distance of the specified uKit ultrasonic sensor using ultrasonic testing

5. uKit infrared sensor ID- [01] get distance



- Get the distance tested by the specified uKit infrared sensor

6. Get the battery voltage



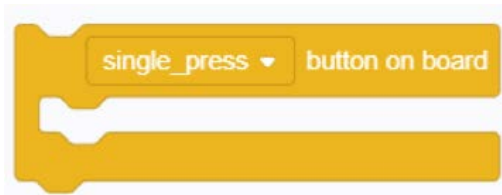
- Get the voltage value of the battery connected to the current Explore

7. Gyroscope [Acceleration (m / s<sup>2</sup>)] Select [X axis]



8. The onboard status of the button is [click]

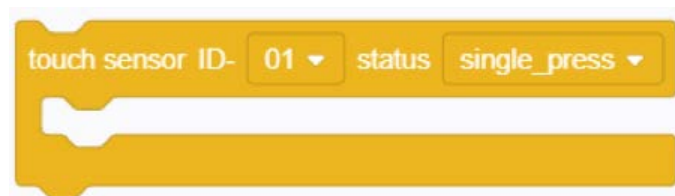
---



- If the on-board state of the button is [click / double-click / long press], the script building block included in the building block is executed, otherwise the building block is skipped

9. Press sensor ID- [01] status is [click]

---



- If the status of the specified installation sensor is [click / double-click / long press], the script building block included in the building block is executed, otherwise the building block is skipped

10. Press the sensor ID- [01] status is [click]

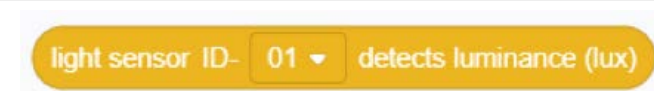
---



When the status of the specified compression sensor is [click / double-click / long press], the condition of this building block is satisfied

11. Brightness sensor ID- [01] obtains the brightness value (lux)

---



- Get the brightness value of the specified brightness sensor

12. Sound sensor ID- [01] obtains the sound intensity value

---



- Get the sound intensity value of the specified sound sensor

### 13. Humidity sensor ID- [01] Get humidity%

---



- Gets the humidity percentage of the specified humidity sensor

### 14. Color sensor ID- [01] detects [R] value

---



- Get the [R / G / B] value of the specified color sensor.

### 15. Color sensor ID- [01] detects that the color is [red]

---



- When the specified color sensor detects that the color is [Red], the condition of this building block is successful



# Logic building blocks

## 1. wait (1000) [ms]



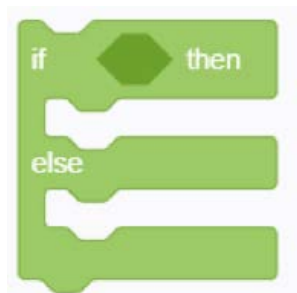
- After waiting for the specified time, execute the following procedures

## 2. If () then



- If the embedding is true, execute the script block contained in the "if" block, otherwise skip this block

## 3. If () then [] otherwise []



- If the embedded <condition> is true, execute the script blocks contained in the "if" block, otherwise execute the script blocks contained in the "other" block

## 4. Use [i] to execute from (1) to (10) in steps of (1)



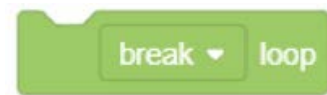
## 5. Repeat [when satisfied] () execute



- When the conditions in [Meet / Until] are satisfied, the contained script content is repeatedly executed

## 6. [Jump Out] Loop

---



- [Jump out \ Skip to next] The current script program interrupts the current loop

## 7. System running time [ms]

---



- Get the current program running time

## 8. (50) [=] (50)

---



- If the specified parameter [equal / not equal / less than / less than / less than / greater than / greater than or equal to], the condition of this building block is satisfied

## 9. () [and] ()

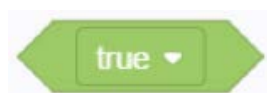
---



- If the specified conditions [and / or], this building block condition holds

## 10. [True]

---



- If the condition is [true / false / empty], the condition of this building block is satisfied

## 11. Not ()

---



- When the specified condition is not true, the condition of this building block is true

## 12. Judge () if true () if false ()

---



- The judgment condition is true, execute the second parameter, otherwise execute the third parameter

## 13. Judge () if true () if false ()

---



- When the judgment condition is false, execute the second condition, otherwise execute the third condition

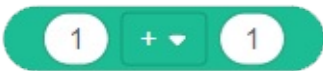
# Math building blocks

## 1.Value



- You can enter an integer or decimal to return the value

## 2. (1) [+] (1)



- Add, subtract, multiply and divide parameters A and B, take
- the remainder and get the power

Addition: () + () = A + B

Subtraction: () - () = A - B

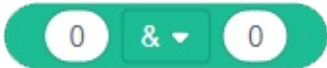
Multiplication: () x () = A x B

Division: () ÷ () = A ÷ B

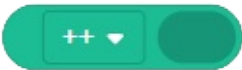
The remainder: () % () = A ÷ B The remainder is C

Power: () ^ () = A ^ B

## 3. (0) & (0)



## 4. ++

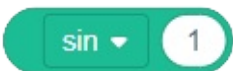


- Note: Monocular operations can only operate on variables

## 5. () [++]



## 6. [sin] (1)



- Get the value obtained by the specified algorithm
- Contains:  $\sin()$ ,  $\cos()$ ,  $\tan()$ ,  $\arcsin()$ ,  $\arccos()$ ,  $\arctan()$ ,  $\ln()$ ,  $\log_{10}()$ ,  $e^{\phantom{x}}$ ,  $10^{\phantom{x}}$

## 7. [Round (rounded)] ()

---



- Operate the value to another value type using the
- selected operation form:

Round (rounded):  $3.1415926 = 3$

Round (upper limit):  $4.24 = 5$

Round (lower limit):  $4.99 = 4$

Take the absolute value:  $-1.24 = 1.24$

Square:  $3 = 3^2$

Square root:  $9 =$



- You can enter an integer or decimal to return the value

## 8. [take the maximum value] (1,2)

---



- Get the maximum (minimum) type of
- parameter A and parameter B:

Take the maximum value:  $(10, 9) = 10$

Take the minimum value:  $(4, 5) = 4$

## 9. Initialize random numbers

---



- Random numbers are not really random, they are deterministic. Initialize the random number, you can make a different random number sequence each time the script is run

## 10. Constraint (1) is between (minimum) (1) and (maximum) (100)

---



- Gets a value that limits the number between two specified numbers

## 11. Mapping (1) from [1,100] to [1,1000]

---



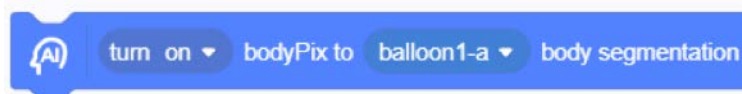
- Maps a specified value from the first interval to the second interval and returns the value after the mapping

# AI building blocks

## AI building blocks



### 1. AI-image segmentation



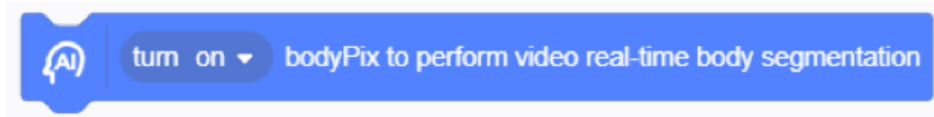
This building block can recognize the human silhouette of a character on the stage, separate the background from the portrait, and retain only the basic portrait. Note that this building block usually only works for portrait pictures with background.

Working principle: This building block uses the open source machine learning model bodyPix, which allows for image segmentation of images or videos in uCode. In real-life application scenarios, portrait segmentation technology can be used for matting and beautification, human body special effects, and post-processing of film and television.

Preview of running effect:



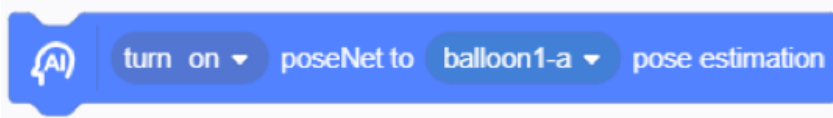
## 2. Real-time portrait segmentation of video



This building block can split the portrait of the video stream, separate the background from the portrait, and only keep the basic portrait.

Working principle: This building block uses the open source machine learning model bodyPix, which allows for image segmentation of images or videos in uCode. In real-life application scenarios, portrait segmentation technology can be used for matting and beautification, human body special effects, and post-processing of film and television.

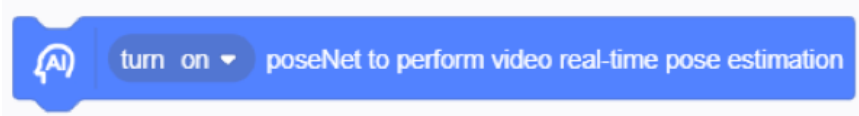
## 3. Posture detection on character modeling



This building block can perform real-time pose prediction on the character modeling on the stage.

Working principle: This building block uses the open source machine learning model PoseNet, which can predict poses of key positions of the human body in images or videos

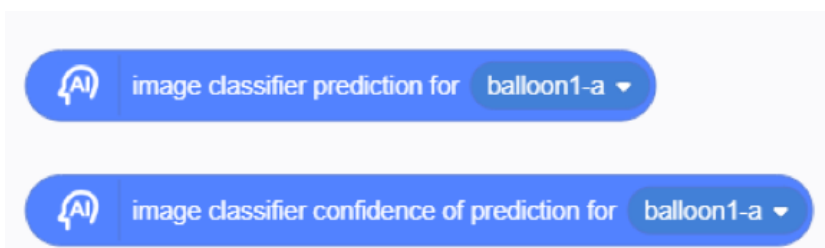
## 4. Posture detection on video



This building block can perform real-time pose prediction on the human body appearing in the video stream.

Working principle: This building block uses the open source machine learning model PoseNet, which can predict poses of key positions of the human body in images or videos.

## 5. Image Classification



This set of blocks can predict the classification of character modeling images and return the corresponding confidence.

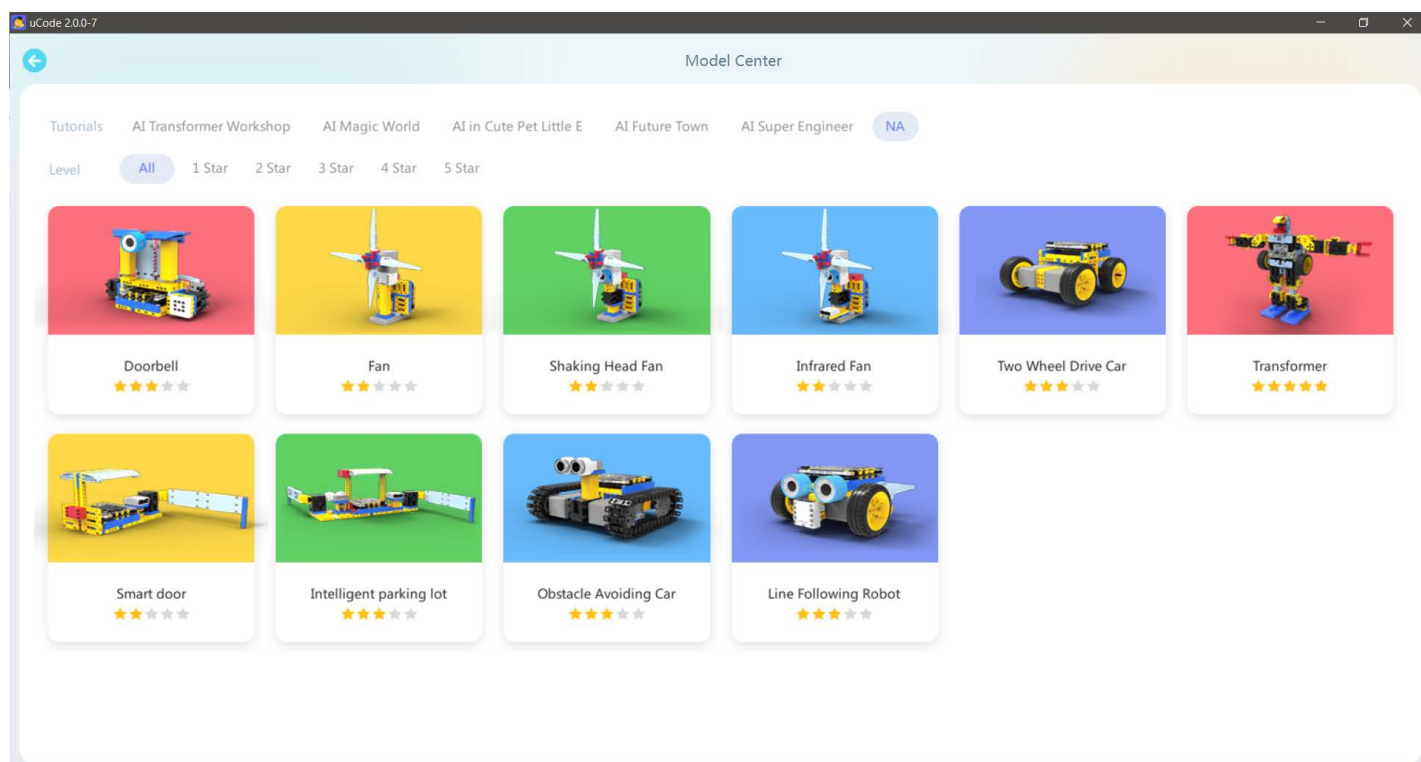
Working principle: This building block uses the open source machine learning model ImageNet, which is trained on a database of about 15 million images (ImageNet). What the model predicts depends on what is included in the training data, what is excluded, and how the images are labeled.



## 3D Building Instructions



uCode provides official custom UKIT Advanced models with 3D building instructions, 360 viewing to quickly build your own model. Users can enjoy the fun of robot building and at the same time subtly establish the thinking logic of spatial structure, laying the foundation for future AI.



# Explore connection

## How to connect with UKIT Advanced board?

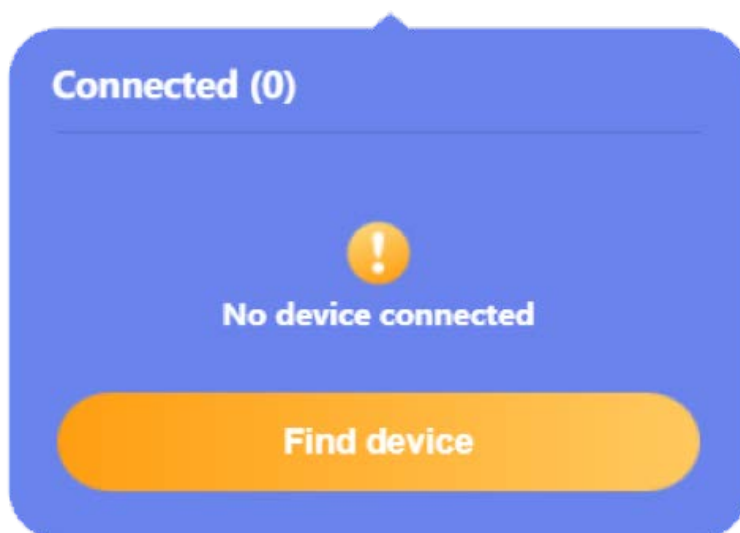
### 1. Connect the hardware

Use the data cable to connect the board to the computer. Note: the UKIT Advanced board is called *Explore*.



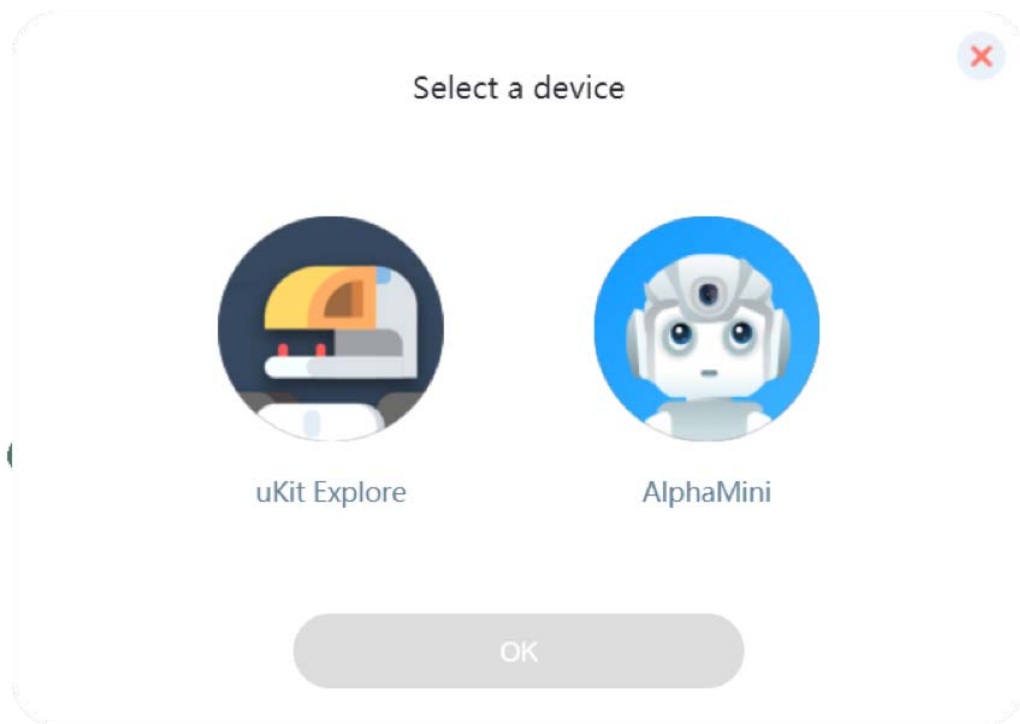
### 2. Find device

Click Connect Device-Find Device in the upper right corner of the main interface



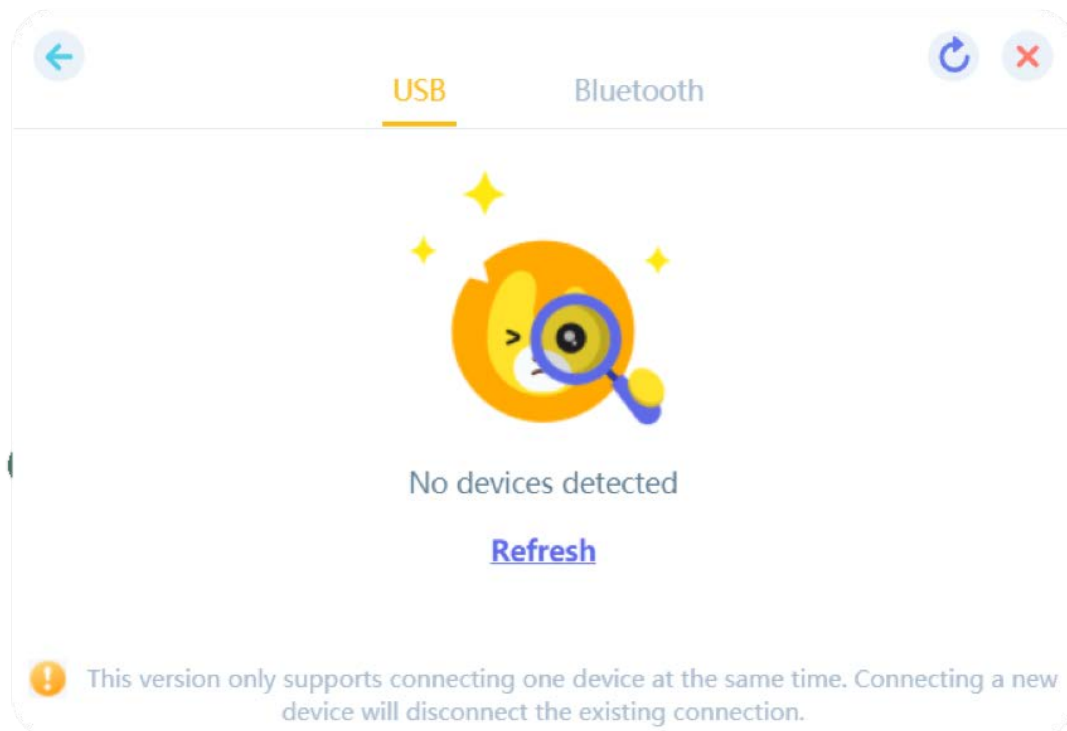
### 3. Select the hardware

Connect the device page and select uKit Explore



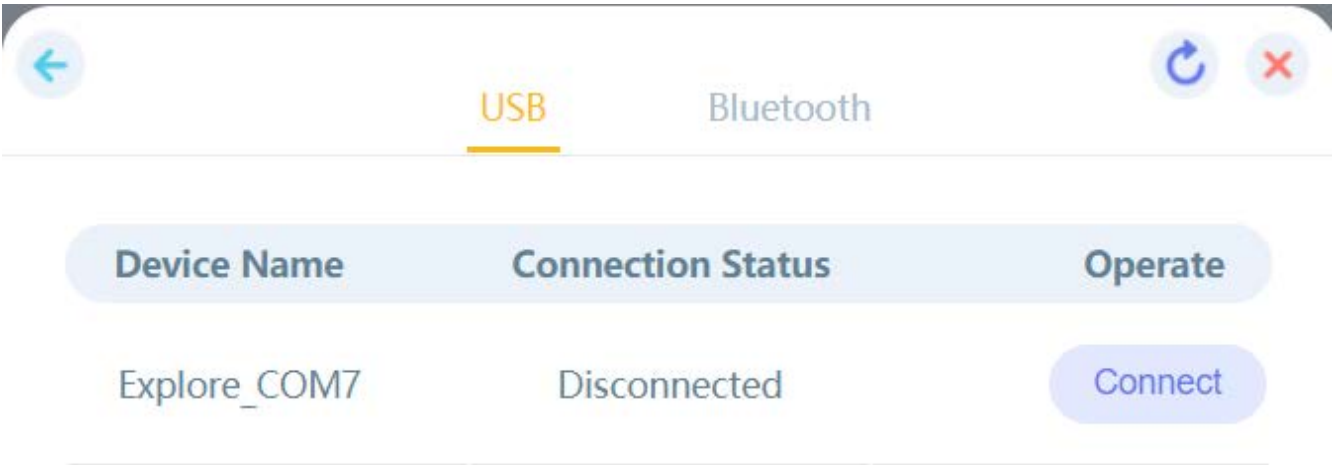
### 4. Select the connection method

Select the connection method, select USB connection or Bluetooth connection



5. Connect the device

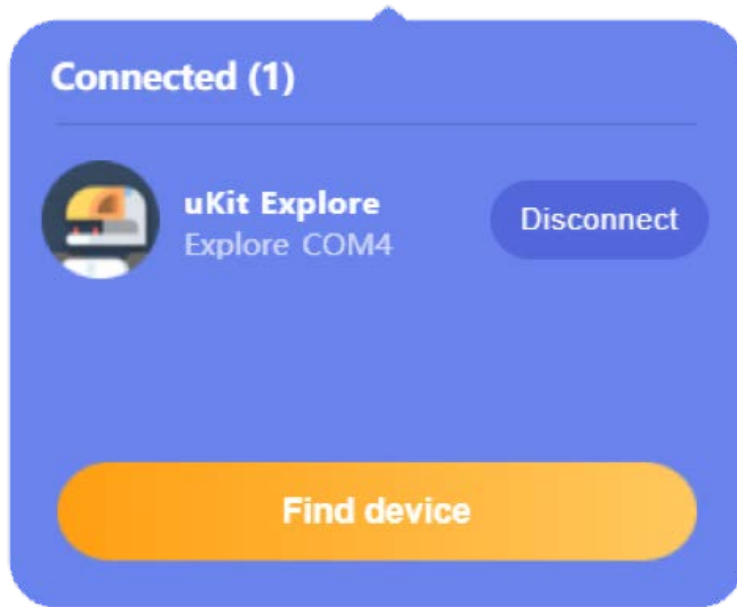
After entering the connection interface, find the device to be connected in the device list and click the Connect button



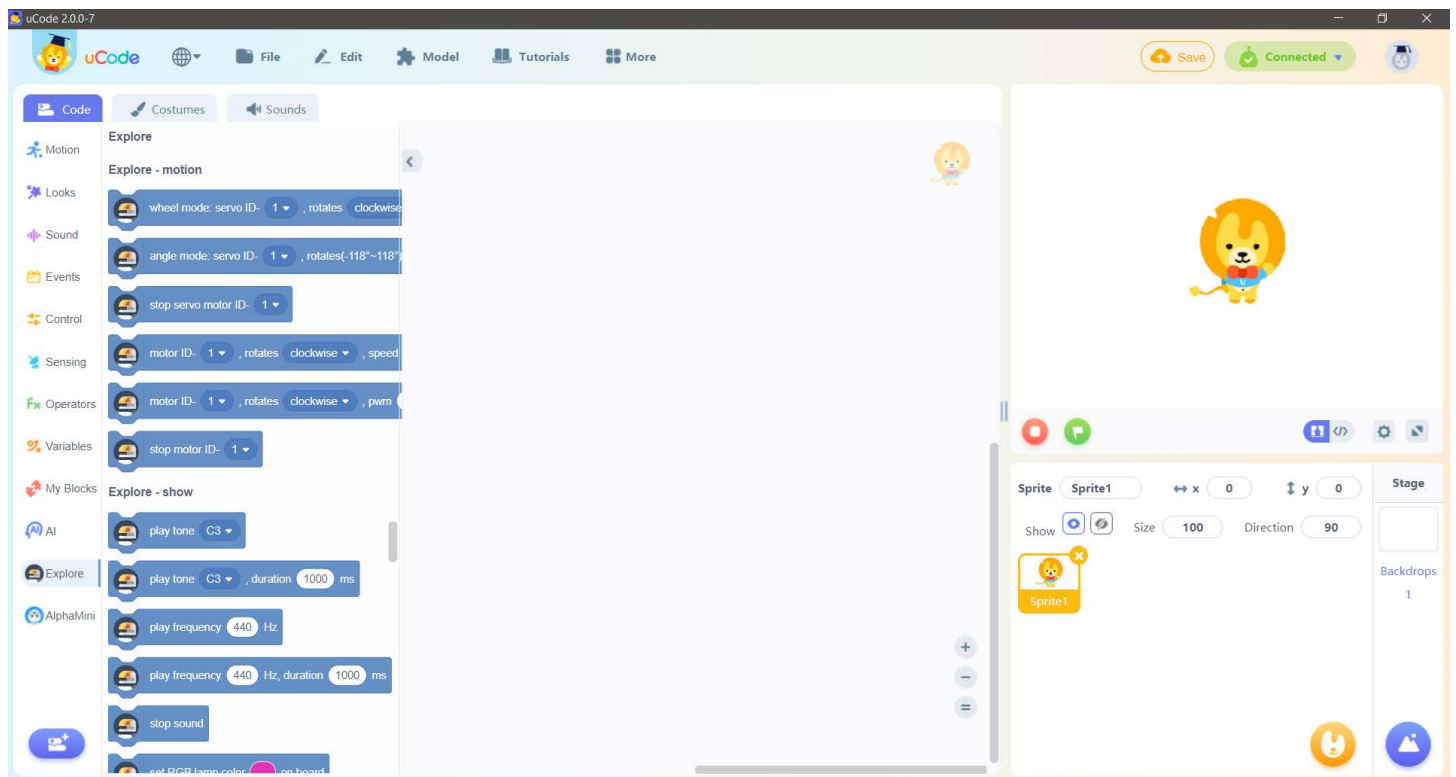
# Upload Mode-Block Description

## Upload mode

1. When the hardware is successfully connected this will appear:



2. Click the Explore category under the code tab, and click the "Upload Mode" button above the stage area on the right



3. In upload mode, after programming with block code, click the "Upload" button to start programming and uploading to the Explore microcontroller.

