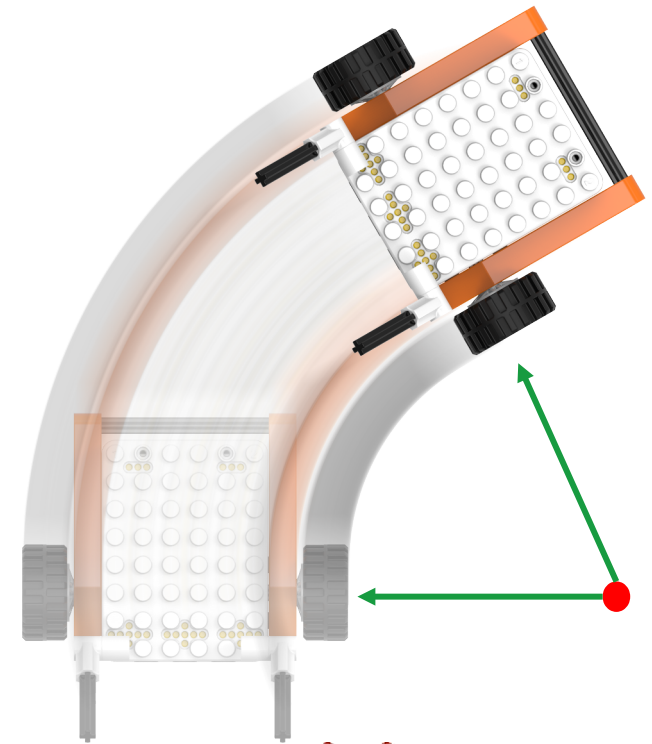
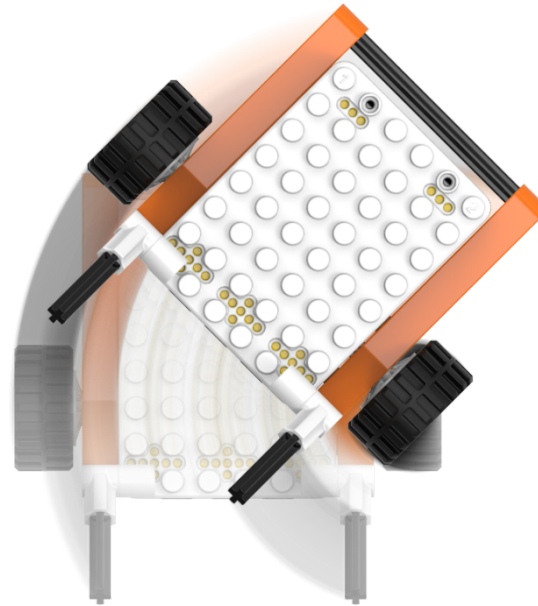
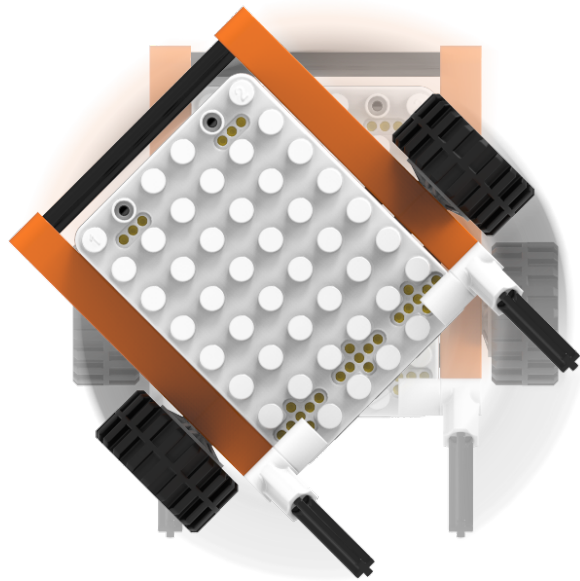




Course Review

1. Module Explanation

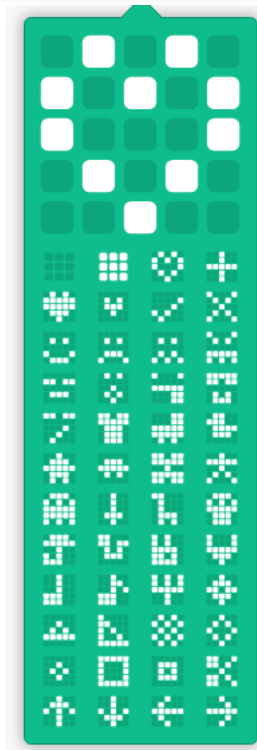
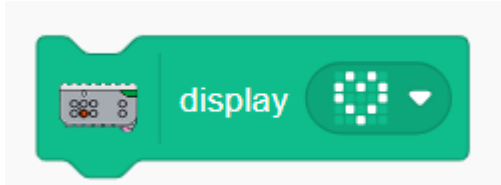
When making a U-turn, choose the appropriate turning mode.



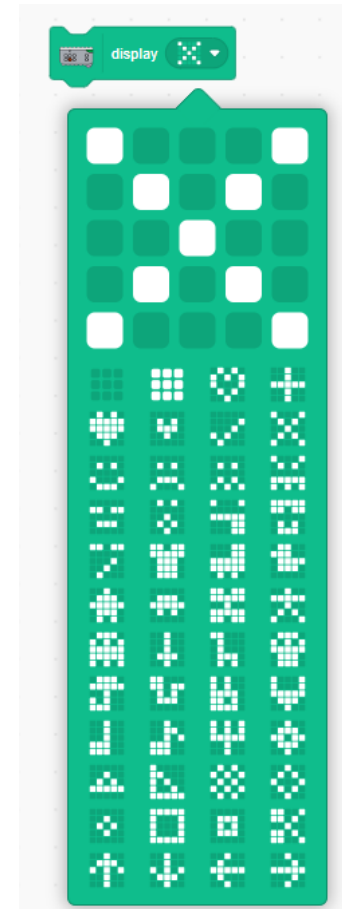


Course Review

2. LED Light Module



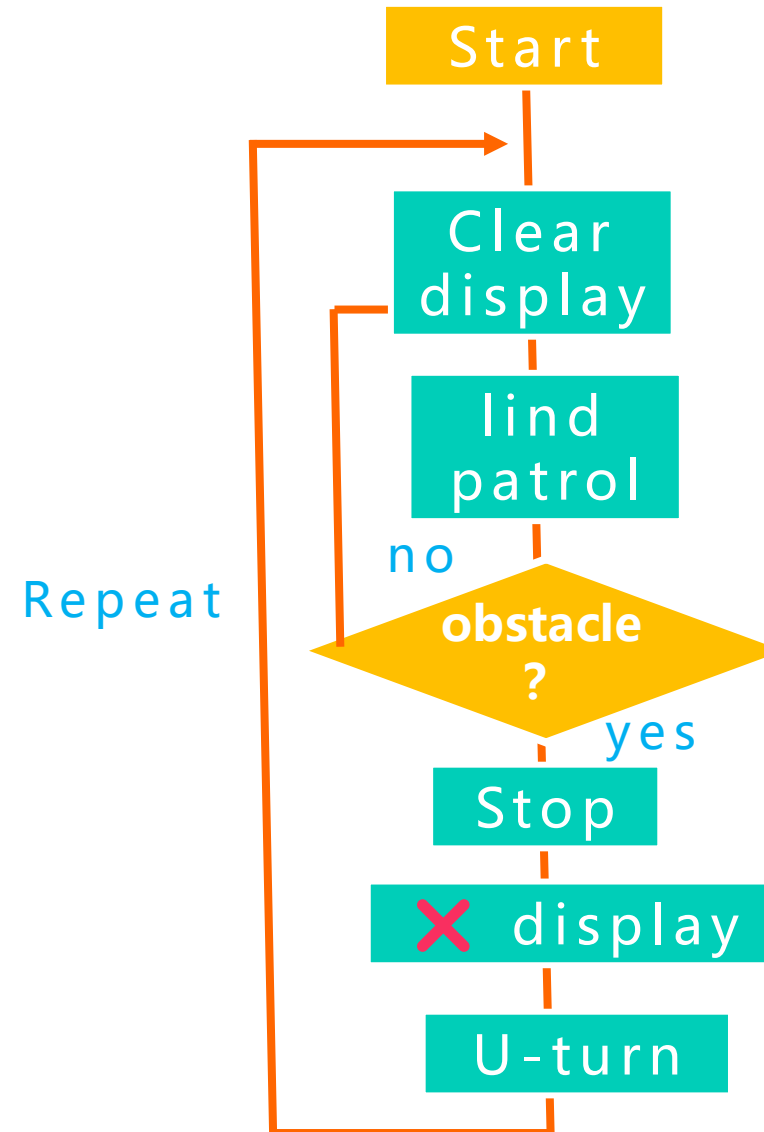
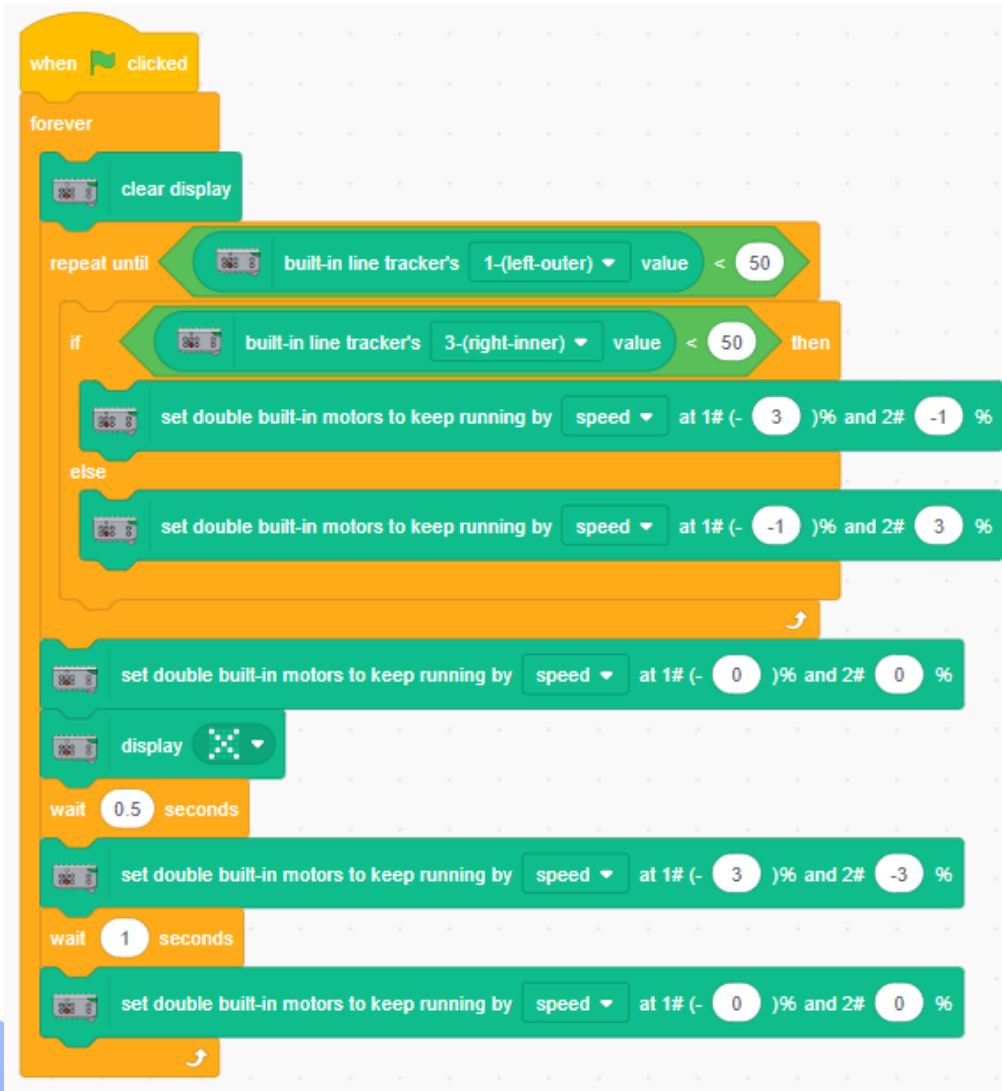
Use the LED display module to draw the pattern you need.





Course Review

3. Program Analysis





Course Review

4. Make a block

```
define go straight
  if built-in line tracker's 3-(right-inner) value < 50 then
    set double built-in motors to keep running by speed at 1# (-3)% and 2# (-1)%
  else
    set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
```

```
define turn
  set double built-in motors to keep running by speed at 1# (-3)% and 2# (-3)%
  wait 1 seconds
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
```

```
when clicked
  forever
    clear display
    repeat until built-in line tracker's 1-(left-outer) value < 50
      go straight
    set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
    display
    wait 0.5 seconds
    turn
```

Please use your own kit.

Do not put any parts in your mouth.

Please clean up after use.

Please raise your hand if you have any questions.



INTRODUCTION





Scenarios

In the previous lesson, we learned how to make the robot car patrol. In this lesson, we will have the robot car follow the black line while also carrying cargo.



Dear kids:

How can the robot car detect cargo in front of it?

How can the robot store the cargo?

Let's embark on our journey of exploration together with the "Patrol Car"!

Carrying Competition

AI Courses

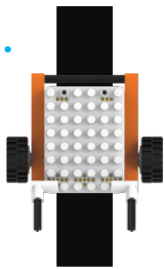




Scenarios

Competition rules:

1. The contestant's robot car starts from the starting point and automatically follows the black line to the loading point. The car picks up the cargo and transports it back to the starting point.
2. The robot car must not deviate from the black line, and it can be manually adjusted in the dashed area (starting point).
3. Let's see whose robot car can transport the most cargo within 2 minutes.



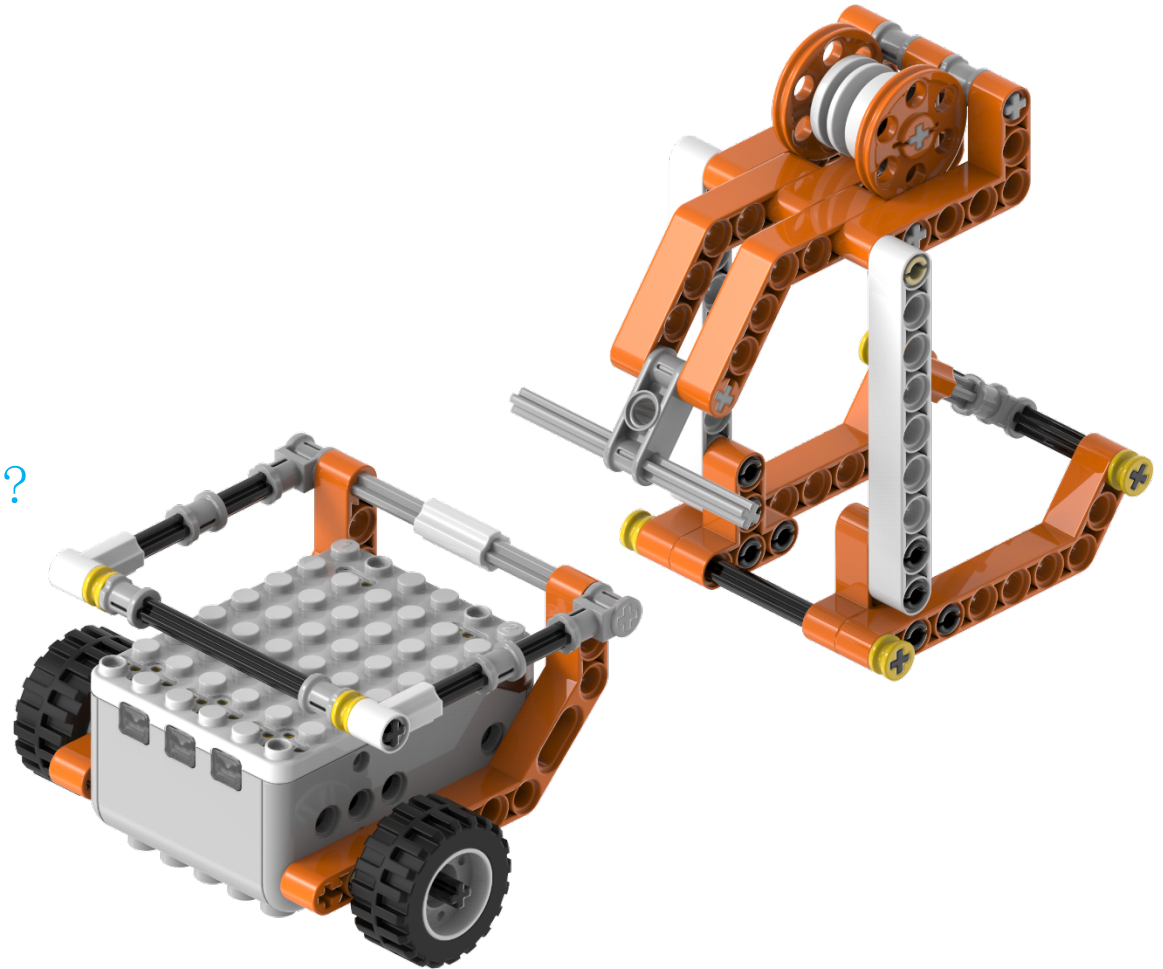


Scenarios

Question :

Kids, do you know:

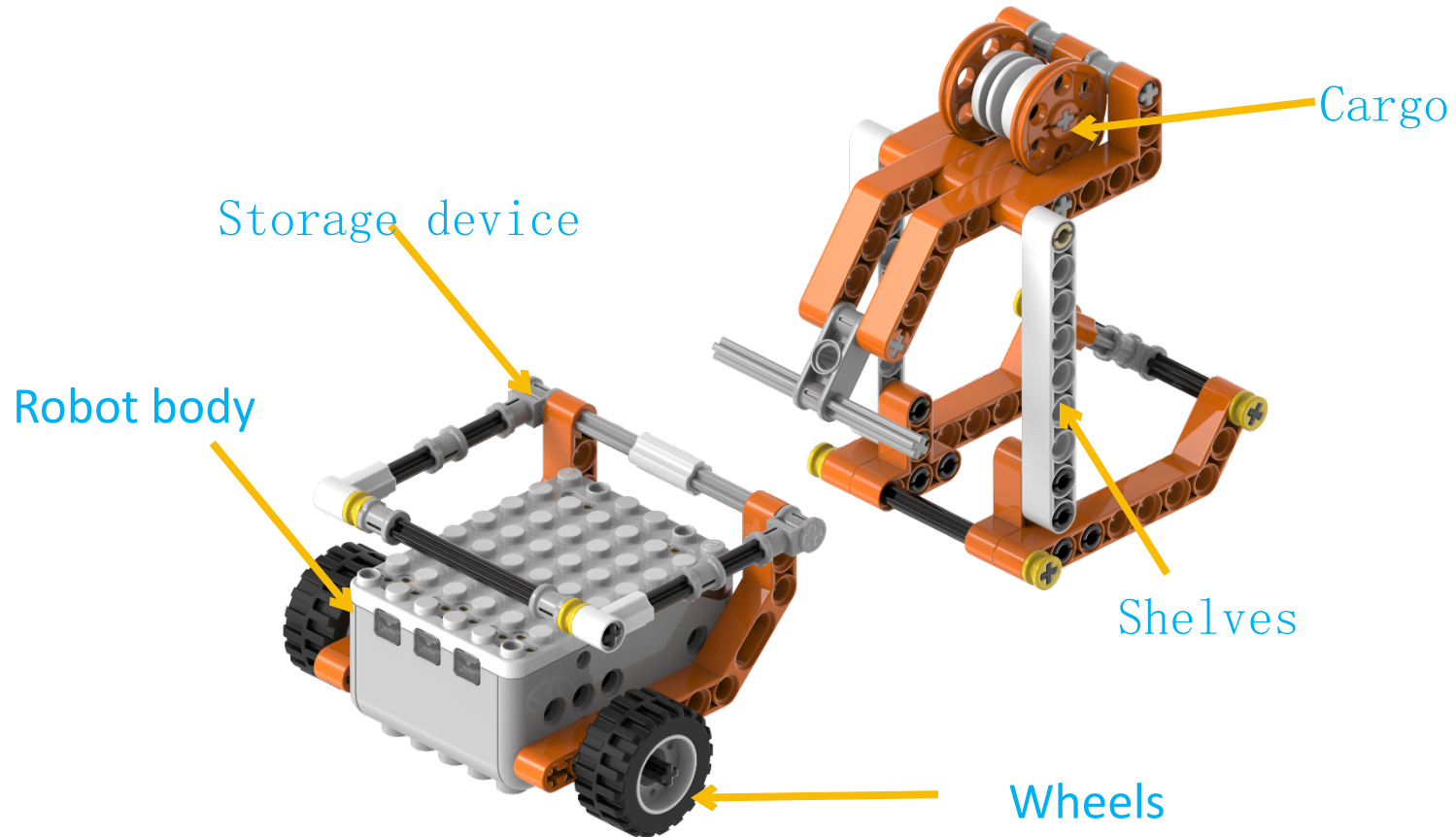
1. How can the robot car detect the cargo in front of it?
2. How can the robot store the cargo?





Scenarios

Today, everyone is a junior engineer. Let's work together to complete the gesture control course!



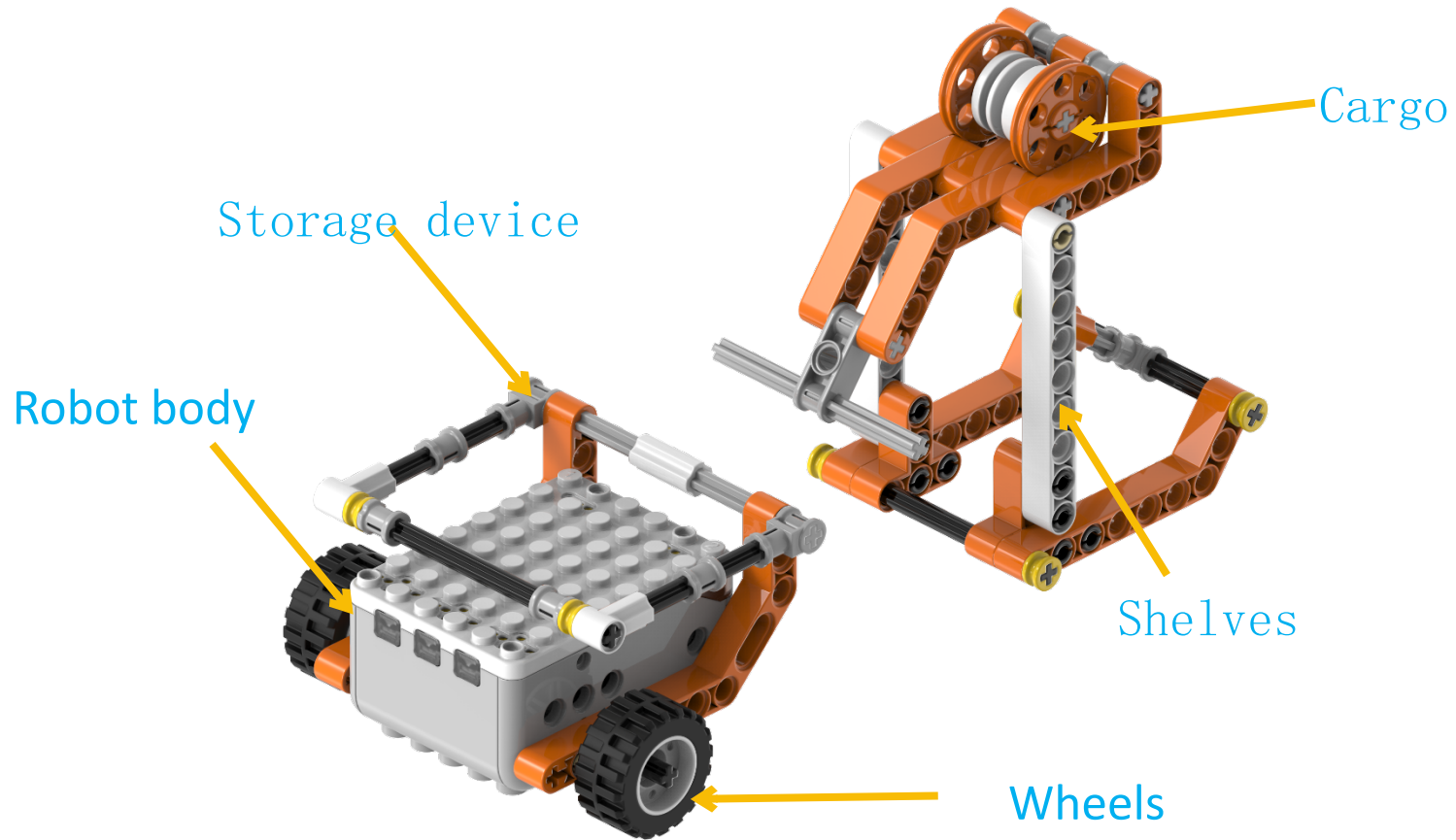
ASSEMBLY





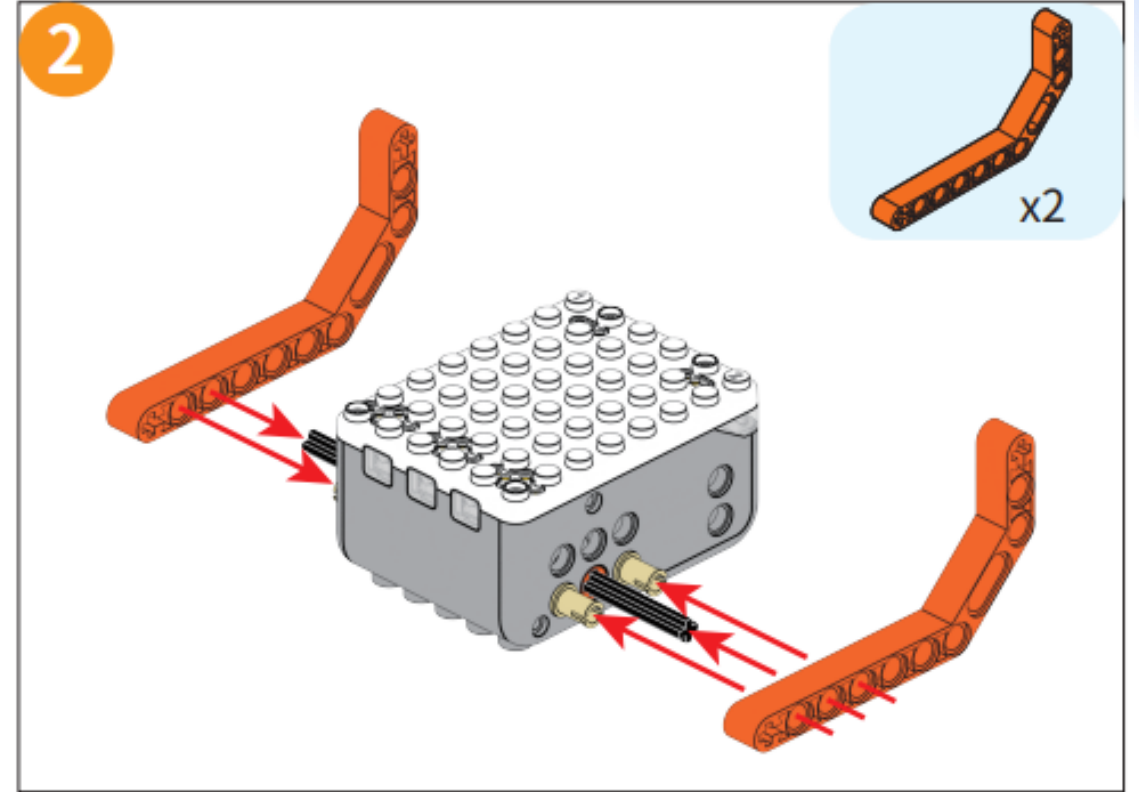
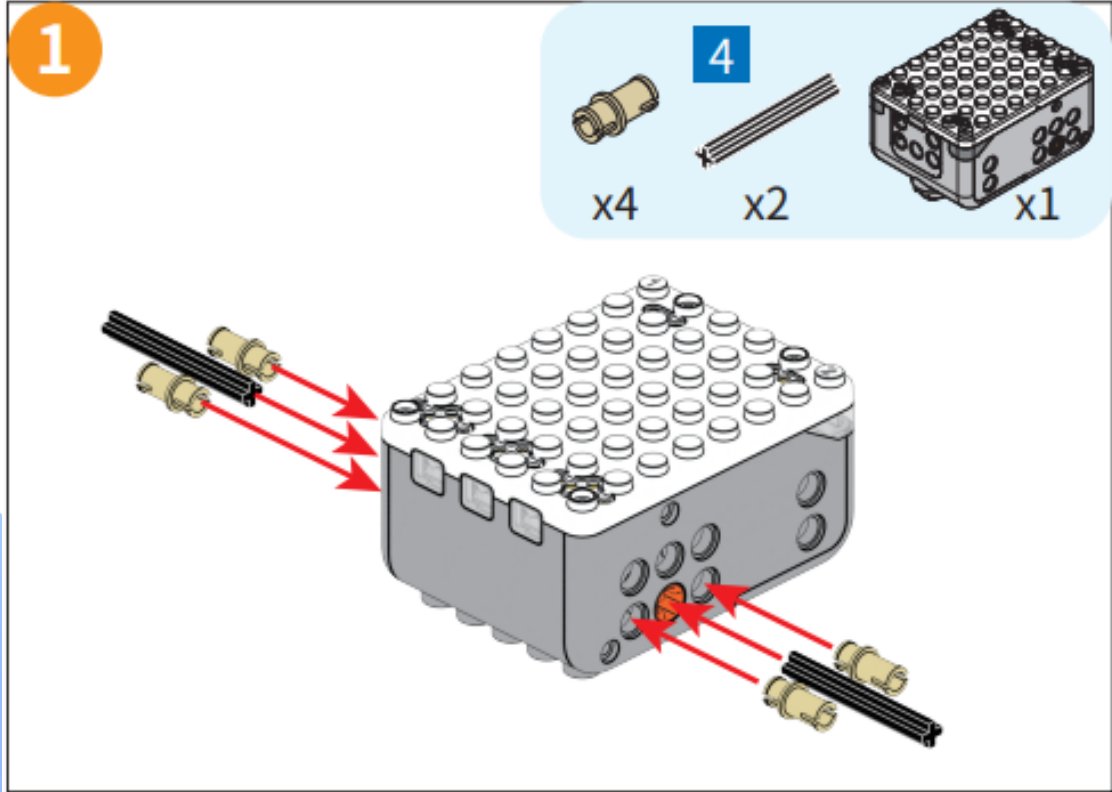
The Final Model

Robot body assembly





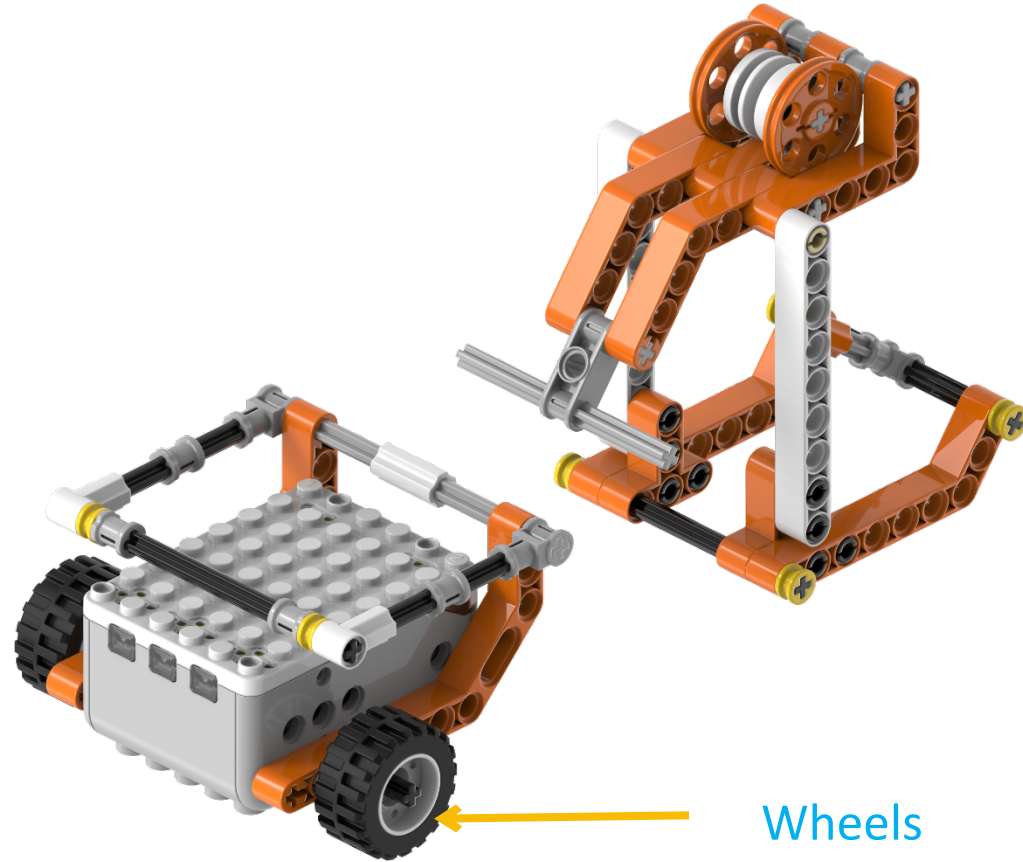
Assembly





Assembly

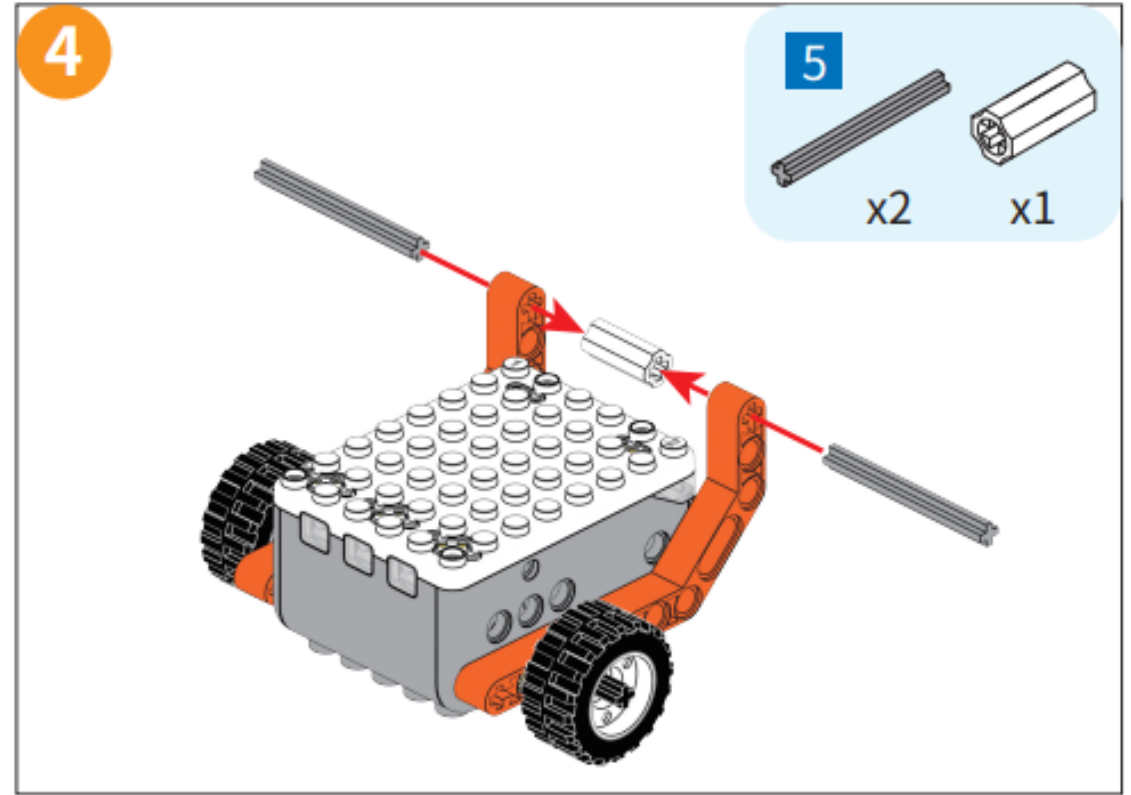
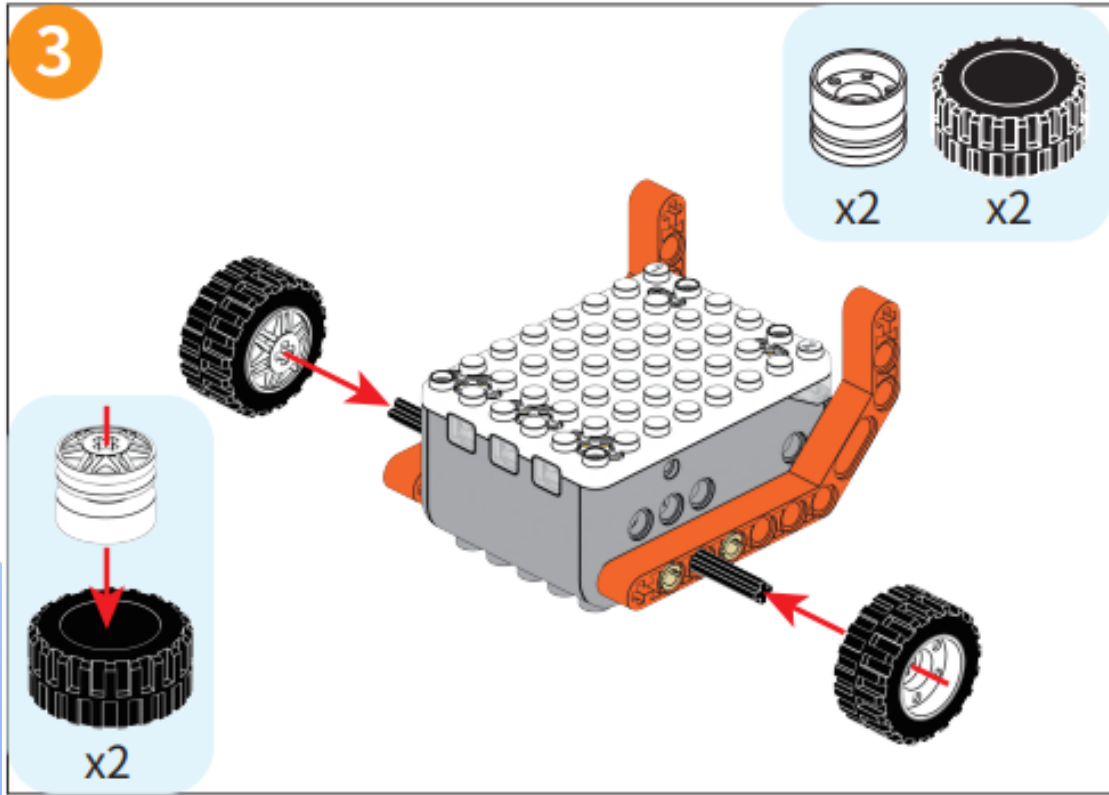
Wheels assembly



Wheels



Assembly

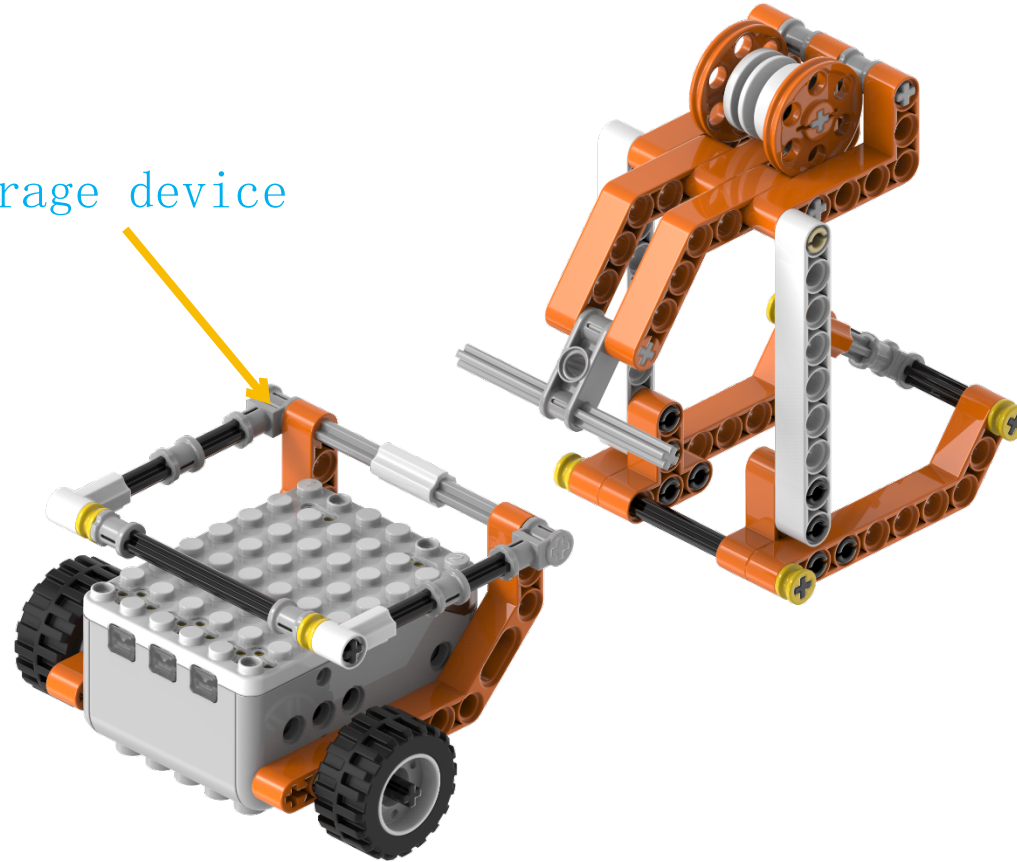




Assembly

Storage device assembly

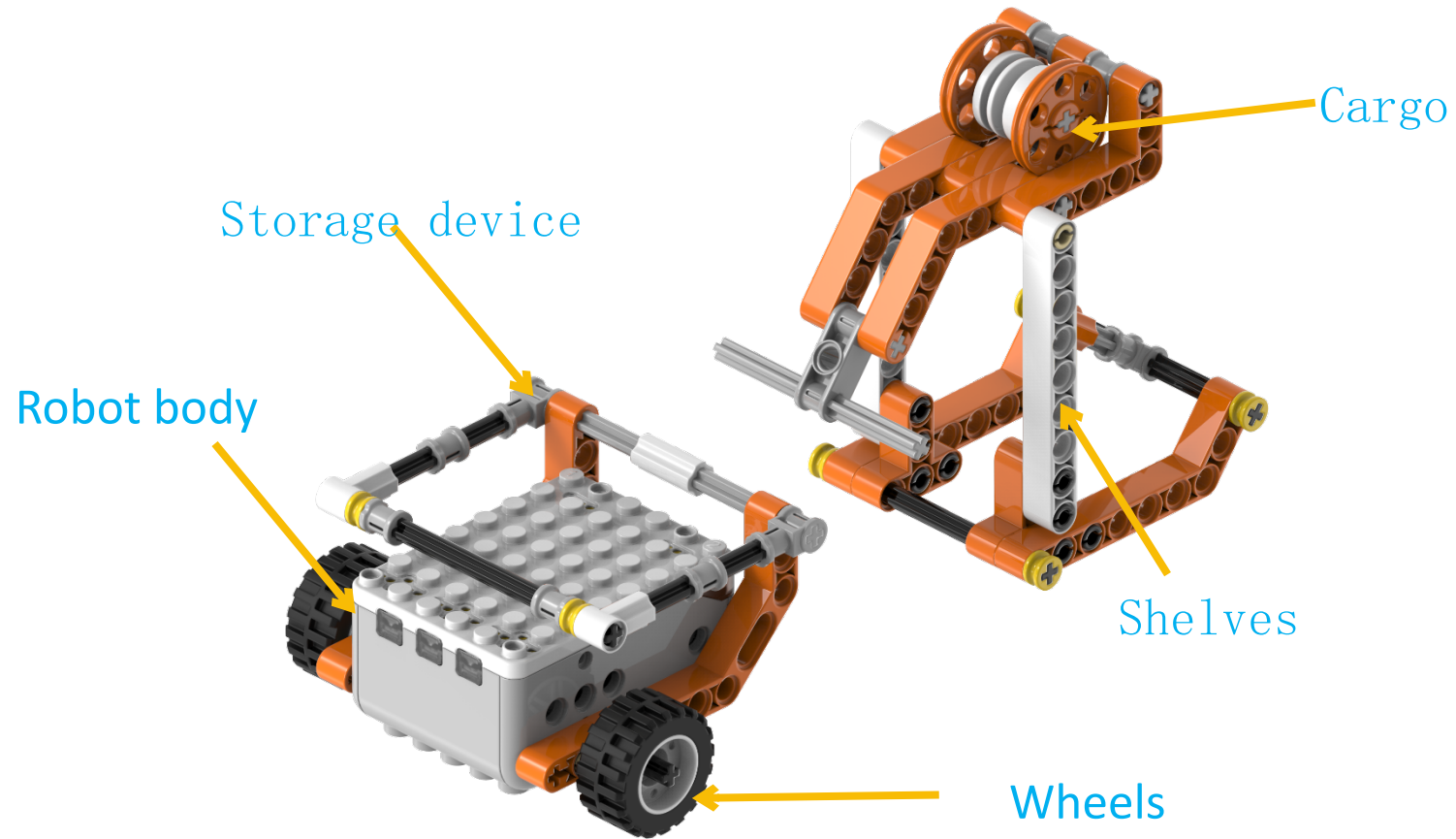
Storage device





Assembly

Shelves and cargo assembly





Assembly

9

Parts list:

- Orange L-shaped Technic beam (3x3 holes): x2
- Black Technic pin: x4
- Orange L-shaped Technic beam (1x7 holes): x2

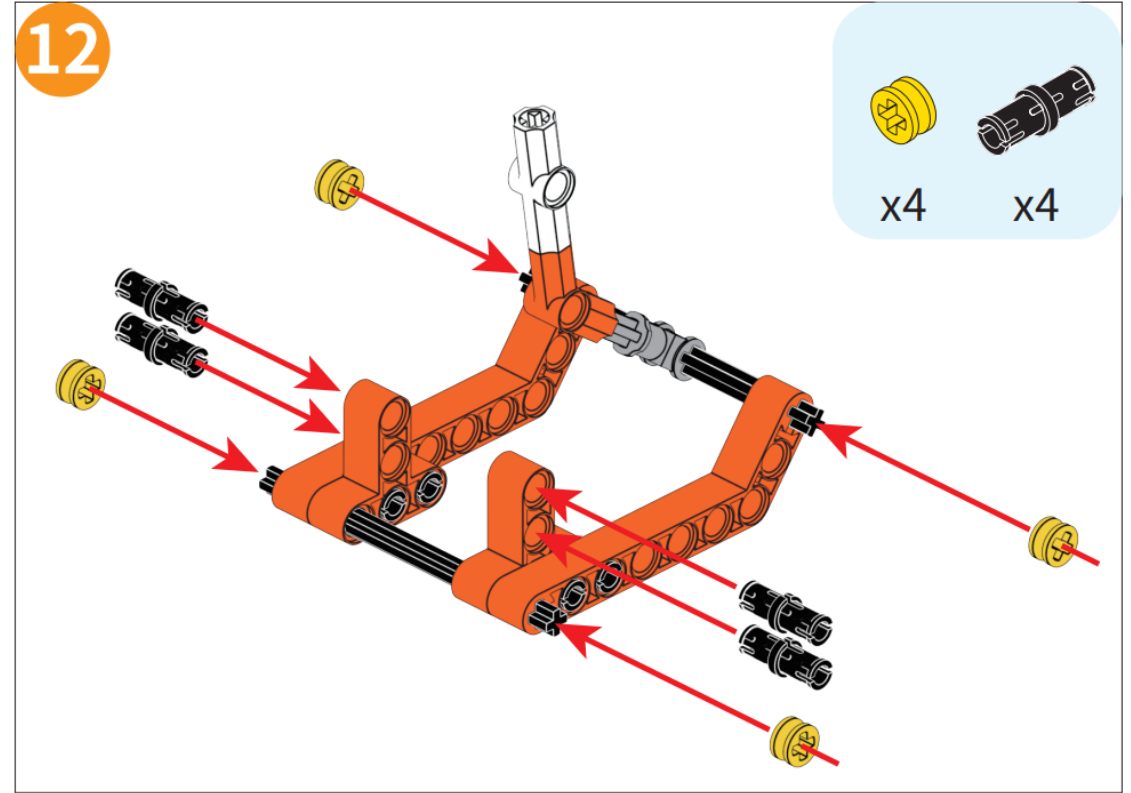
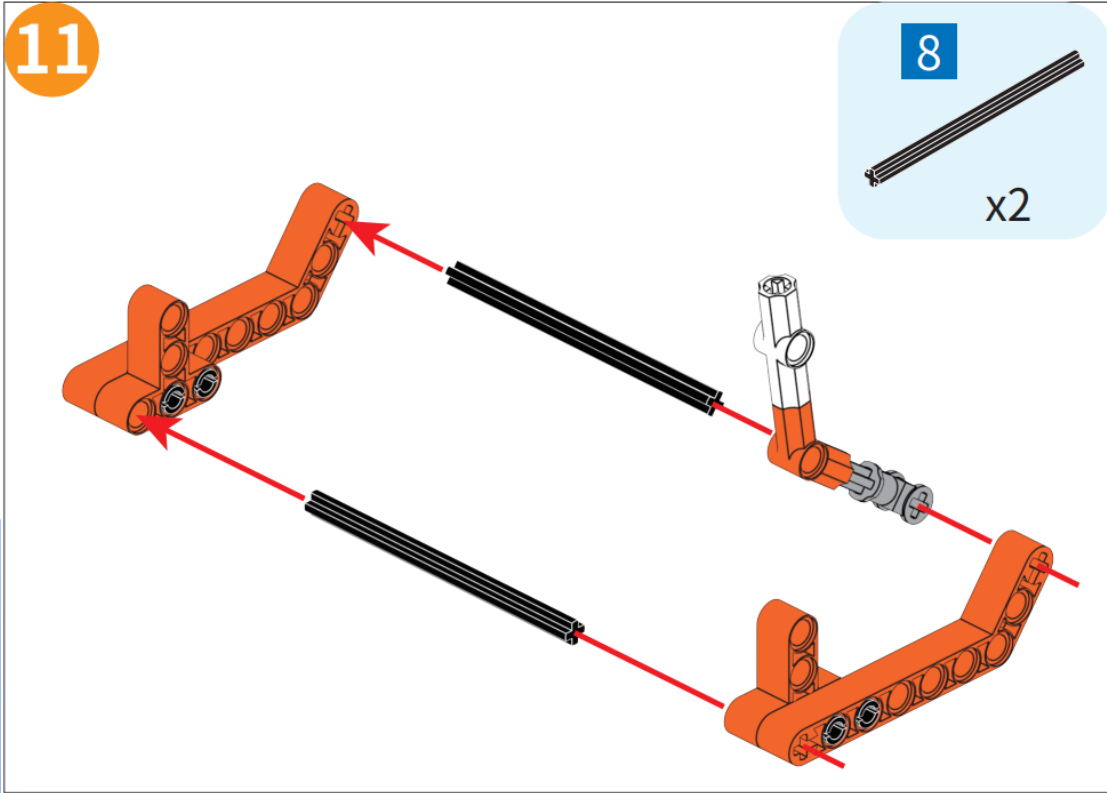
10

Parts list:

- Red Technic pin: x1
- Grey Technic axle: x1
- Grey Technic pin: x1
- White Technic pin: x1
- Orange L-shaped Technic beam (3x3 holes): x1
- Orange L-shaped Technic beam (1x7 holes): x1



Assembly





Assembly

13

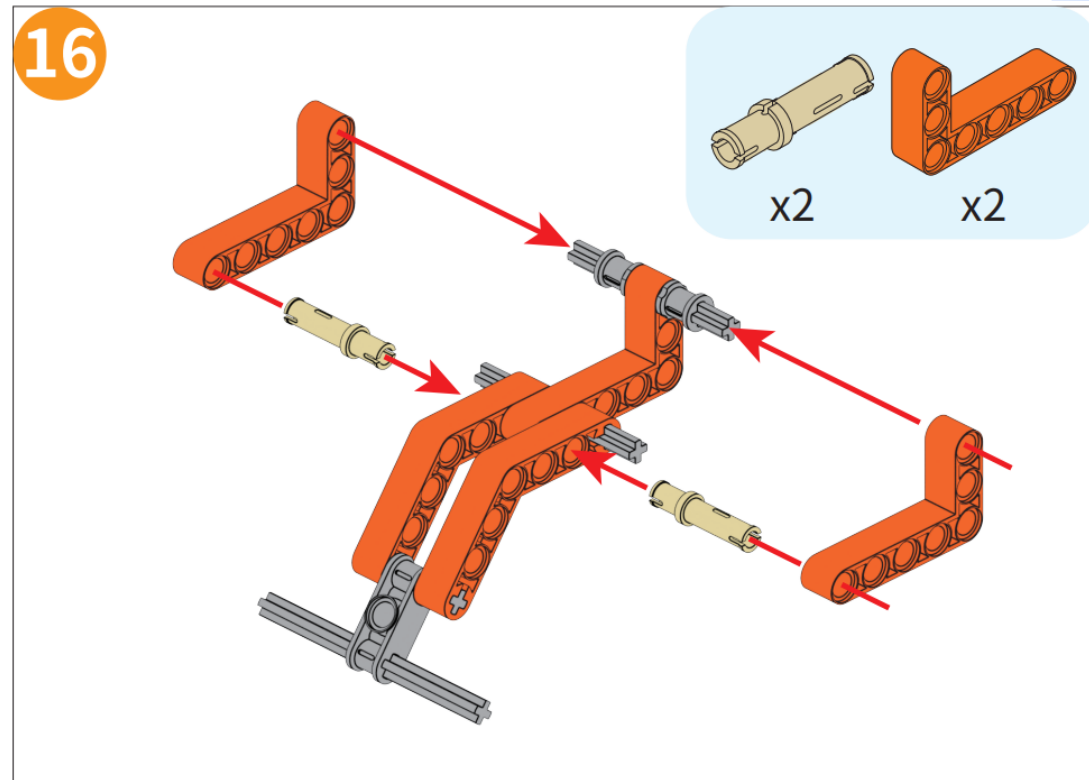
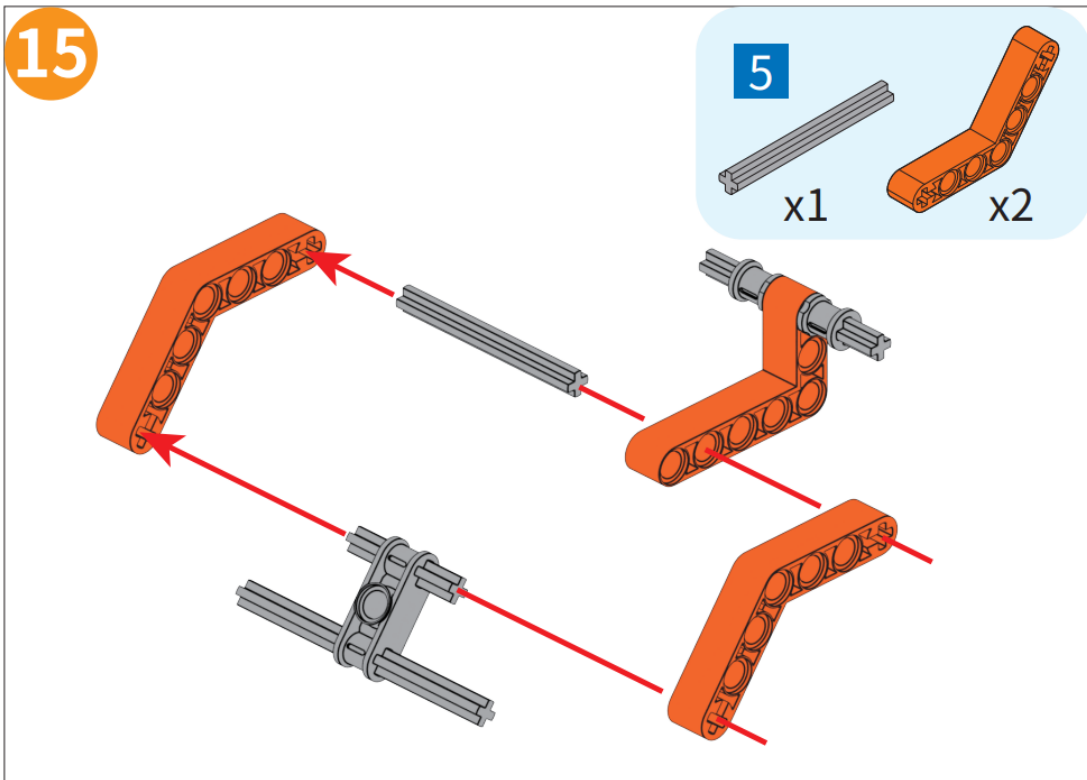
5
x2 x1 x1

14

3 **7**
x1 x1 x1

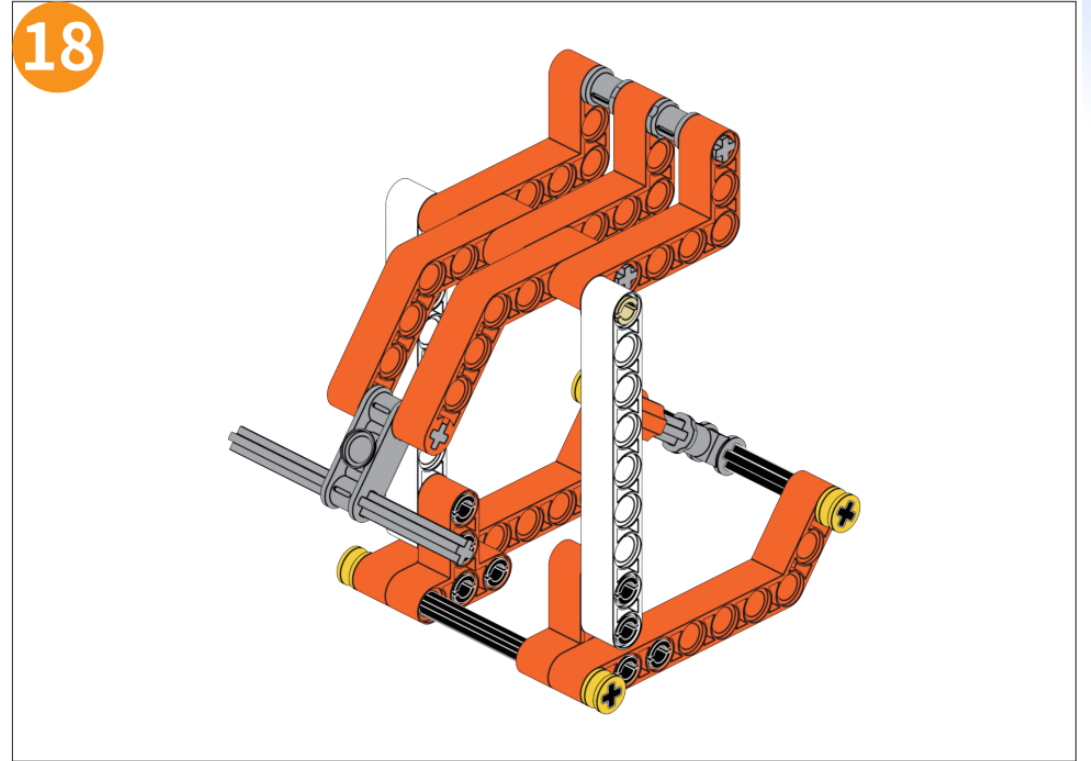
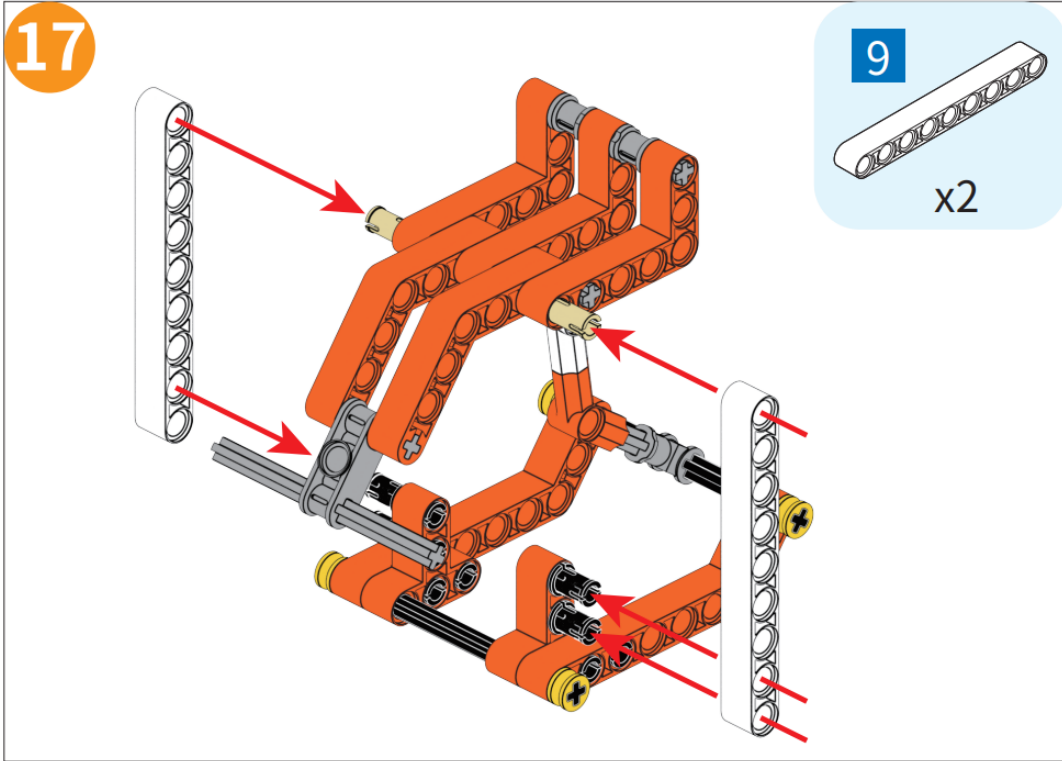


Assembly





Assembly





Assembly

19

3

x1 x1 x2

This diagram illustrates step 19 of the assembly process. It shows a grey axle being inserted into a white hub. The hub is already mounted on an orange gear. A second orange gear is shown to the right, with a red arrow indicating it should be placed on the axle. A legend in the top right corner of the diagram area lists the required parts: 3 grey axles (x1), 1 white hub (x1), and 2 orange gears (x2).

20

This diagram illustrates step 20 of the assembly process. It shows a completed sub-assembly being attached to the main robot chassis. The sub-assembly consists of orange beams, grey axles, and a white hub. The main chassis is shown in a perspective view, with the sub-assembly being positioned to be attached to the rear axle.

PROGRAMMING





Introductions

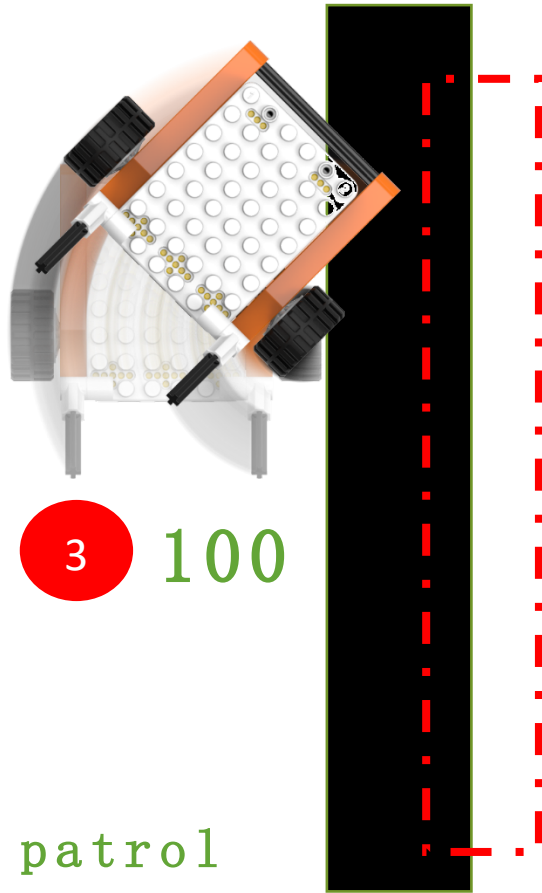
Module Explanation





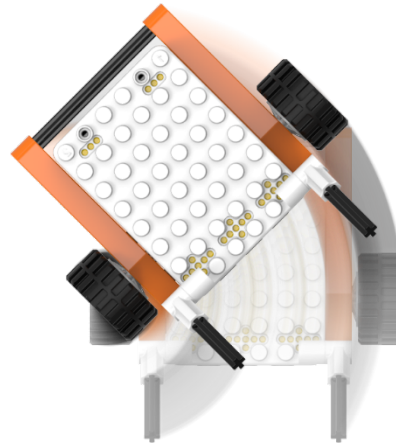
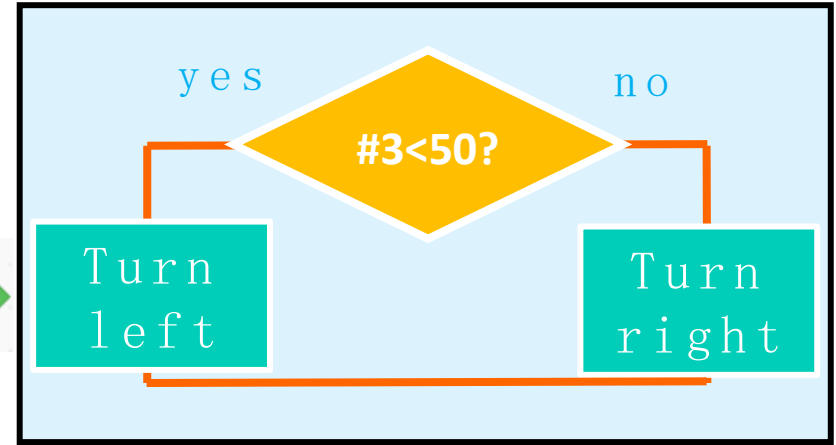
Introductions

Program Explanation



```

built-in line tracker's 3-(right-inner) value < 50
  
```



```

if [built-in line tracker's 3-(right-inner) value < 50] then
  set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
else
  set double built-in motors to keep running by speed at 1# (3)% and 2# 1%
  
```

Right line patrol

$$\text{average value} = (100 + 0) / 2 = 50$$

3 0

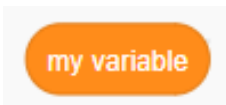


Introductions

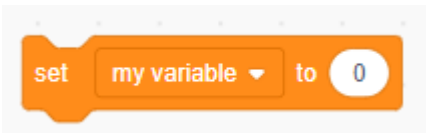
Program Explanation



Make the subroutine loop a certain number of times.



A variable is a quantity that changes during the course of a program.



A variable can be assigned or set.



Multiply two sets of data and output the result.



Introductions

Program Explanation

Make the robot follow a line for a certain distance.

```
when clicked
repeat 10
  if built-in line tracker's 3-(right-inner) value < 50 then
    set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
  else
    set double built-in motors to keep running by speed at 1# (-3)% and 2# 1%
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
wait 0.2 seconds
```

```
when clicked
set my variable 25
repeat my variable * 500
  if built-in line tracker's 3-(right-inner) value < 50 then
    set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
  else
    set double built-in motors to keep running by speed at 1# (-3)% and 2# 1%
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
wait 0.2 seconds
```

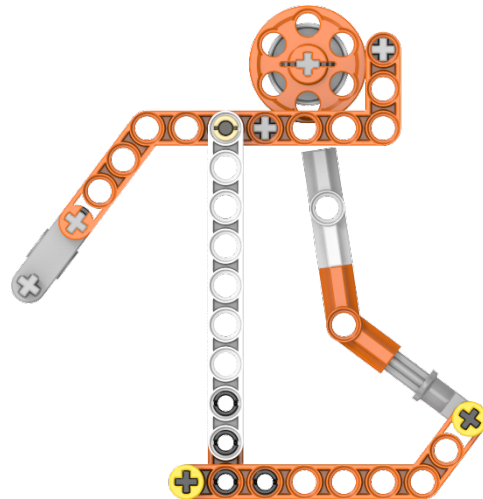
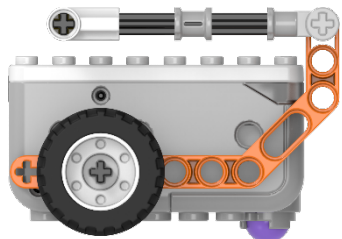
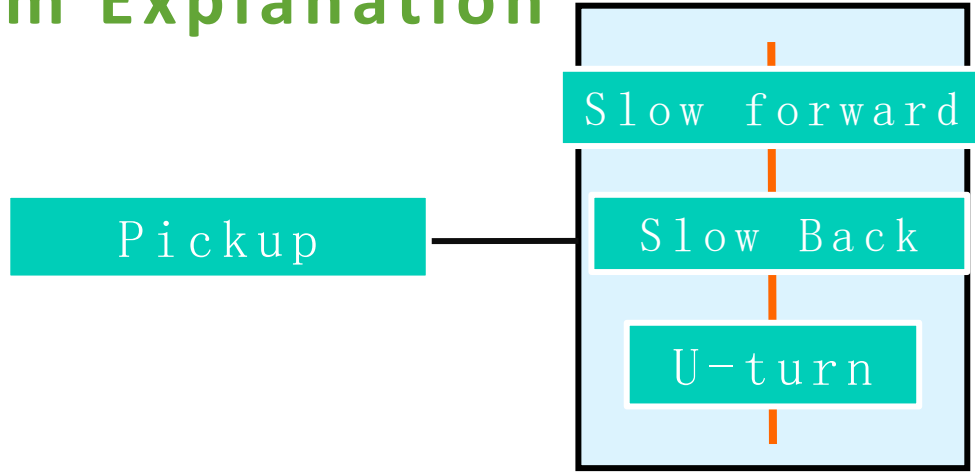
Adjusting this value allows you to control the robot's line-following distance.

Running the program offline showed that the robot barely moved. After testing, we found that 500 iterations allow the robot to move a certain distance, so we will use 500 as a unit.



Introductions

Program Explanation



```

    set double built-in motors to keep running by speed at 1# (- 0 )% and 2# 0 %
    set double built-in motors to keep running by speed at 1# (- -2 )% and 2# -2 %
    wait 1 seconds
    set double built-in motors to keep running by speed at 1# (- 0 )% and 2# 0 %
    set double built-in motors to keep running by speed at 1# (- -3 )% and 2# -3 %
    wait 1.5 seconds
    set double built-in motors to keep running by speed at 1# (- 0 )% and 2# 0 %
  
```



Introductions

Module Explanation

Simplify the program by using custom blocks.

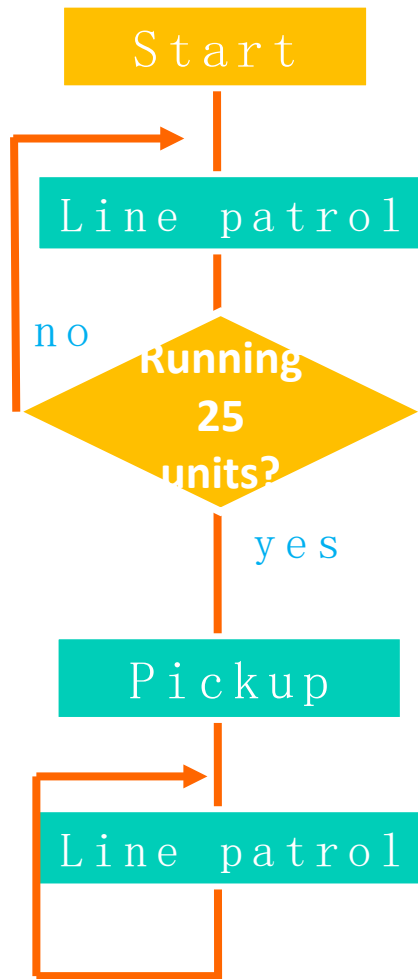
```
define Go straight
if built-in line tracker's 3-(right-inner) value < 50 then
  set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
else
  set double built-in motors to keep running by speed at 1# (-3)% and 2# 1%
```

```
define Get
  set double built-in motors to keep running by speed at 1# (-1)% and 2# 1%
  wait 4 seconds
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
  set double built-in motors to keep running by speed at 1# (-2)% and 2# -2%
  wait 1 seconds
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
  set double built-in motors to keep running by speed at 1# (-3)% and 2# 3%
  wait 1.5 seconds
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
```



Introductions

Module Explanation



```
when green flag clicked
  set my variable to 25
  repeat (my variable * 500)
    Go straight
  set double built-in motors to keep running by speed at 1# (- 0)% and 2# 0%
  wait 0.2 seconds
  Get
  forever
    Go straight
```

The code block shows the implementation of the module. It starts with a 'when green flag clicked' event. The first block is 'set my variable to 25'. This is followed by a 'repeat' loop where the number of repetitions is 'my variable * 500'. Inside the loop is a 'Go straight' block. After the loop, there is a 'set double built-in motors to keep running by speed at 1# (- 0)% and 2# 0%' block. This is followed by a 'wait 0.2 seconds' block, a 'Get' block, and a 'forever' loop containing a 'Go straight' block.



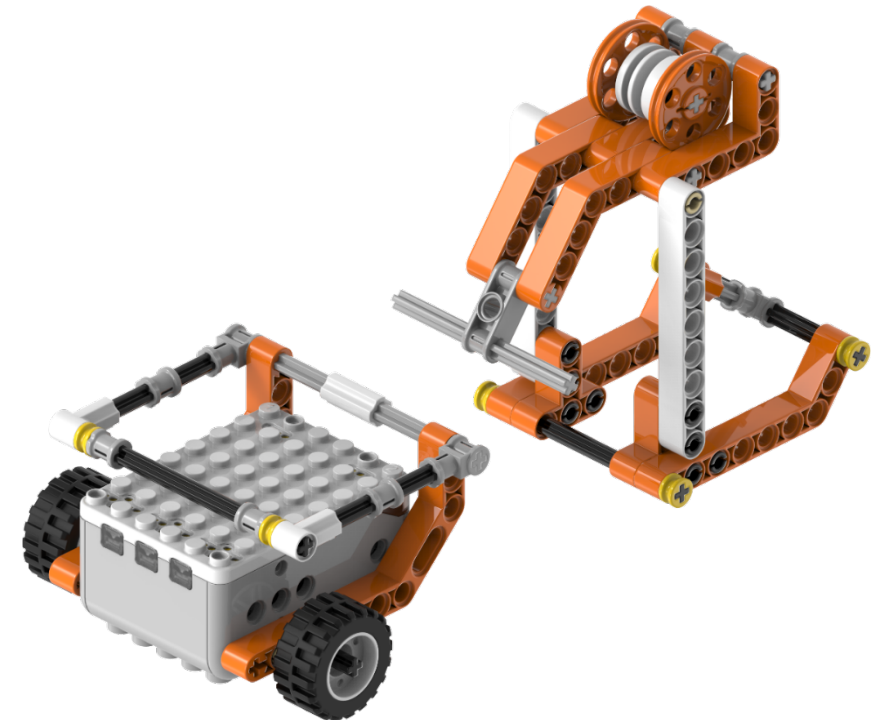
Play and Try

Let's play:

Click the Start button to see if the robot can get the goods back.



```
when clicked
  set my variable to 25
  repeat my variable * 500
    Go straight
  set double built-in motors to keep running by speed at 1# (- 0)% and 2# 0%
  wait 0.2 seconds
  Get
  forever
    Go straight
```

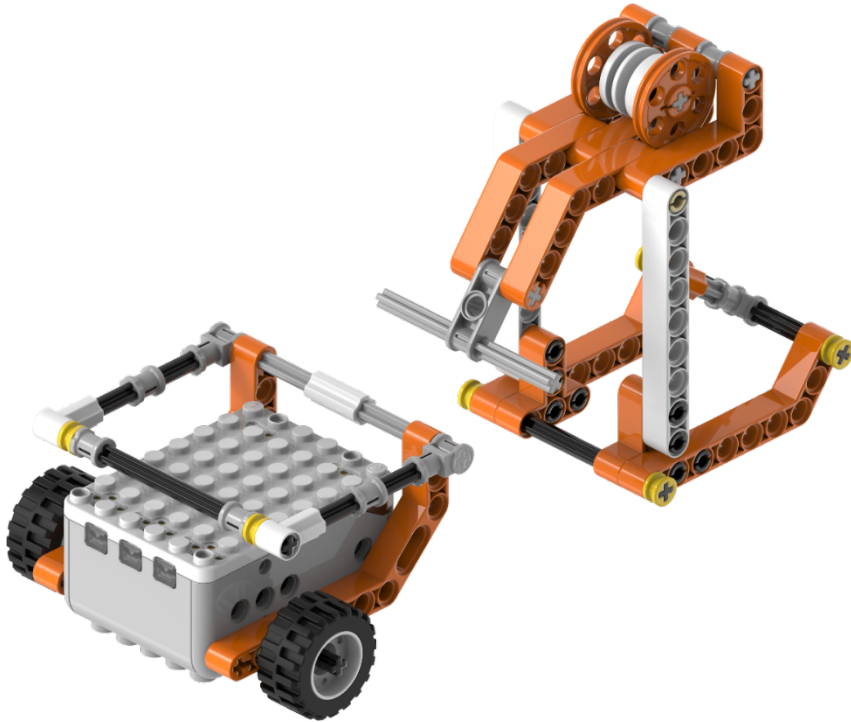




Play and Try

Let's compete:

Contestants, get ready! Let's see whose car can transport the goods back to the starting point the fastest!



Kids, is there a way to make the robot even stronger?

CREATION





Create

1. How can we make the robot repeat its task?

The robot cannot be adjusted manually anymore.

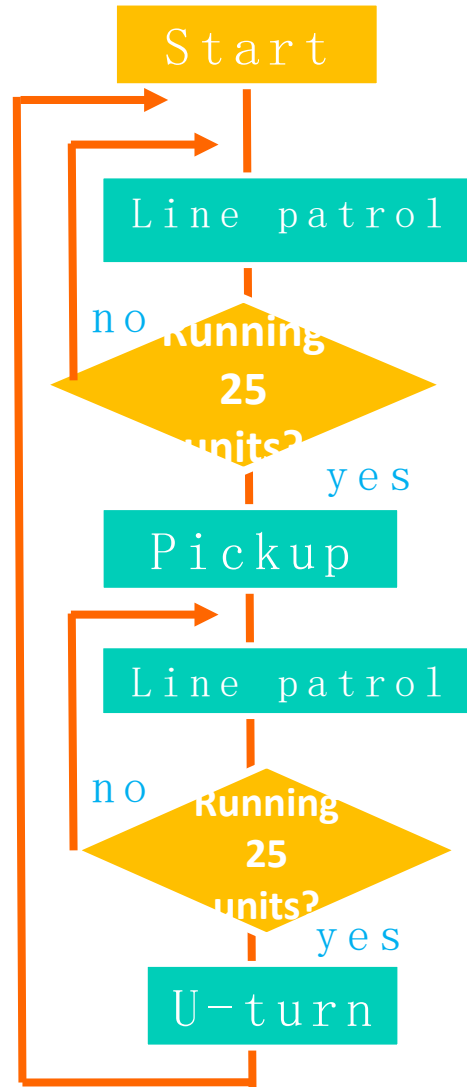




Introductions

Module Explanation

Repeat



```
when clicked
  forever
    set my variable to 25
    repeat my variable - 500
      Go straight
    set double built-in motors to keep running by speed at 1# (- 0)% and 2# 0%
    wait 0.2 seconds
    Get
    set my variable to 25
    repeat my variable - 500
      Go straight
    set double built-in motors to keep running by speed at 1# (- 0)% and 2# 0%
    wait 0.2 seconds
    set double built-in motors to keep running by speed at 1# (- -3)% and 2# 3%
    wait 0.5 seconds
    set double built-in motors to keep running by speed at 1# (- 0)% and 2# 0%
    wait 0.1 seconds
```



Create

1. How can we make the robot repeat a task?

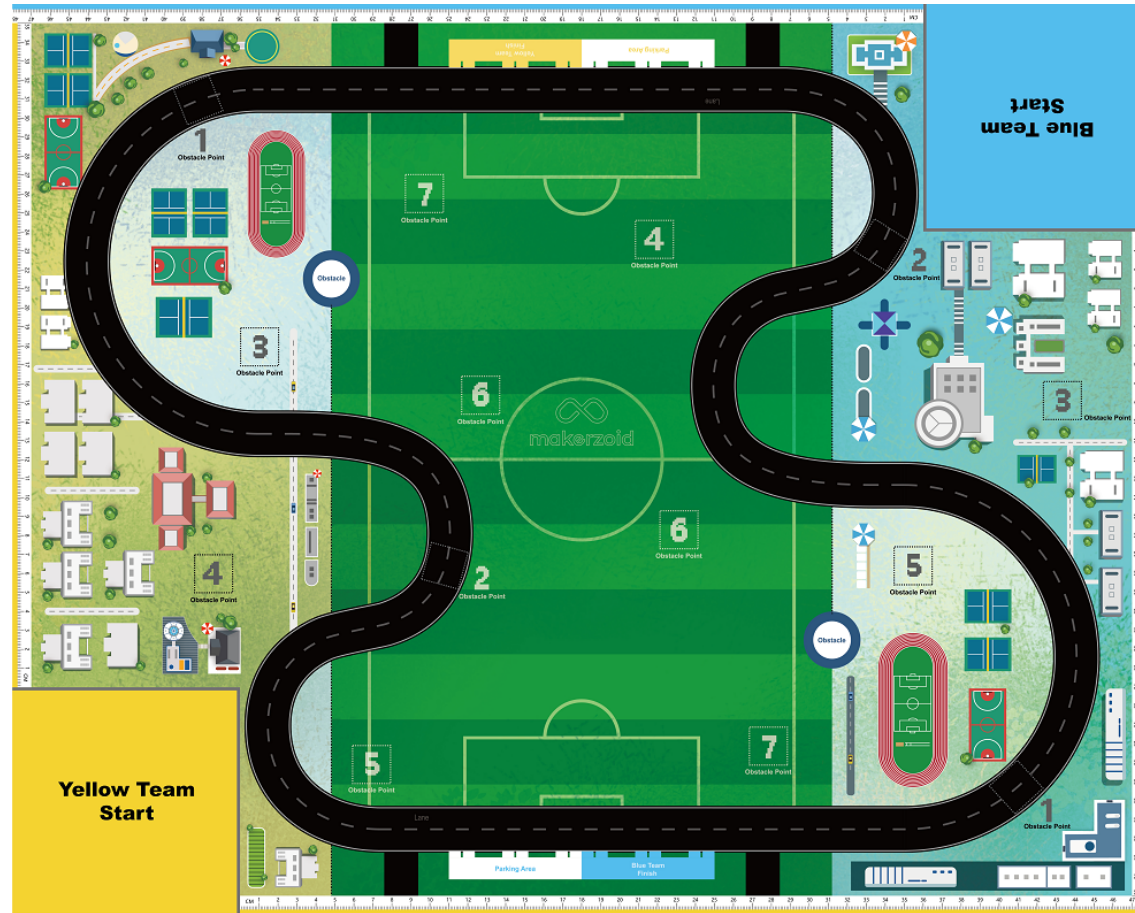
Try to make the robot complete the task automatically.





Create

2. Chasing competition





Introductions

Module Explanation



My Blocks

Sometimes a part of the program may need to be used multiple times, which can cause excessive repetition of program units. We can simplify the program by using custom blocks.

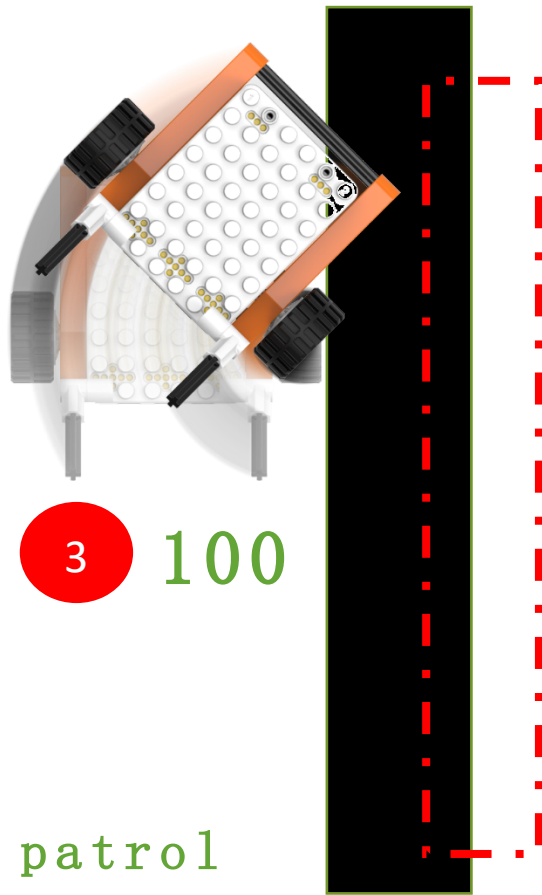
SUMMARY



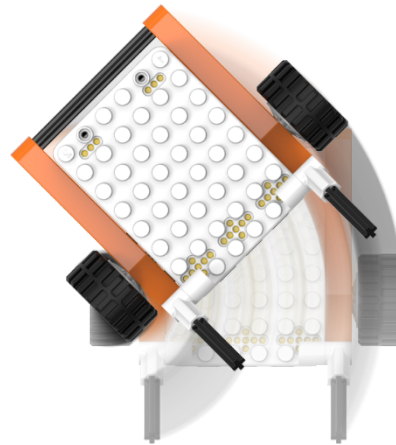
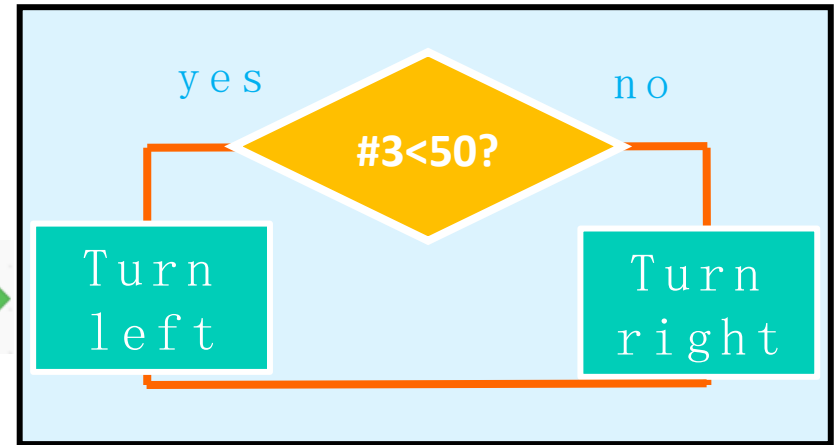


Summary

1. Right line patrol



```
built-in line tracker's 3-(right-inner) value < 50
```



```
if [built-in line tracker's 3-(right-inner) value < 50] then
  set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
else
  set double built-in motors to keep running by speed at 1# (3)% and 2# 1%
```

Right line patrol

average value = (100 + 0) / 2 = 50

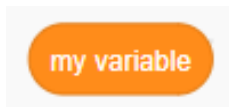


Summary

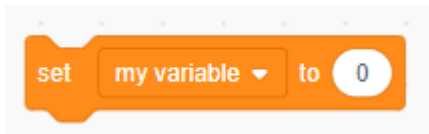
2. Module Explanation



Make the subroutine loop a certain number of times.



A variable is a quantity that changes during the course of a program.



A variable can be assigned or set.



Multiply two sets of data and output the result.



Summary

3. Line patrol distance

Make the robot follow a line for a certain distance.

```
when clicked
repeat 10
  if built-in line tracker's 3-(right-inner) value < 50 then
    set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
  else
    set double built-in motors to keep running by speed at 1# (-3)% and 2# 1%
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
wait 0.2 seconds
```

```
when clicked
set my variable 25
repeat my variable * 500
  if built-in line tracker's 3-(right-inner) value < 50 then
    set double built-in motors to keep running by speed at 1# (-1)% and 2# 3%
  else
    set double built-in motors to keep running by speed at 1# (-3)% and 2# 1%
  set double built-in motors to keep running by speed at 1# (0)% and 2# 0%
wait 0.2 seconds
```

Adjusting this value allows you to control the robot's line-following distance.

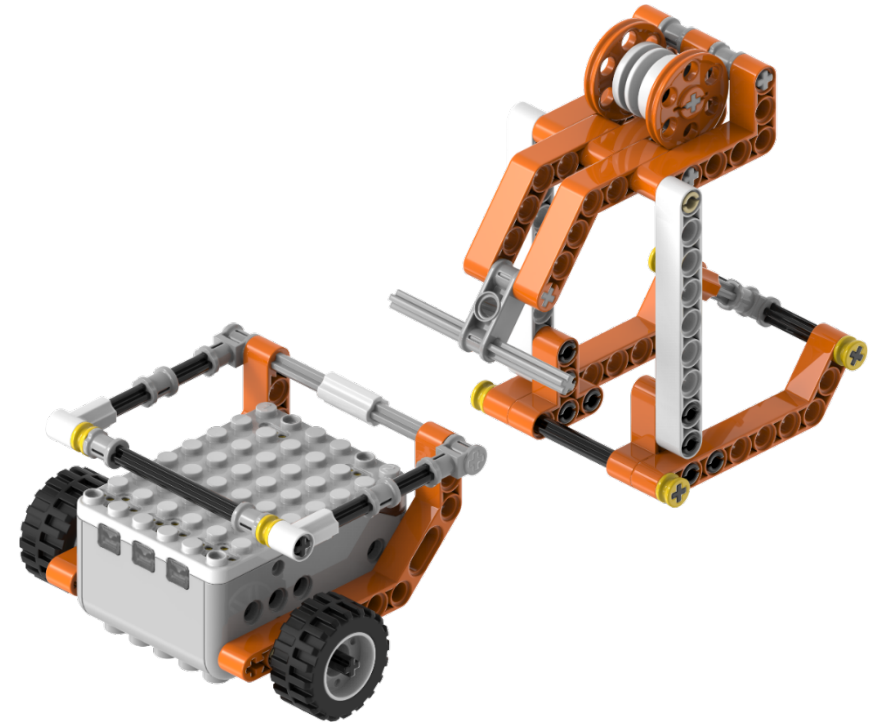
Running the program offline showed that the robot barely moved. After testing, we found that 500 iterations allow the robot to move a certain distance, so we will use 500 as a unit.



Summary

4. Complete the competition

```
when clicked
set my variable to 25
repeat my variable * 500
  Go straight
set double built-in motors to keep running by speed at 1# (-0)% and 2# (0)%
wait 0.2 seconds
Get
forever
  Go straight
```



SHARE WITH YOUR PARENTS

Share the knowledge about the Carrying Competition with your mom and dad when you get home!

