

# 1. Introduction to Cutebot

## 1.1. Introduction

---

ELECFREAKS Cutebot is a rear-drive smart car driven by dual high speed motors.

There are many on-board equipments on the Cutebot including ultrasonic sensor and distance sensor, two RGB LED headlights and clearance lamps on the bottom, two line-tracking probes, an active buzzer as the horn and so on! Let's drive your first smart car!

### Characteristics

---

- Rear-drive high speed motors featuring strong power.
- Tiny structure with an arc shape featuring crashproof and comfortable feel.
- Only batteries and ultrasonic sensor need to be assembled featuring easy installation.

## 1.2. Pictures

---



### 1.3. Parameters

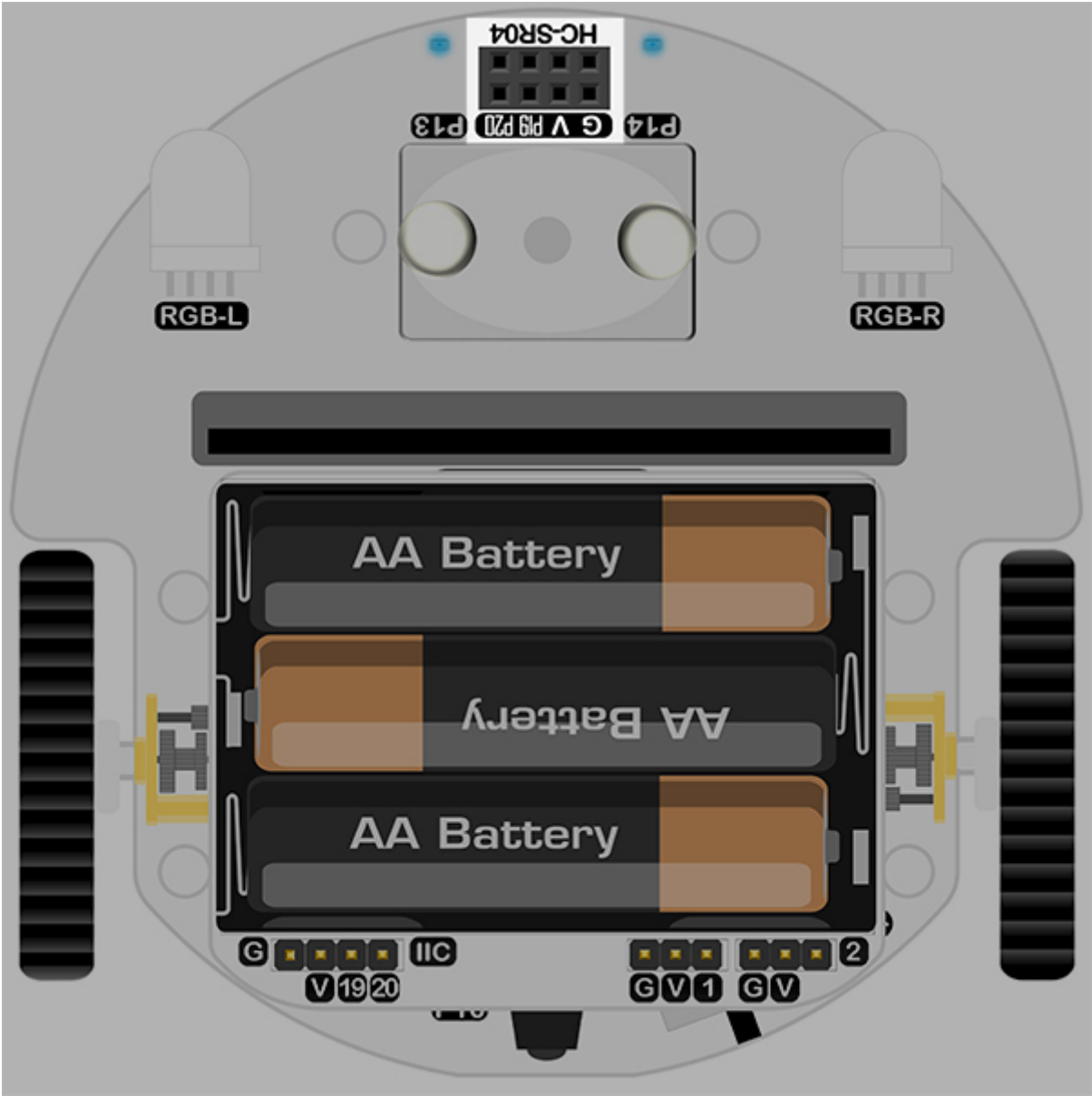
Items	Parameter
Working Voltage	3.5V – 5V
Dimension	85.68mm X 85.34mm X 38.10mm
Buzzer	Active buzzer connects to P0
Infrared Control	Connect to P16
RGB Headlights	2 x RGB
Rainbow LED	2 x Neopixel connect to P15
Connection	IIC Port(P19,P20)、Ultrasonic Port、P1、P2(GVS lead-out)

Motor Type	GA12-N20 DC micro gear deceleration motor(300 RPM)
------------	--

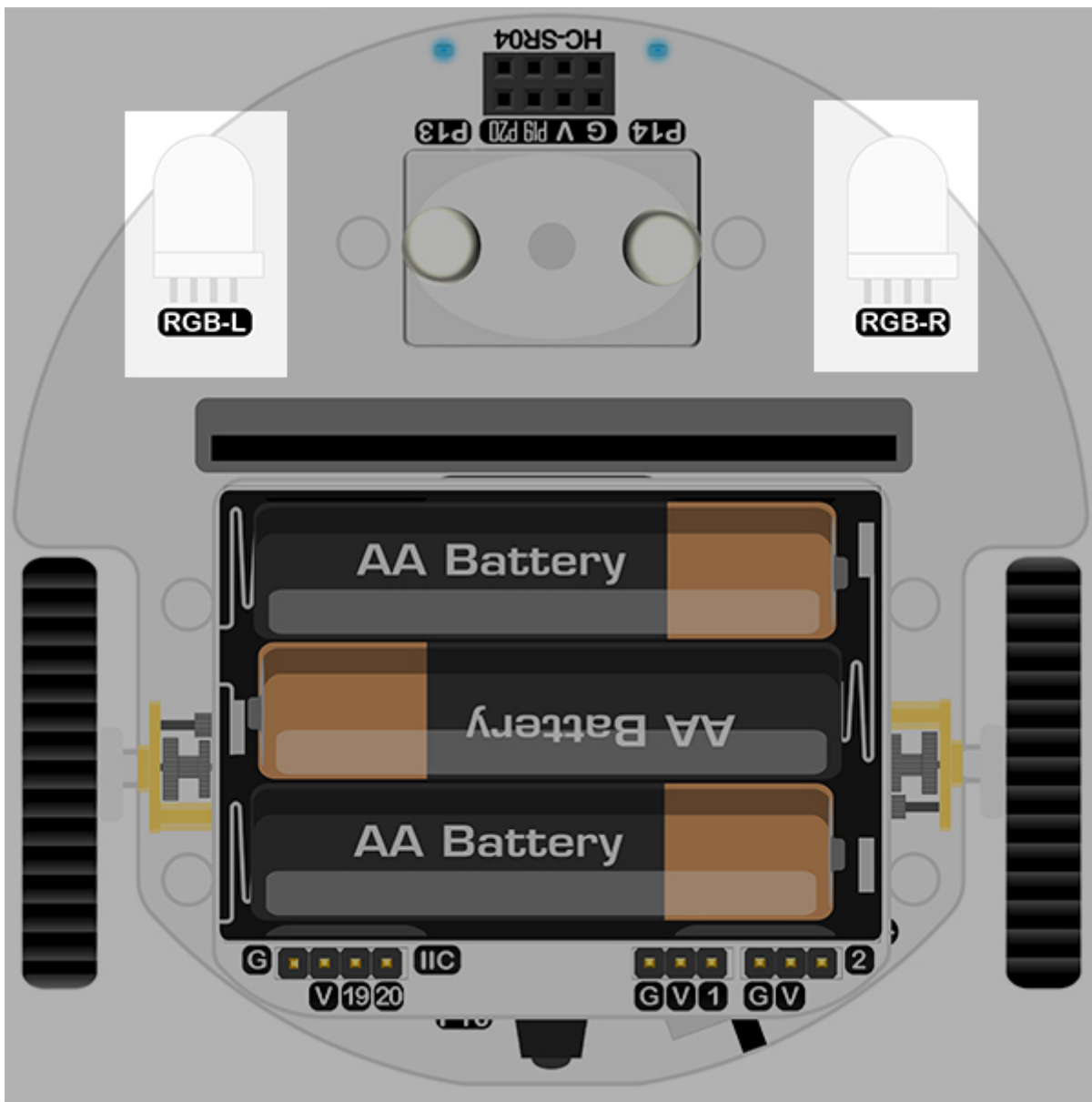
Ultrasonic Sensor Type Items	HC-SR04(2cm-400cm Contactless distance detection, precision $\pm 1.5\text{mm}$ ) Parameter
---------------------------------	---

## 1.4. Main Modules Introduction

Ultrasonic connection and micro:bit IIC port are placed in the front part of the Cutebot.

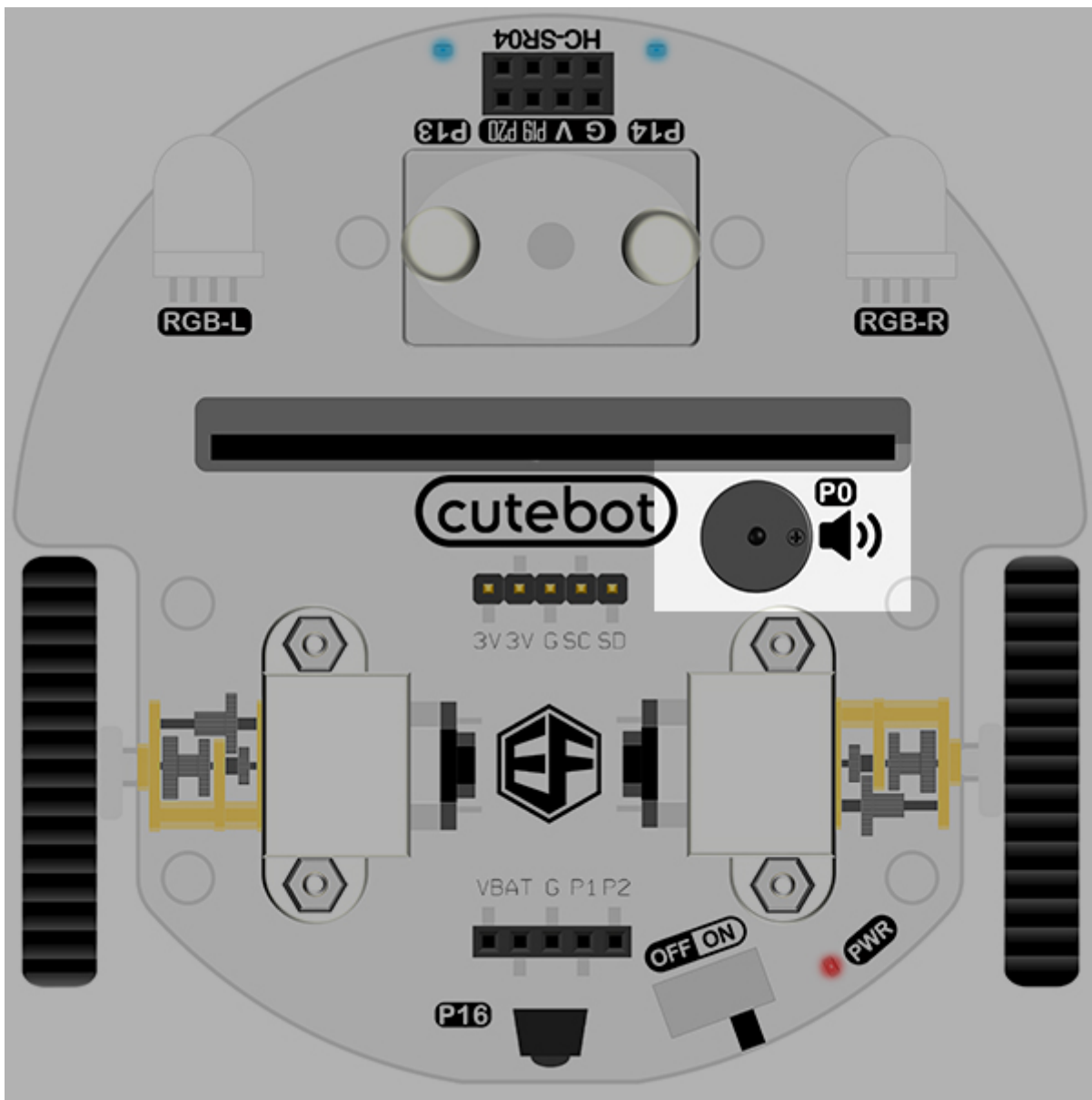


Two full color RGB lights controlled by the expansion board are placed on both side of the front part.

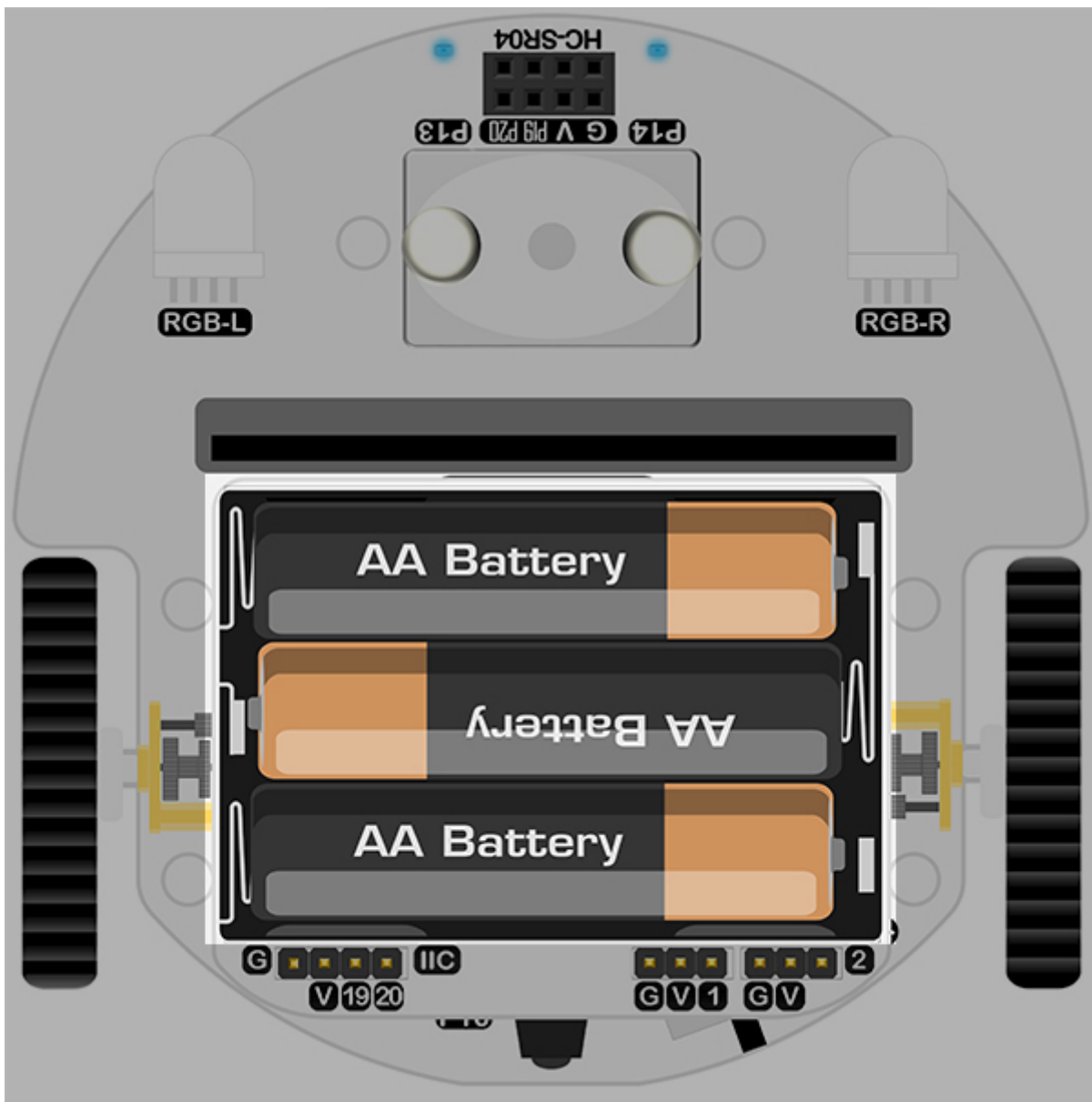


The on-board buzzer connecting to P0 port on the micro:bit can be alarmed by the bricks in [Music](#)

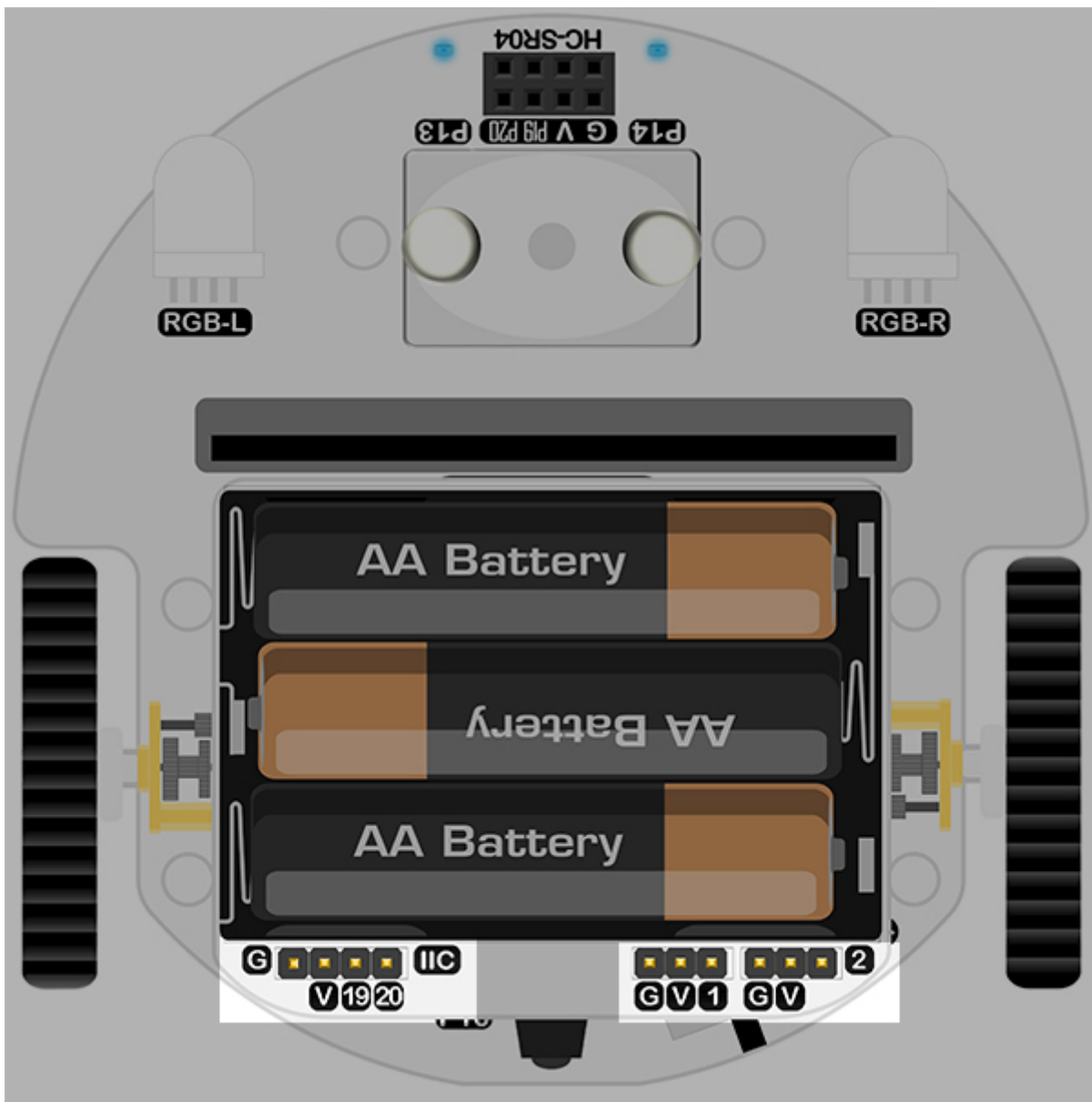




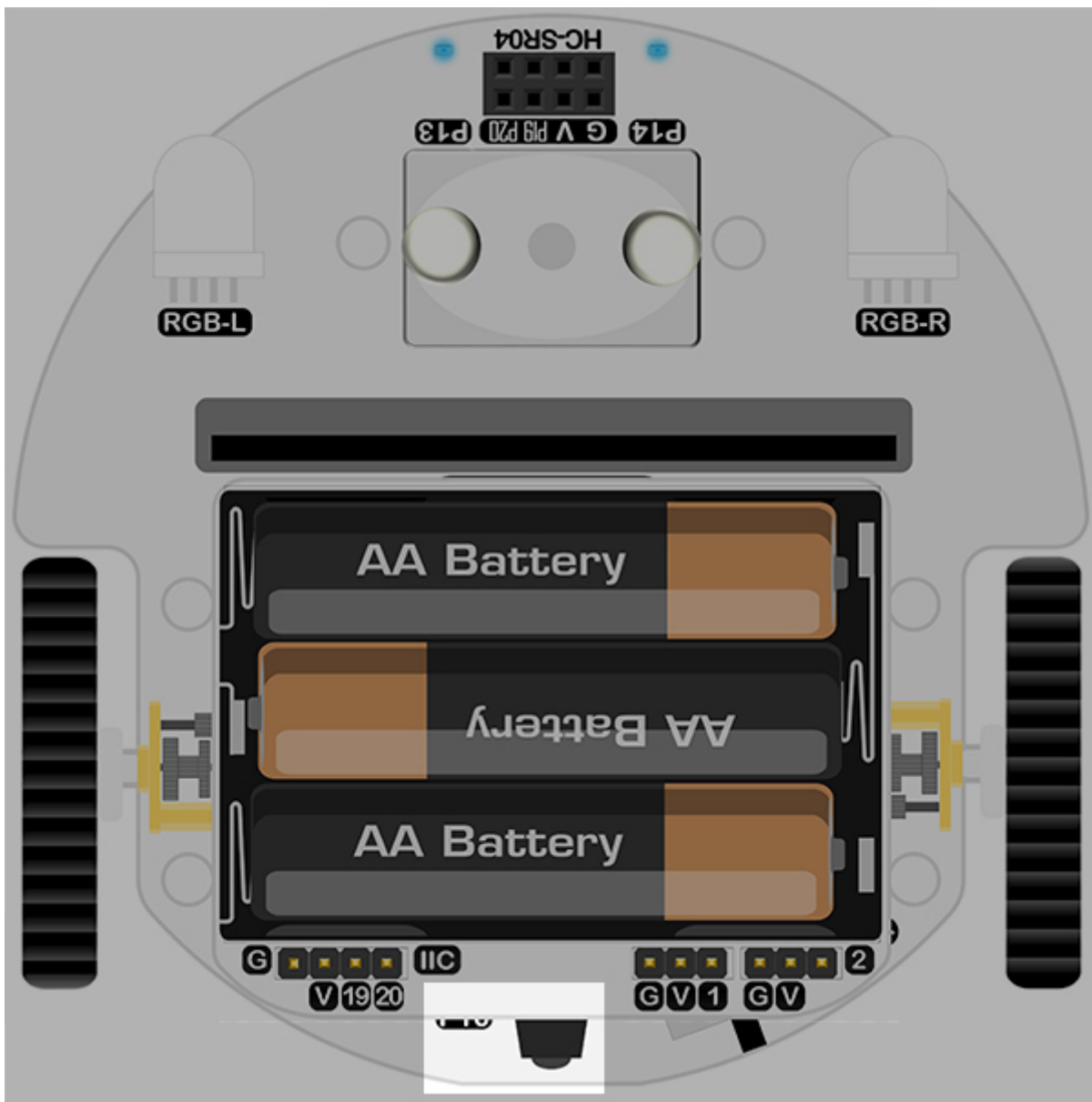
An expansion board for 3x AA batteries is placed in the right above part of the Cutebot.



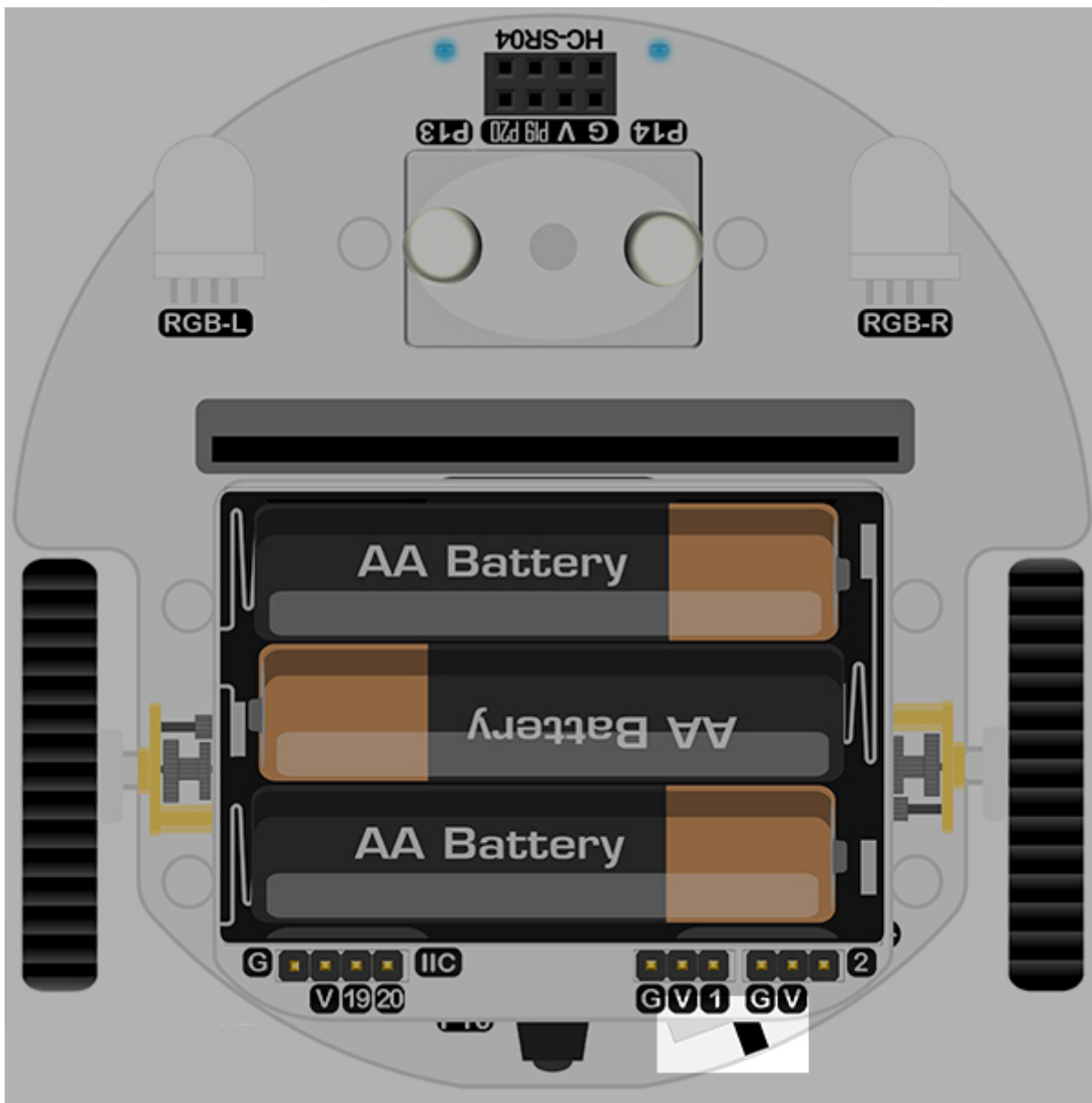
The IIC port and P1,P2 IO connections are equipped in the battery expansion board.



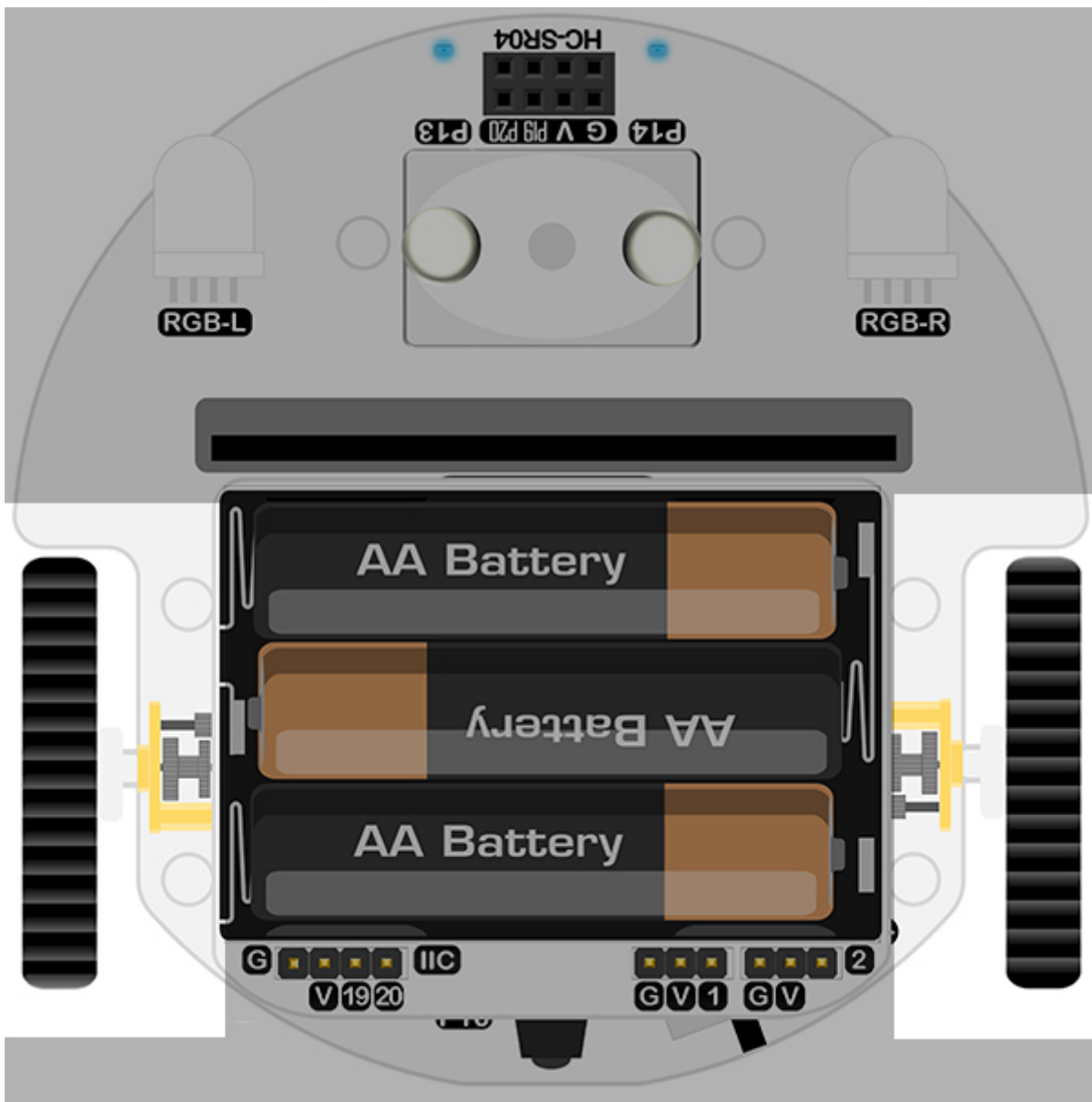
The infrared probes connecting to P16 port of the micro:bit are placed on the tail part of the Cutebot.



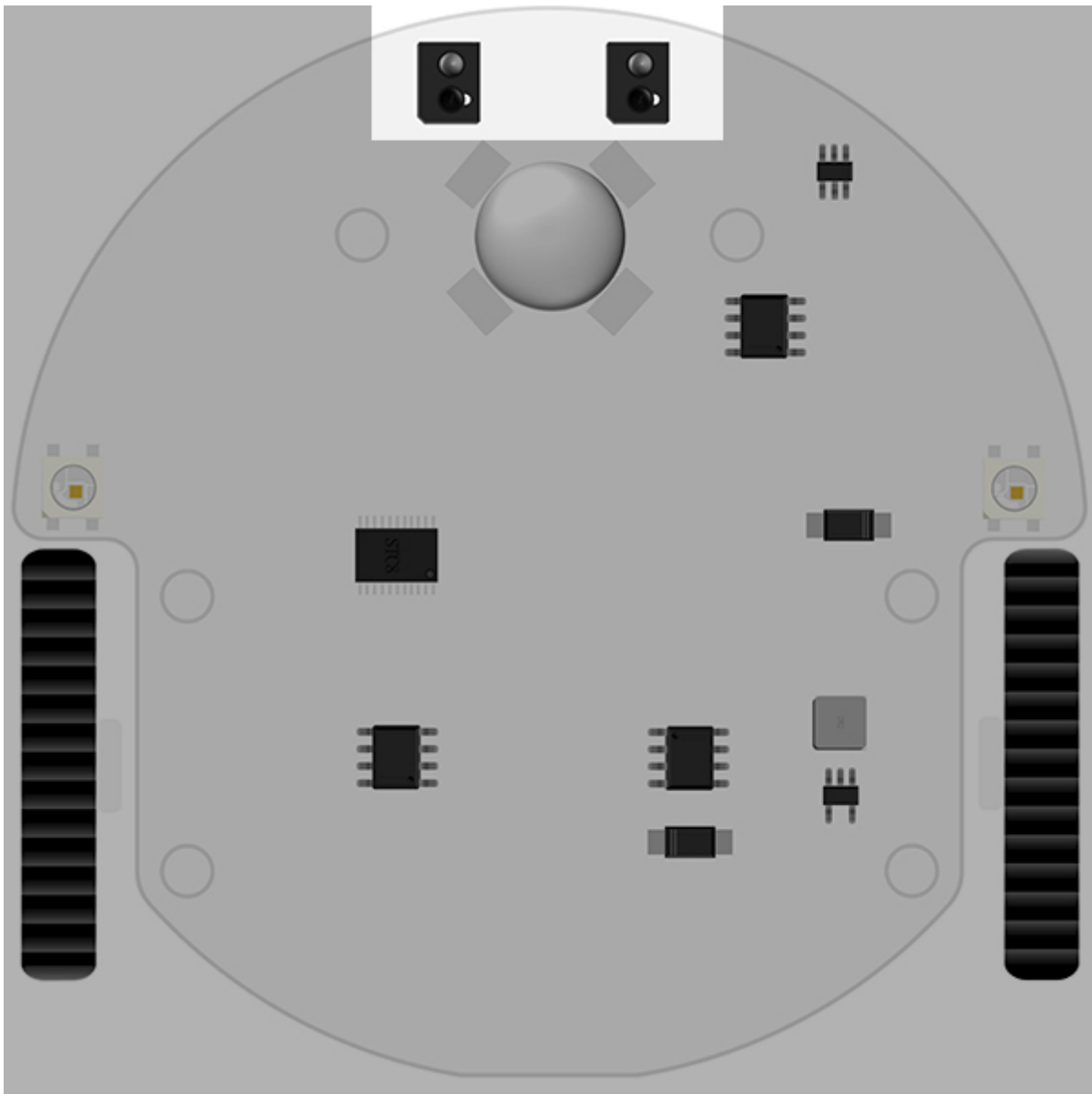
The master switch are placed besides the infrared probes and with on/off status showing by the LED.



The two wheels on both side are driven by DC micro gear deceleration motors(300 RPM).

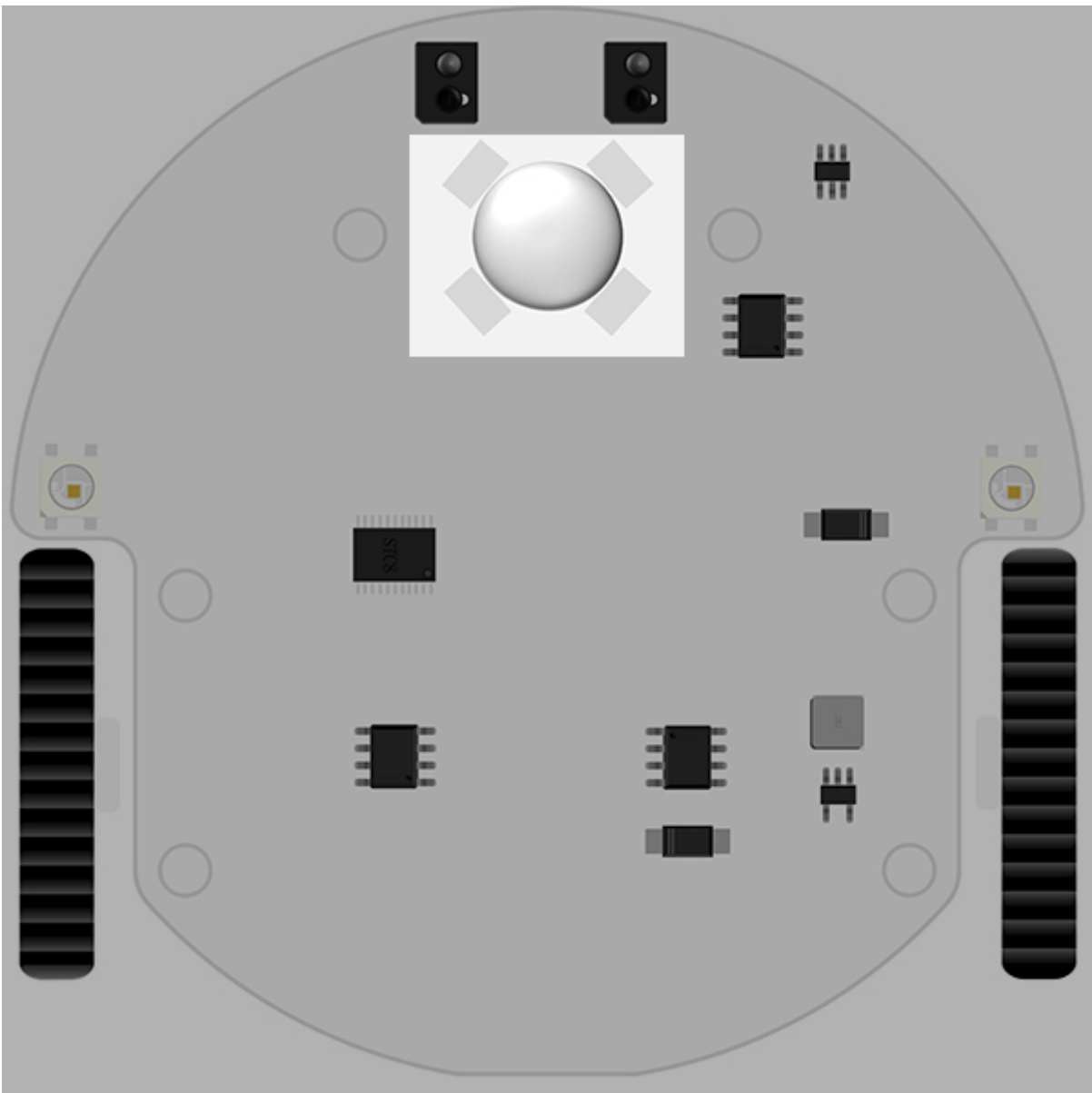


The two line-tracking probes connecting to P13&P14 on the micro:bit are used to detect the black line and its edges.



---

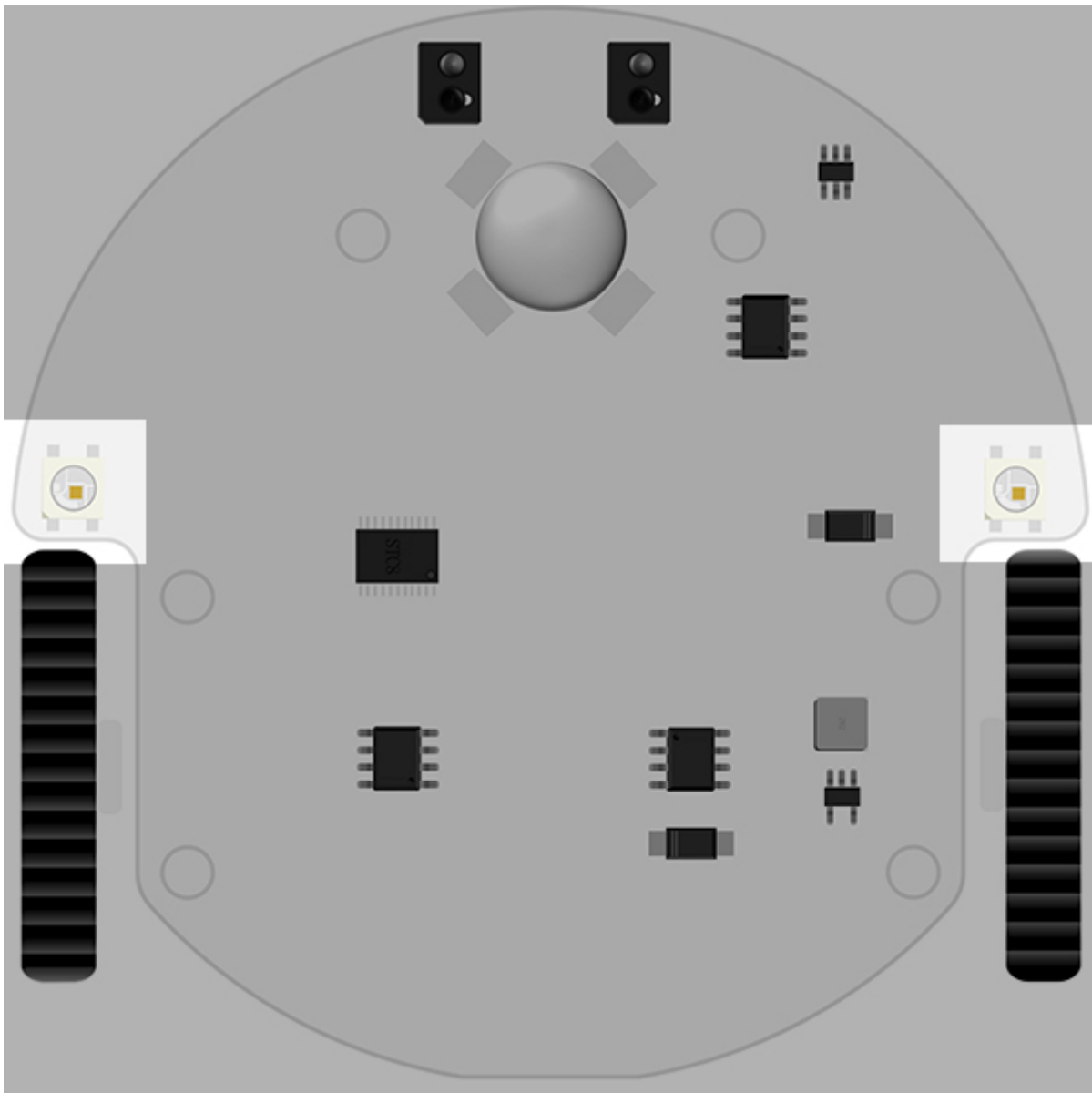
A universal wheel is placed in the front bottom of the Cutebot, an all-direction drive can be realized by the different speed of the left and right wheels.



---

The two full color Rainbow LEDs programmed by [Neopixel](#) connecting to P15 on the micro:bit are placed on both bottom side of the Cutebot and can be used as the clearance lamps or others.





## 1.5. Components list

---

- 1 x Cutebot car
- 1 x Battery Holder
- 1 x HC-SR04 Ultrasonic Sensor
- 1 x Line-tracking Map
- 1 x Brochure

## 1.6. Files

---

## 1.7. FAQ

---

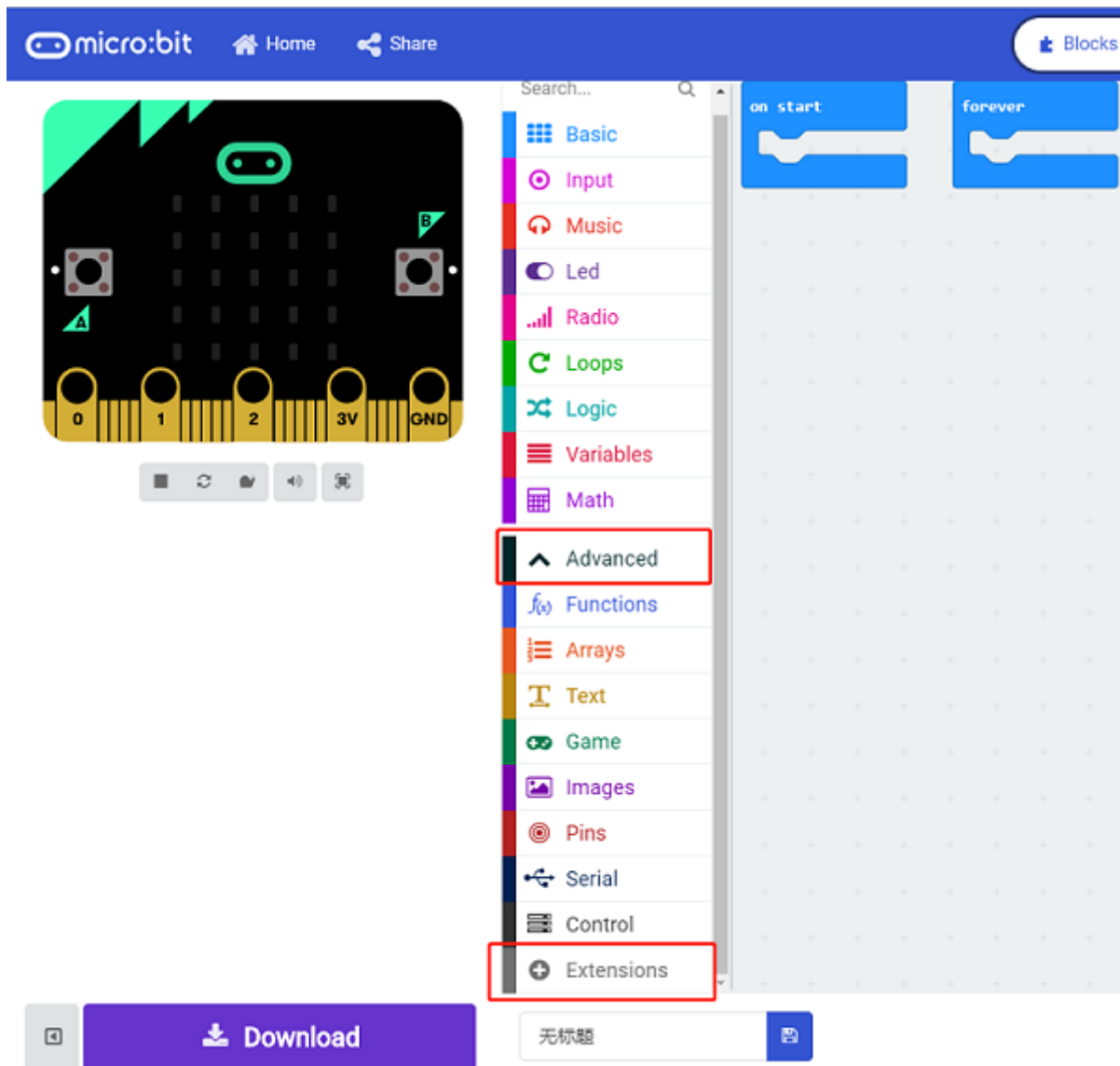
## 2. Add Package for Cutebot

### 2.1. Purpose

- A new package is required if you want to use the expansion bricks for Cutebot.
- Steps for adding package are list below.

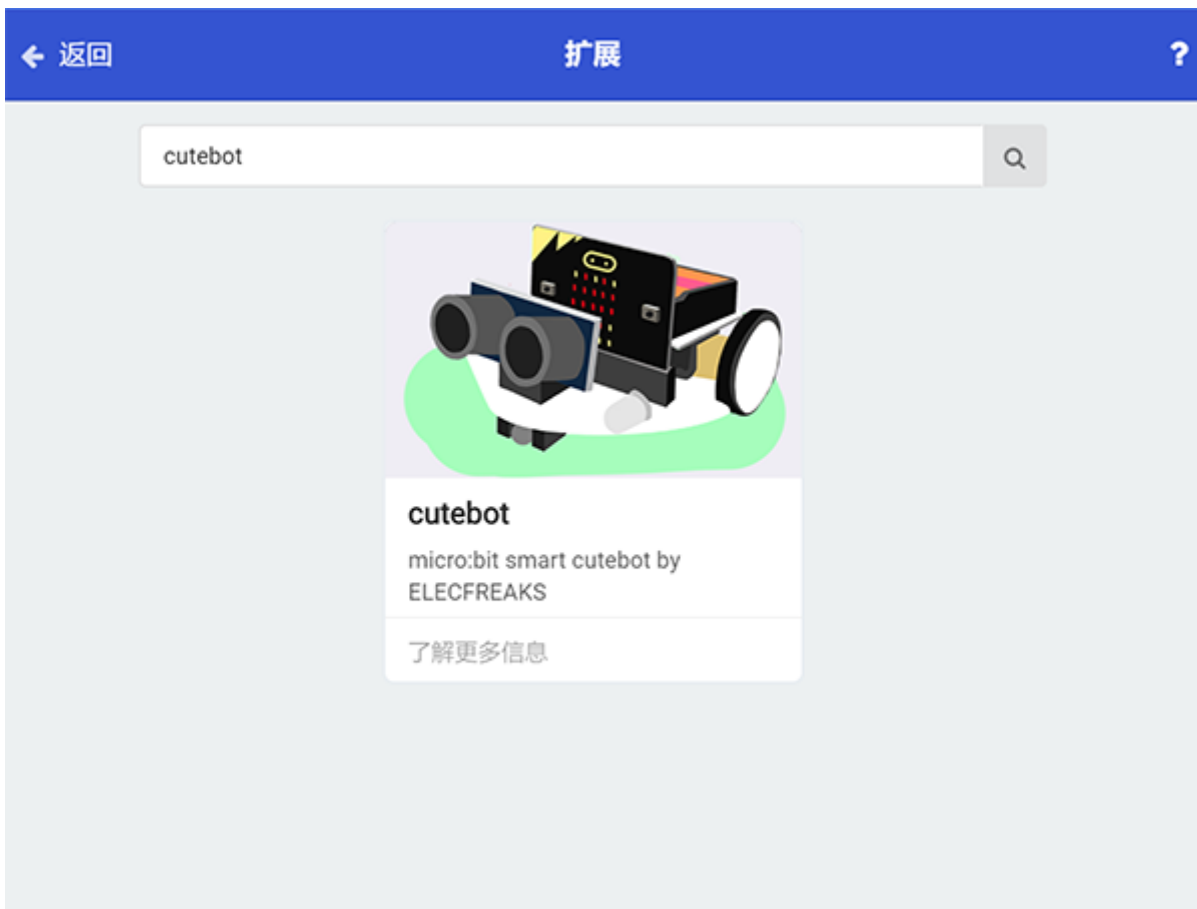
#### Step 1

- Click “Extensions” in the “Advanced” drawer to see the adding bricks menu.



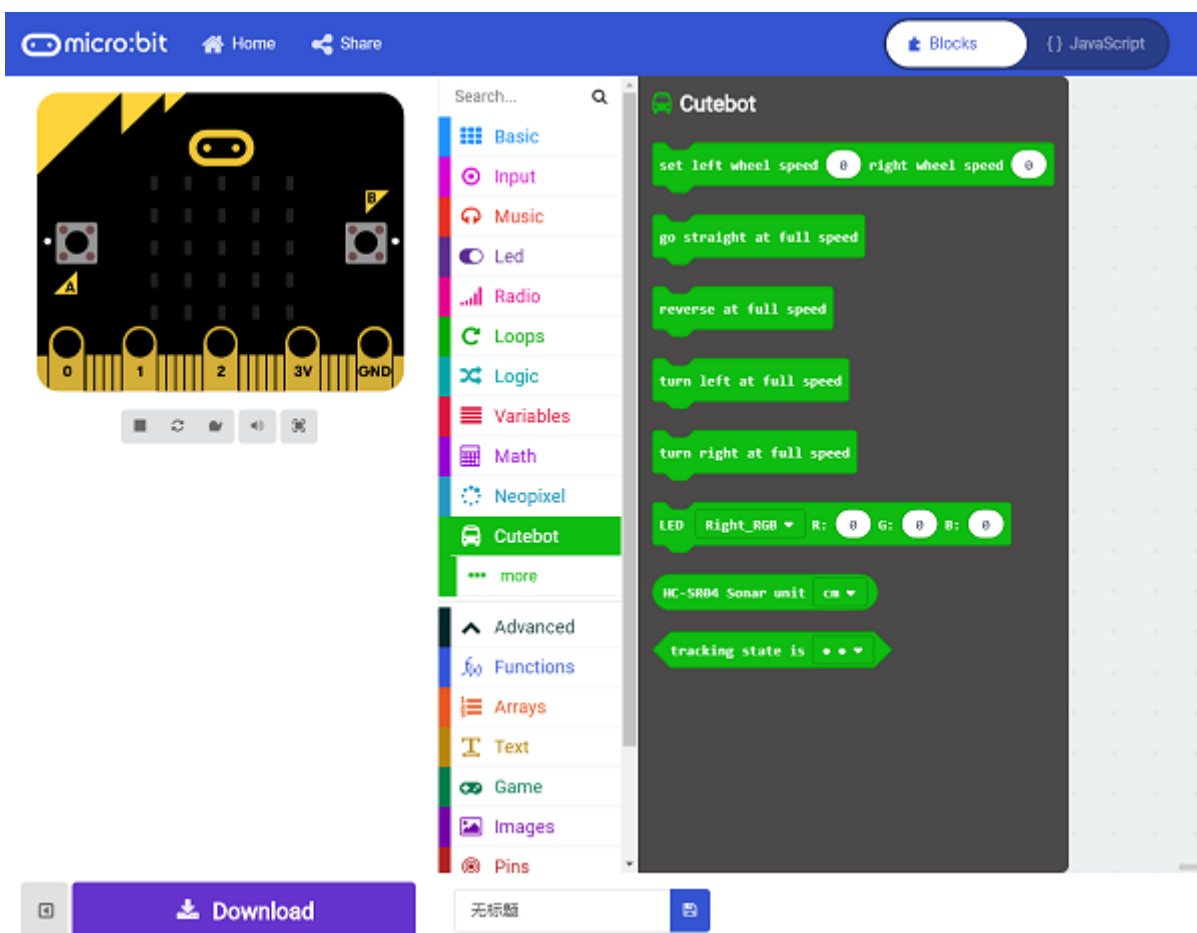
#### Step 2

- Search “cutebot” in the box and click it to add the package.



### Step 3

- Completed



## 2.2. Bricks Introduction

---



set left wheel speed 0 right wheel speed 0

- This brick helps to adjust the speed of both wheels.
- 



go straight at full speed

- This brick helps the car to move at its full speed.
-



**reverse at full speed**

- This brick helps the car to reverse at its full speed.
- 



**turn left at full speed**

- This brick helps the car to turn left at its full speed.
-

turn right at full speed

- This brick helps the car to turn right at its full speed.
- 

tracking state is



- This brick helps to detect the line-tracking status for the line-tracking modules.
- 

HC-SR04 Sonar unit

cm

- This brick helps to detect the distance for the ultrasonic sensor.



- This brick helps to control the color of the RGB lights on both sides.

## 2.3. FAQ

---

Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## 2.4. Relevant Files

---

## 3. Case 01: Move Forward or Reverse at the Full Speed

### 3.1. Purpose

---

- Learn the basic functions of Cutebot-move forward or reverse.

### 3.2. Materials

---

- 1 x Cutebot Kit

### 3.3. Software Platform

---

MicroSoft makecode

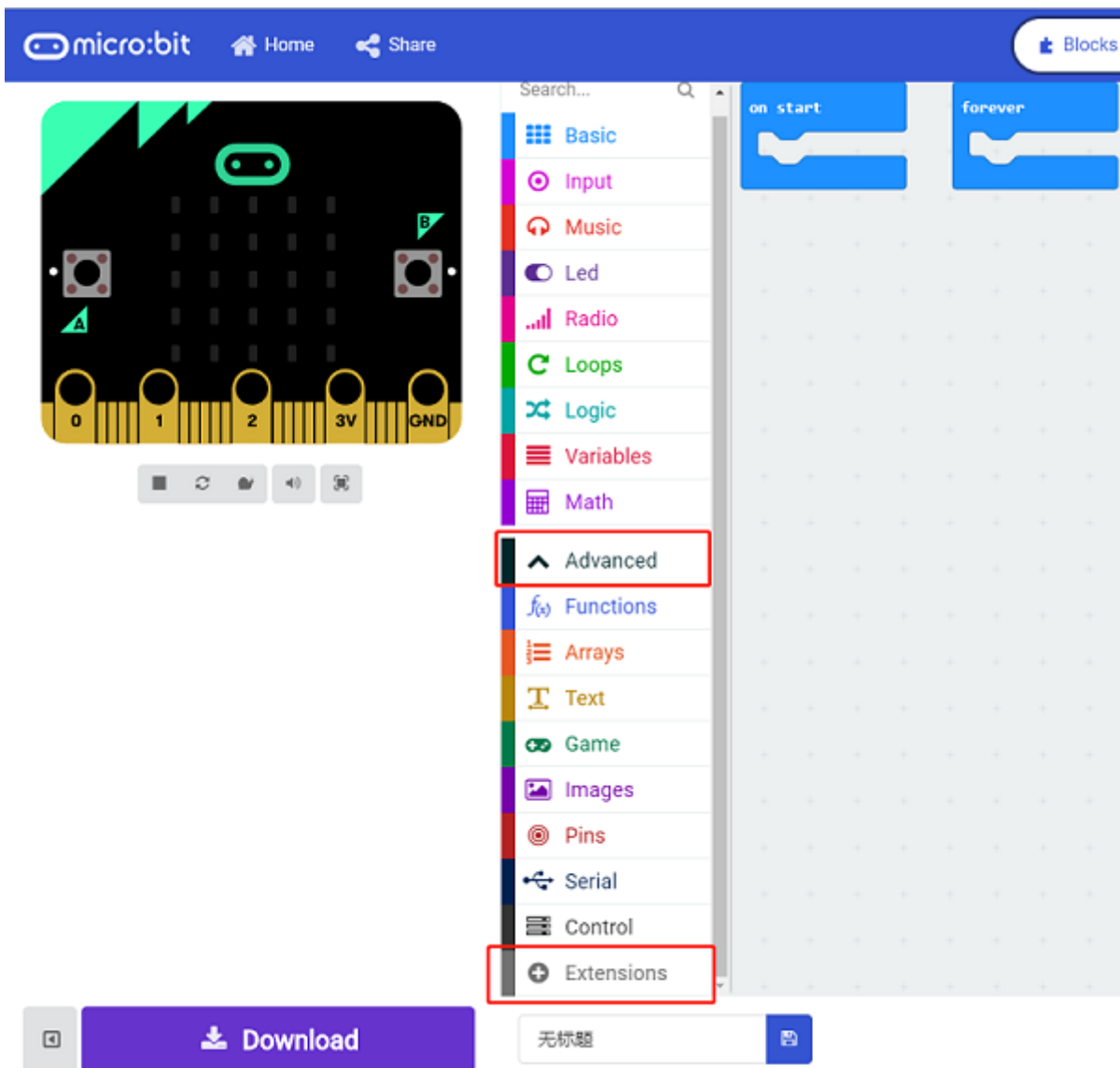
### 3.4. Programming

---

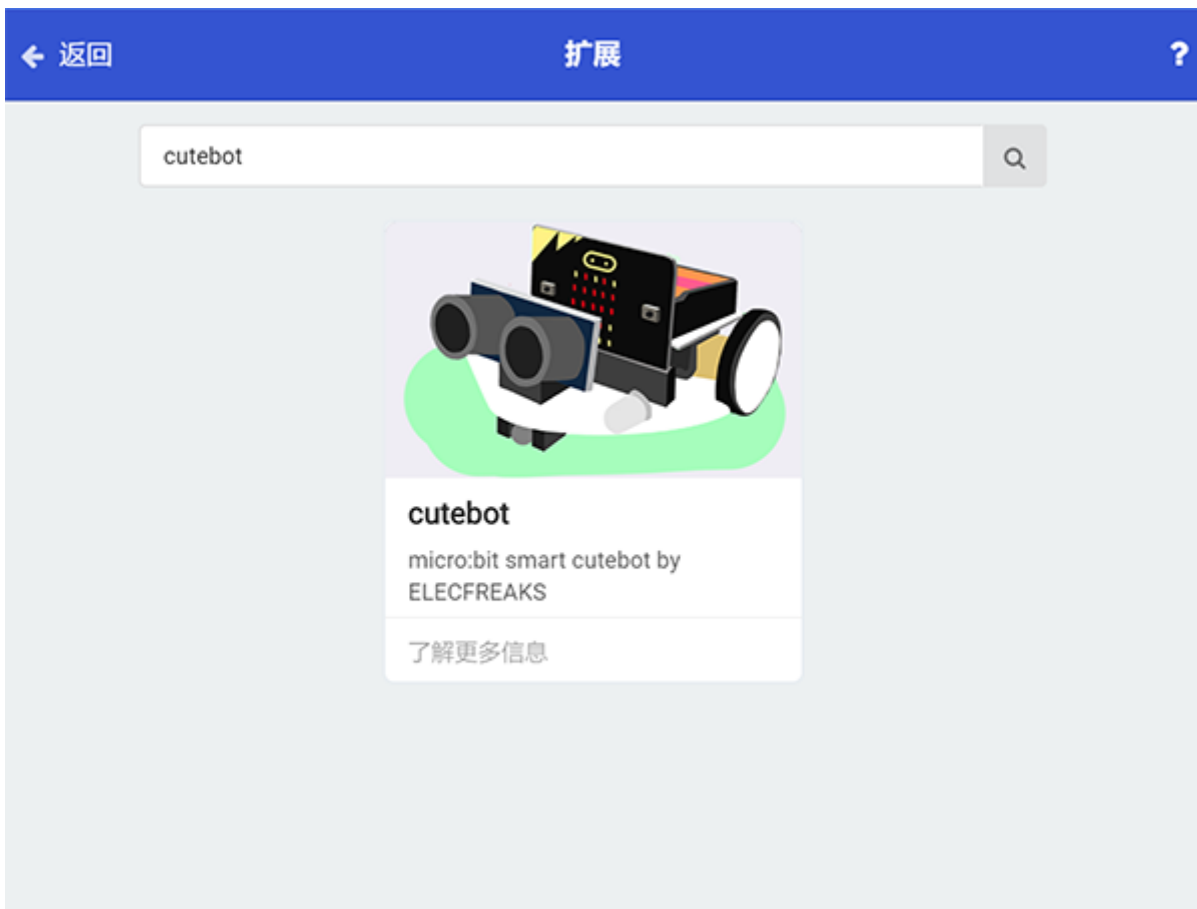
#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.





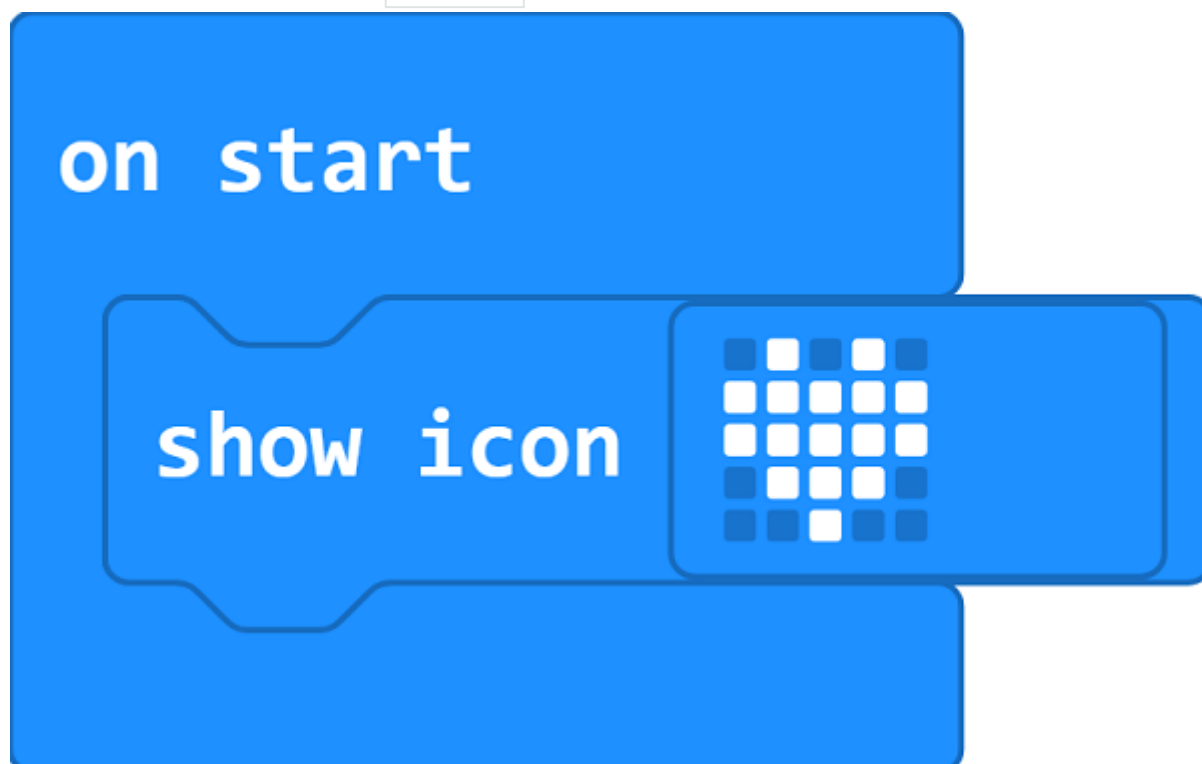
- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Choose “show icon” in the `On start` bricks.



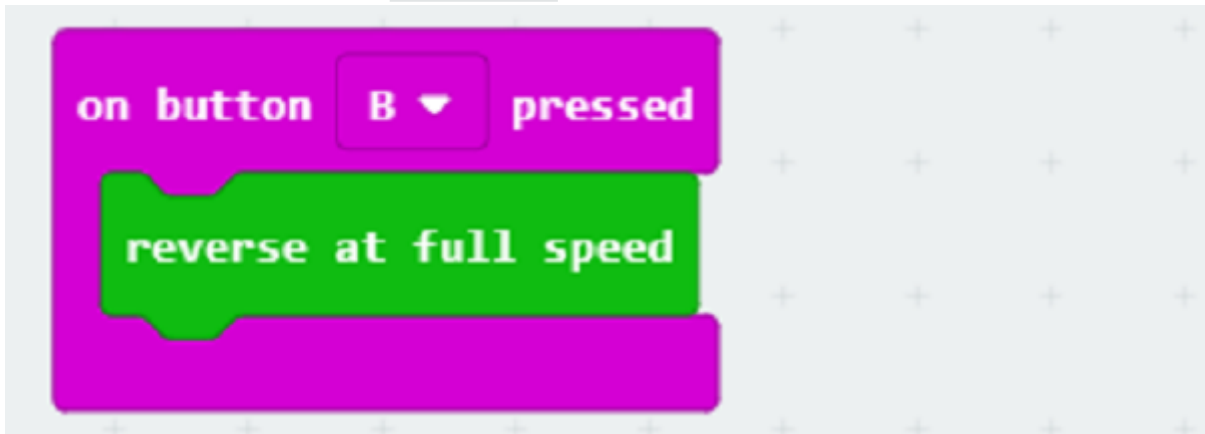
## Step 3

- Drag `go straight at full speed` brick into `on button A pressed` brick.



## Step 4

- Drag `reverse at full speed` brick into `on button B pressed` brick.



## Programming

Links: [https://makecode.microbit.org/\\_LXJCwmAsf4dV](https://makecode.microbit.org/_LXJCwmAsf4dV)

You can also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼    📄 Edit

---

## **3.5. Result**

---

- After button A being pressed, the car moves forward at its full speed.
- After button B being pressed, the car reverses at its full speed.

## **3.6. Exploration**

---

How to program the car to stop moving after pressing button A ?

## **3.7. FAQ**

---

## **3.8. Relevant Files**

---

## 4. Case 02: Speed Up Gradually

### 4.1. Purpose

---

- In case 01, we can find the Cutebot moves too fast to go steadily(the universal wheel goes off the ground) at the beginning.
- We will learn to gradually speed up the car for a steady move at the beginning in this case.

### 4.2. Materials

---

- 1 x Cutebot Kit

### 4.3. Software Platform

---

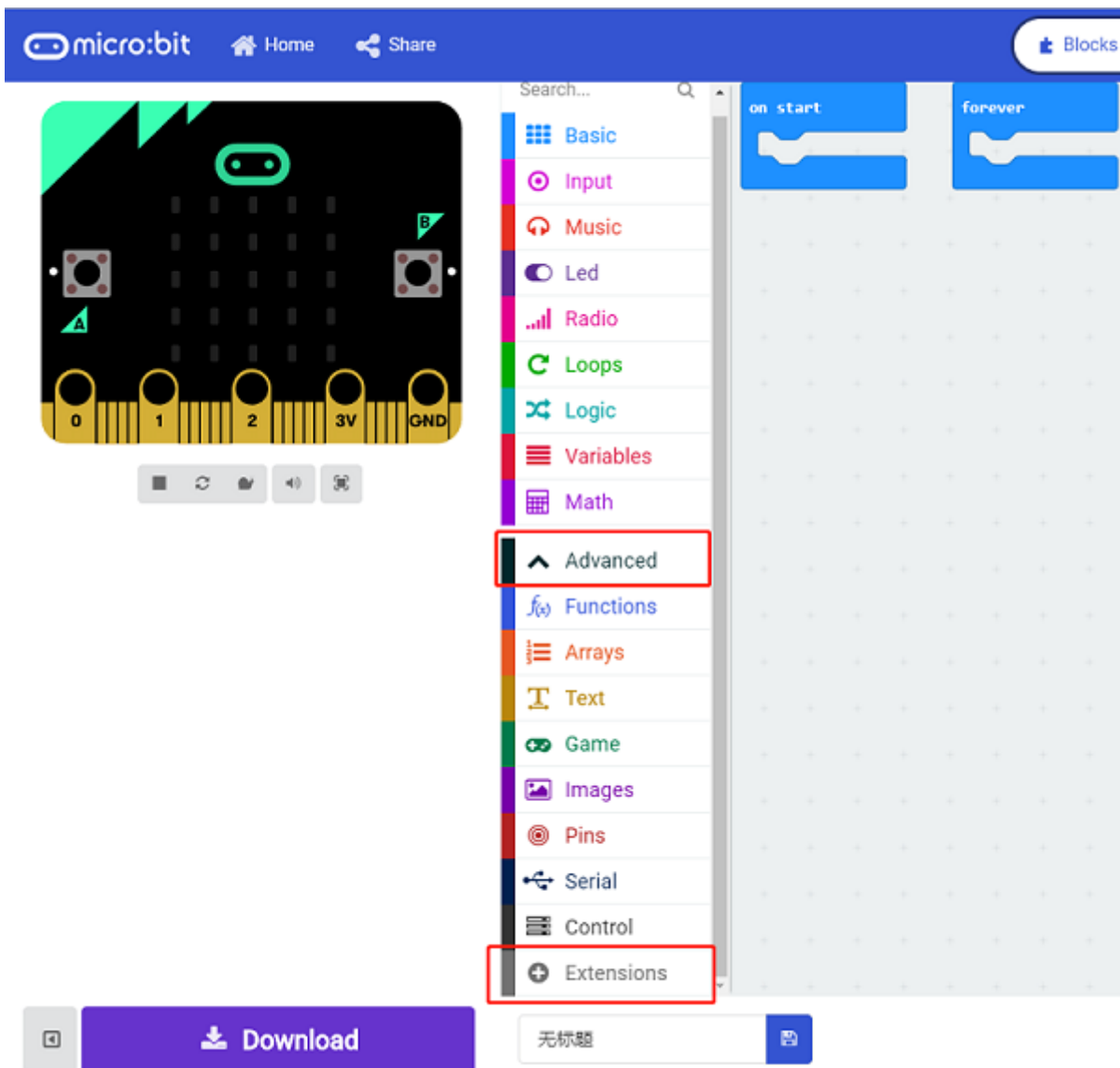
MicroSoft makecode

### 4.4. Programming

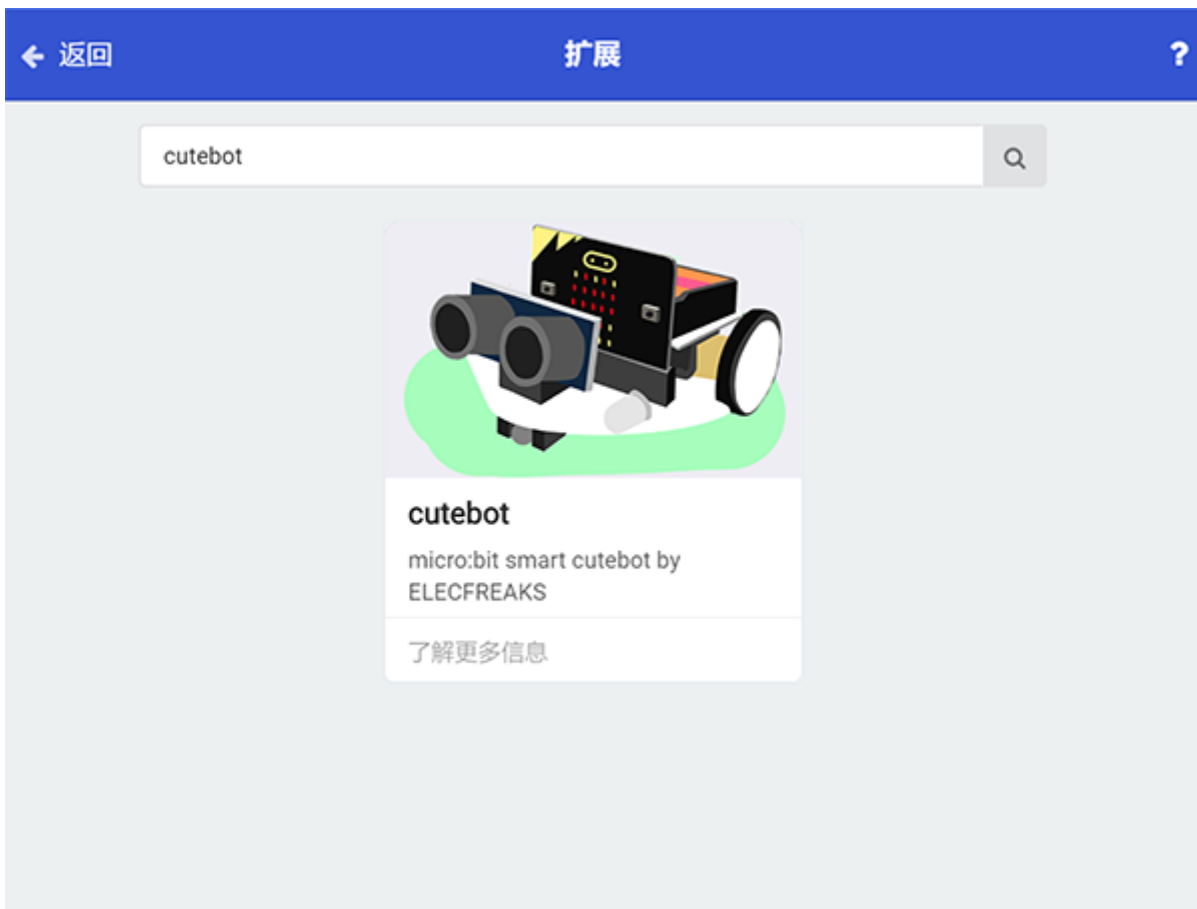
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



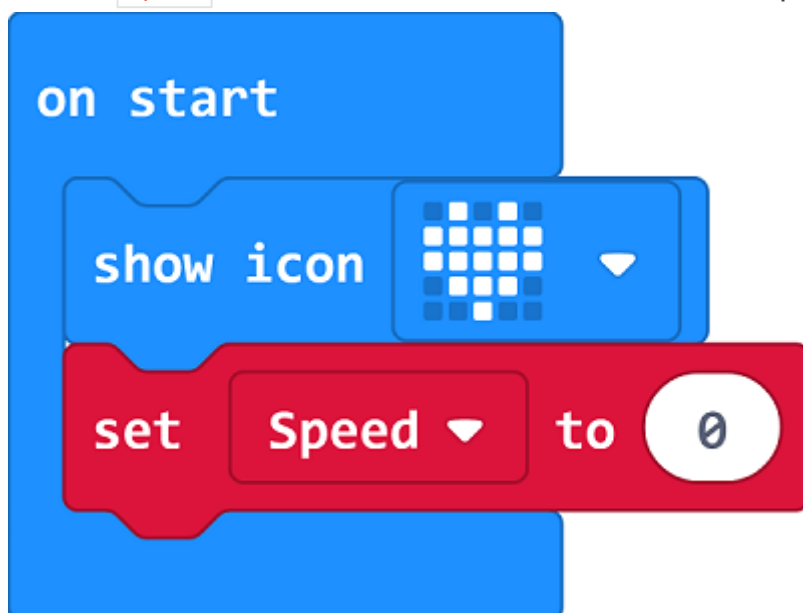
- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Choose “show icon” in the `On start` brick.
- Set the `speed` variable to 0 which means the on start speed is 0.



## Step 3

- Drag setting speed bricks for left and right wheel and set the value as `speed` in “forever” brick, then add one to `speed`.

- If `speed` is `100` which is the maximum speed, set `speed` to 0 and restart it.

```
forever
  if Speed = 100 then
    set Speed to 0
    +
    set left wheel speed Speed right wheel speed Speed
    change Speed by 1
```

The image shows a Scratch script with the following blocks:

- A blue **forever** loop block.
- A teal **if** block with the condition `Speed = 100` and the word **then**.
- A red **set** block: `set Speed to 0`.
- A teal **+** block (comment block).
- A green **set** block: `set left wheel speed Speed right wheel speed Speed`.
- A red **change** block: `change Speed by 1`.

## Programming

Links: [https://makecode.microbit.org/\\_6X6aA3cKKMAAt](https://makecode.microbit.org/_6X6aA3cKKMAAt)

You can also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▾    ↗ Edit

---

## 4.5. Result

---



- The Cutebot speeds up gradually and the universal wheel will not go off the ground due to the high speed.

## **4.6. Exploration**

---

- How to program to make the car speed up gradually and then speed down gradually?

## **4.7. FAQ**

---

## **4.8. Relevant Files**

---

## 5. Case 03: Dance in Figure-of-eight

### 5.1. Purpose

---

- Make your Cutebot move in the figure-of-eight.
- Cutebot is a car with three wheels and the direction is adjusted by the different speed of the left and right wheels.

### 5.2. Materials

---

- 1 x Cutebot Kit

### 5.3. Software Platform

---

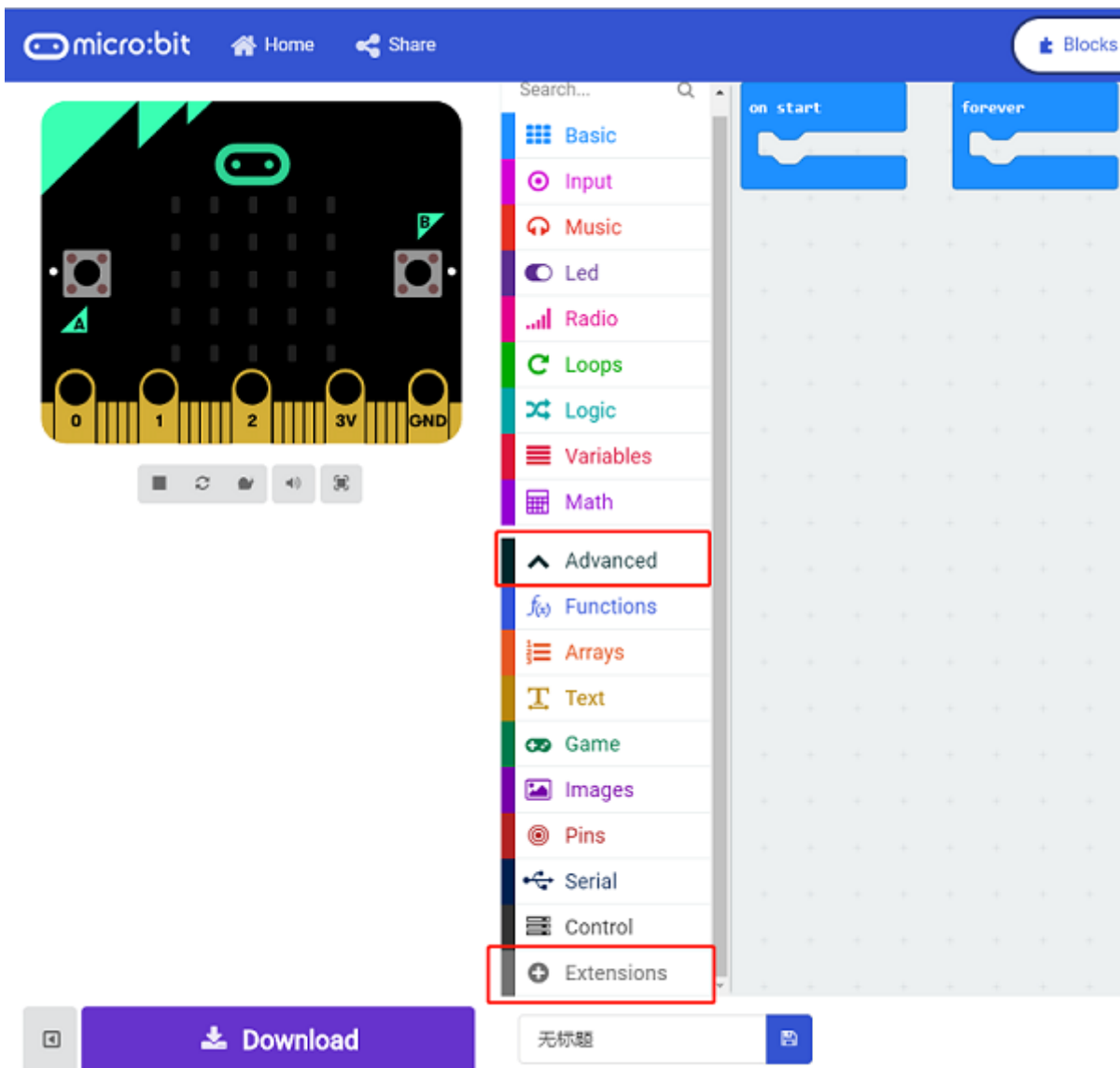
MicroSoft makecode

### 5.4. Programming

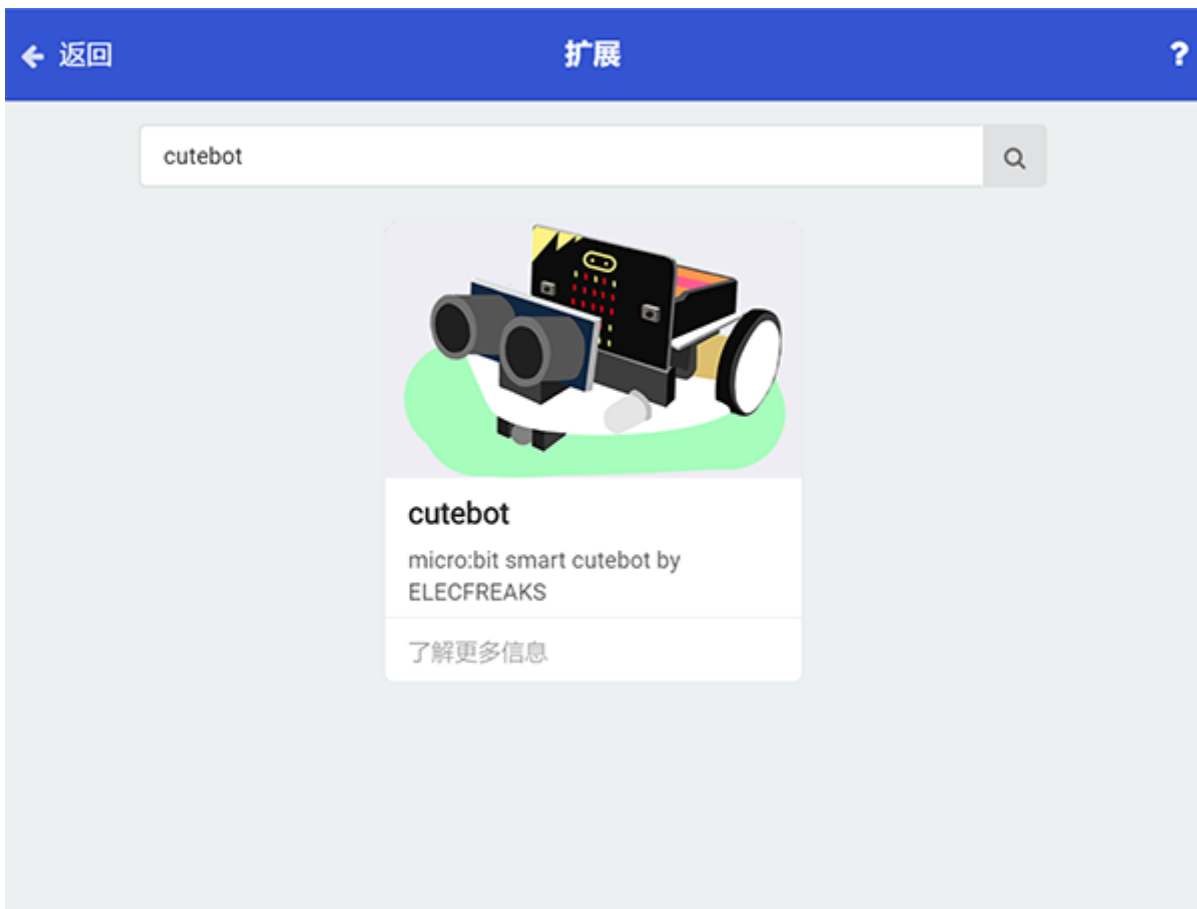
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



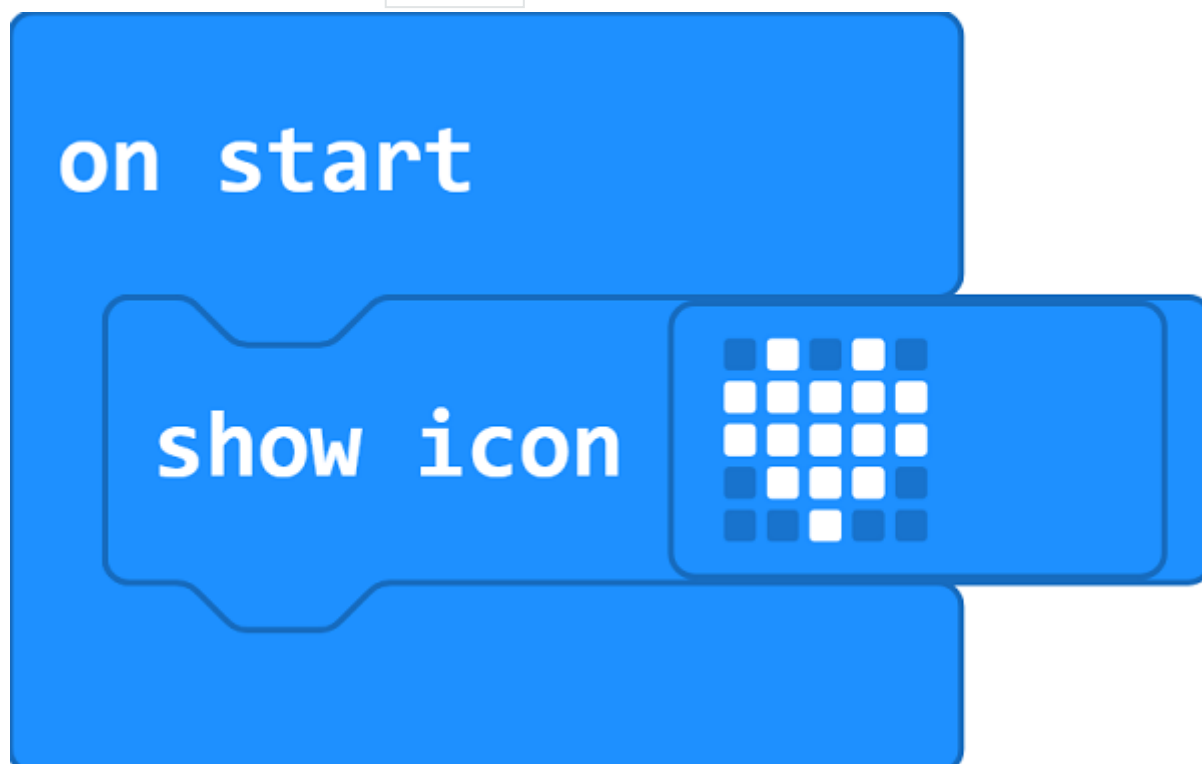
- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Choose “show icon” in the `On start` bricks.



## Step 3

- Drag “go straight at full speed” and “set left wheel speed, right wheel speed” bricks into the `Forever` brick in turns.
- Divide the “figure-of eight” track into six parts: move forward for 200ms at the beginning, set the speed of the left wheel is faster than the right and set to move for 1000ms after, then go straight for 200ms. Right now you have completed the half part of “figure-of-eight”.
- Complete the second half part of “figure-of-eight” in a similar way.

```
forever
  go straight at full speed
  pause (ms) 200
  set left wheel speed 100 right wheel speed 40
  pause (ms) 1000
  go straight at full speed
  pause (ms) 200
  go straight at full speed
  pause (ms) 200
  set left wheel speed 40 right wheel speed 100
  pause (ms) 1000
  go straight at full speed
  pause (ms) 200
```

Links: [https://makecode.microbit.org/\\_EPpWzRUqwAHA](https://makecode.microbit.org/_EPpWzRUqwAHA)

You can also download it directly below:



---

## 5.5. Result

---

- The Cutebot moves in the “figure-of-eight”.

## 5.6. Exploration

---

- How to program if we want to make the Cutebot move in a square shape?

## 5.7. FAQ

---

## 5.8. Relevant Files

---

## 6. Case 04: Run at Random

### 6.1. Purpose

---

- Make your Cutebot move(move forward, reverse or change direction) as if in “his” mind.

### 6.2. Materials

---

- 1 x CutebotKit

### 6.3. Software Platform

---

MicroSoft makecode

### 6.4. Programming

---

- 

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.

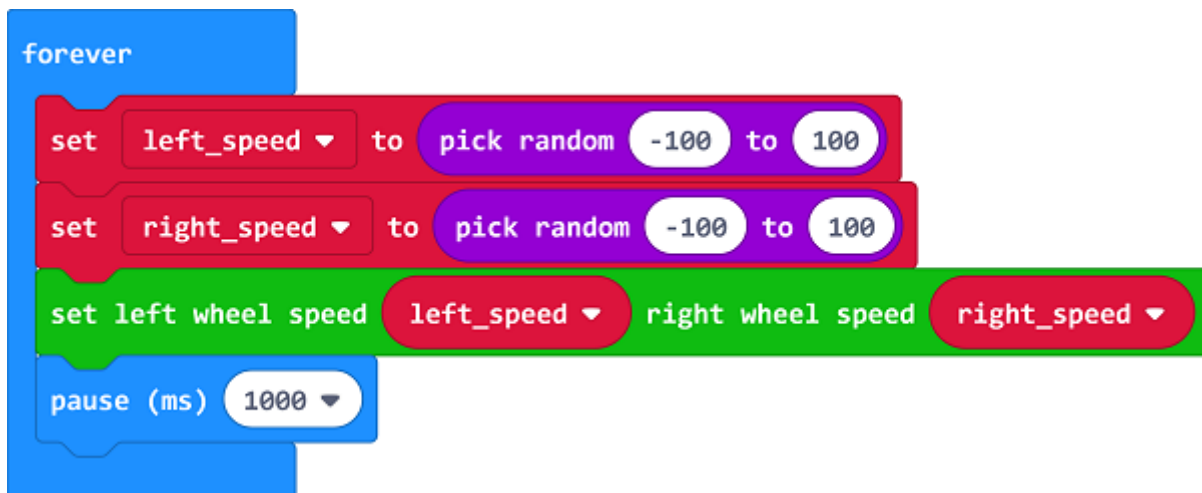
#### Step 2

- Choose “show icon” in the `On start` bricks.



### Step 3

- Drag `left_speed` and `right_speed` bricks into `forever` brick to set a speed at random from `-100` to `100` of the two wheels.
- Assign the two variables of the speed to the left and right wheels.



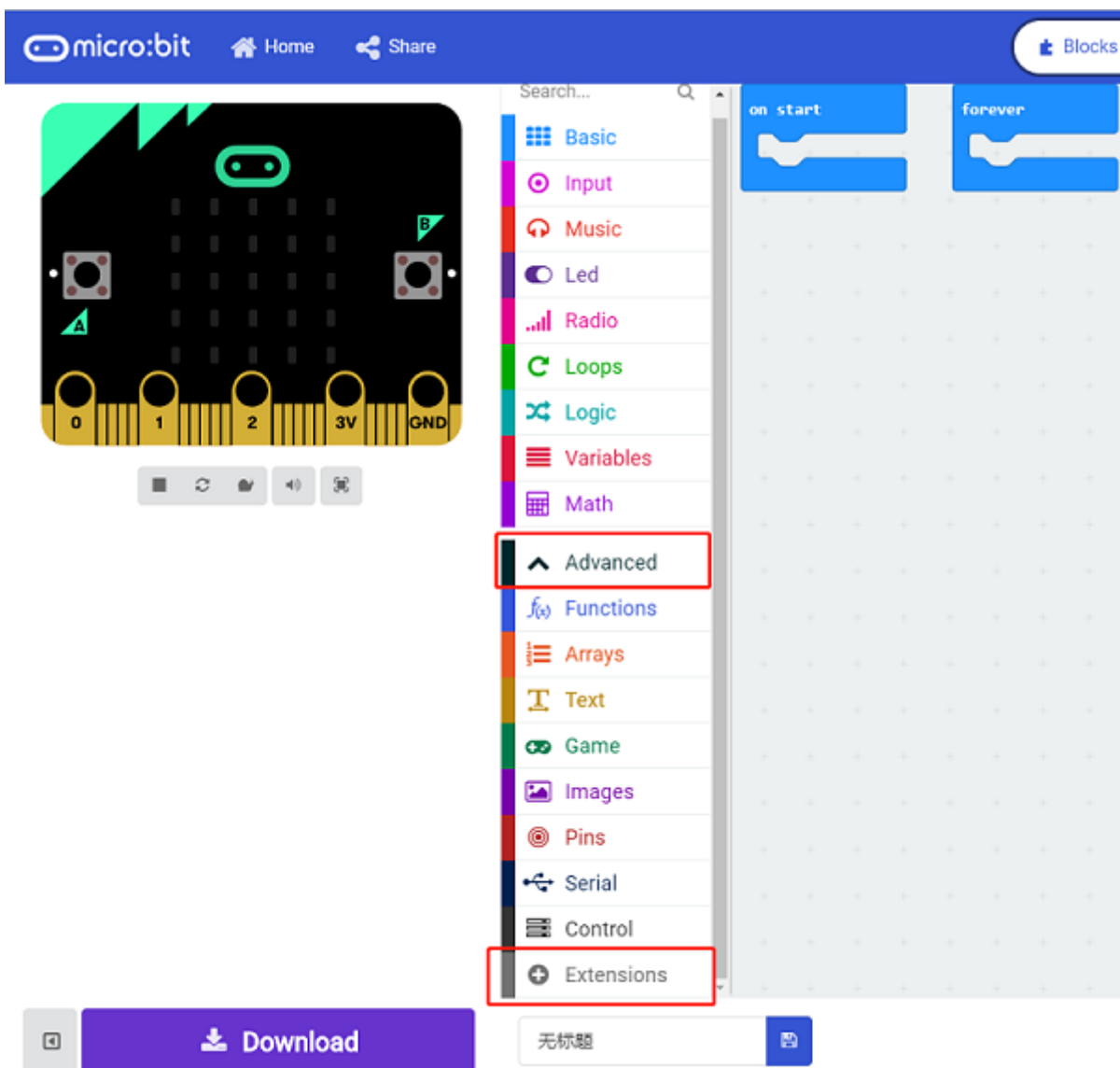
### Programming

Links: [https://makecode.microbit.org/\\_UFETasLycR3g](https://makecode.microbit.org/_UFETasLycR3g)

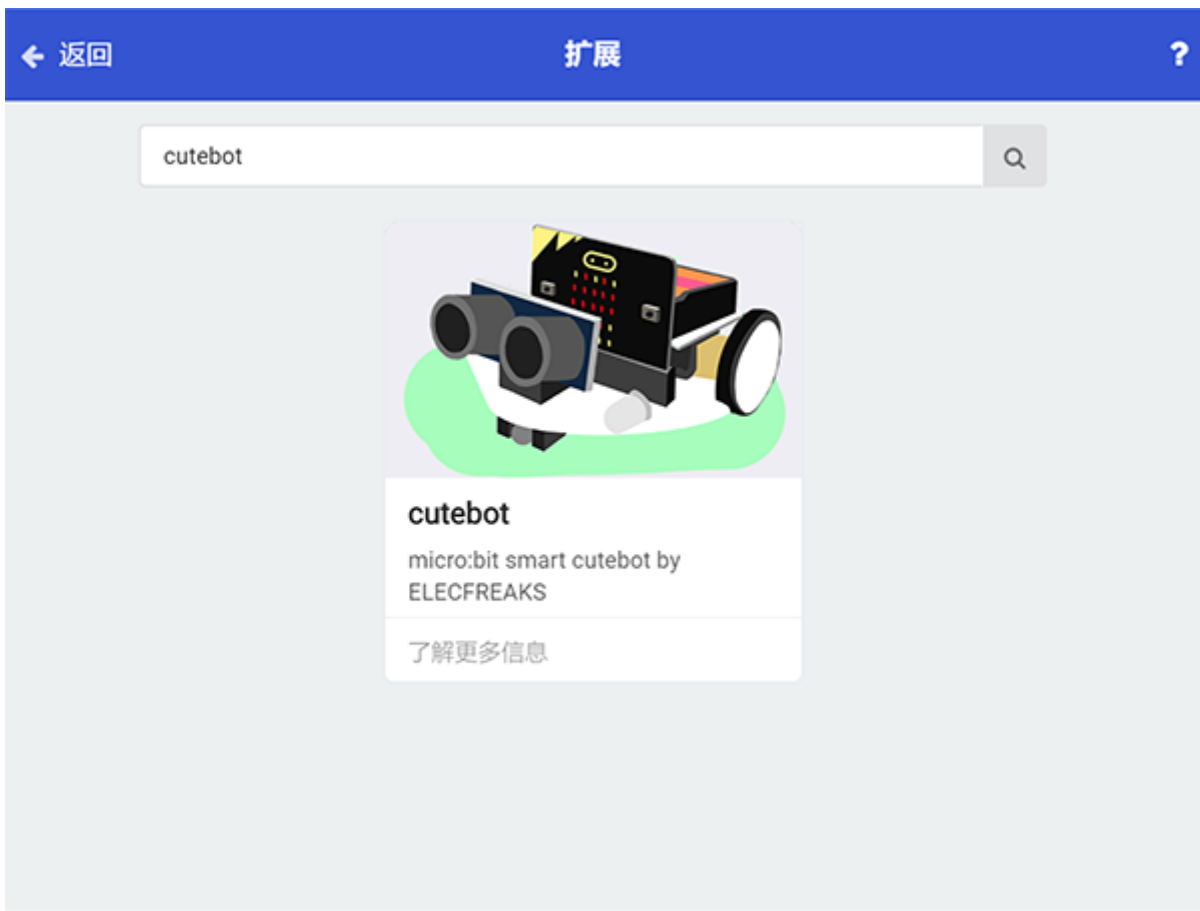
You can also download it directly below:







- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## 6.5. Result

---

- 小车随机前进，后退或者转向。 The Cutebot moves forward, reverses or changes its direction at random.

## 6.6. Exploration

---

## 6.7. FAQ

---

## 6.8. Relevant Files

---

## 7. Case 05: Automatic Headlights

### 7.1. Purpose

---

- Make your Cutebot turn on its headlights automatically in the darkness.

### 7.2. Materials

---

- 1 x Cutebot Kit

### 7.3. Software Platform

---

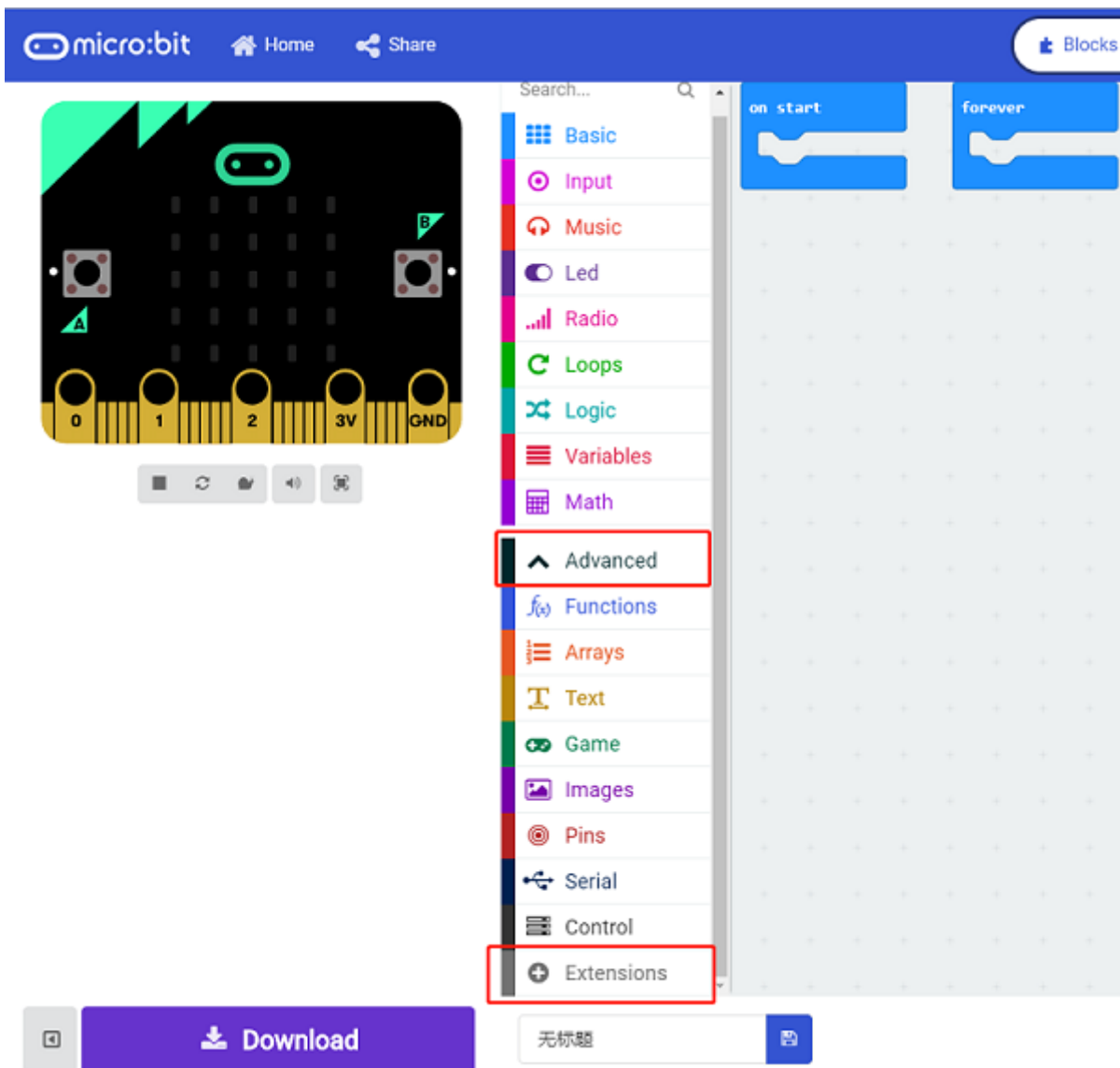
MicroSoft makecode

### 7.4. Programming

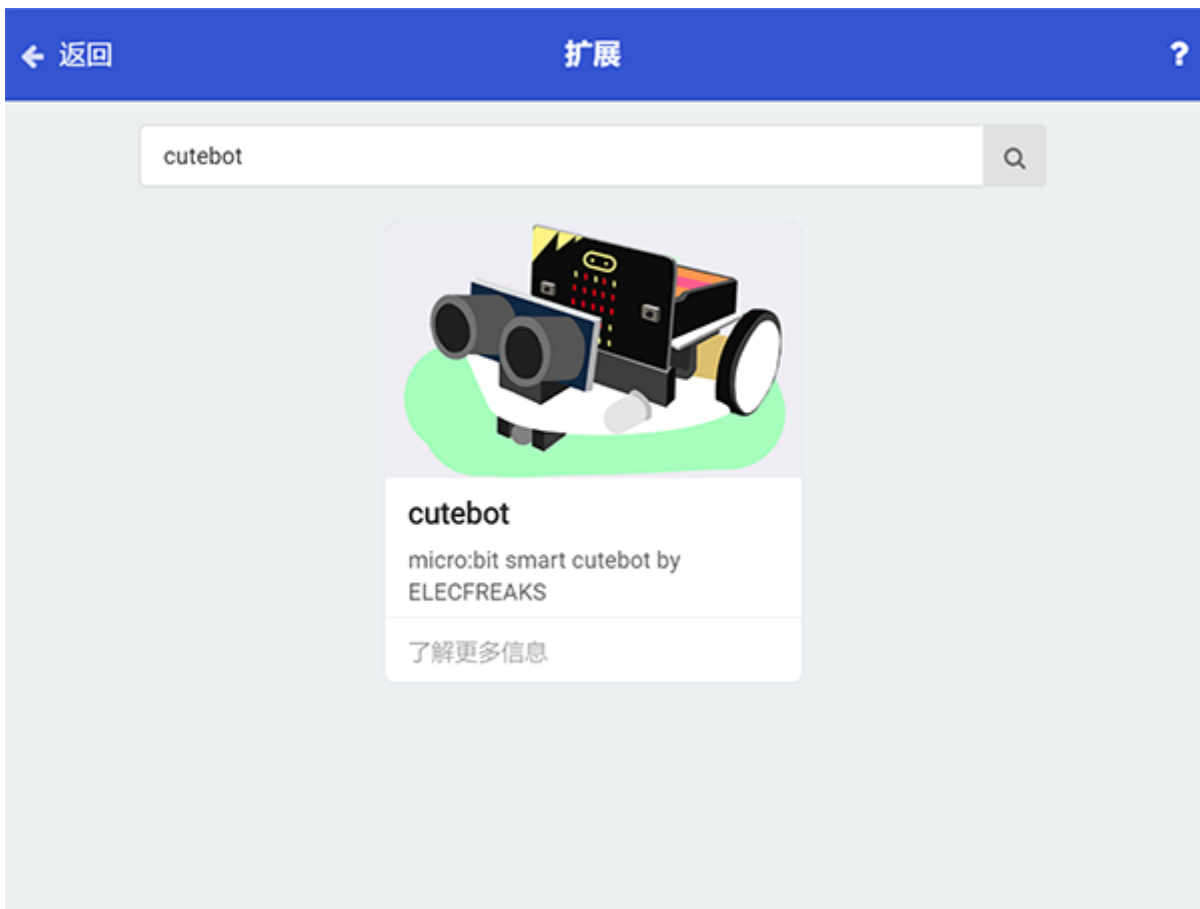
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Set go straight at full speed in `On start`.



## Step 3

- Drag "If...else..." brick into `forever` brick to judge if the light level is below `10`, if yes, set the value of both RGB LEDs as `255`, (The combined light is white).

- While the value is over  , set the value of both RGB LEDs as  to turn off the lights.

```
forever
  if light level < 10 then
    LED Left_RGB R: 255 G: 255 B: 255
    LED Right_RGB R: 255 G: 255 B: 255
  else
    LED Left_RGB R: 0 G: 0 B: 0
    LED Right_RGB R: 0 G: 0 B: 0
```

## Programming

Links: [https://makecode.microbit.org/\\_EYaDJkH4WCff](https://makecode.microbit.org/_EYaDJkH4WCff)

You can also download it directly below:

▶ Simulator 🧩 Blocks JS JavaScript ▼ 🔗 Edit

---

## **7.5. Result**

---

- The headlights turn on automatically when going into the darkness and turn off after passing the darkness area.

## **7.6. Exploration**

---

- How to program to make the lights turn on in different colors when going into the darkness in different time? (The value of RGB helps to set the color)

## **7.7. FAQ**

---

## **7.8. Relevant Files**

---

## 8. Case 06: Steering&Clearance Lamps

### 8.1. Purpose

---

- Make your Cutebot turn on its steering and clearance lamps when making a turn.

### 8.2. Materials

---

- 1 x Cutebot Kit

### 8.3. Software Platform

---

MicroSoft makecode

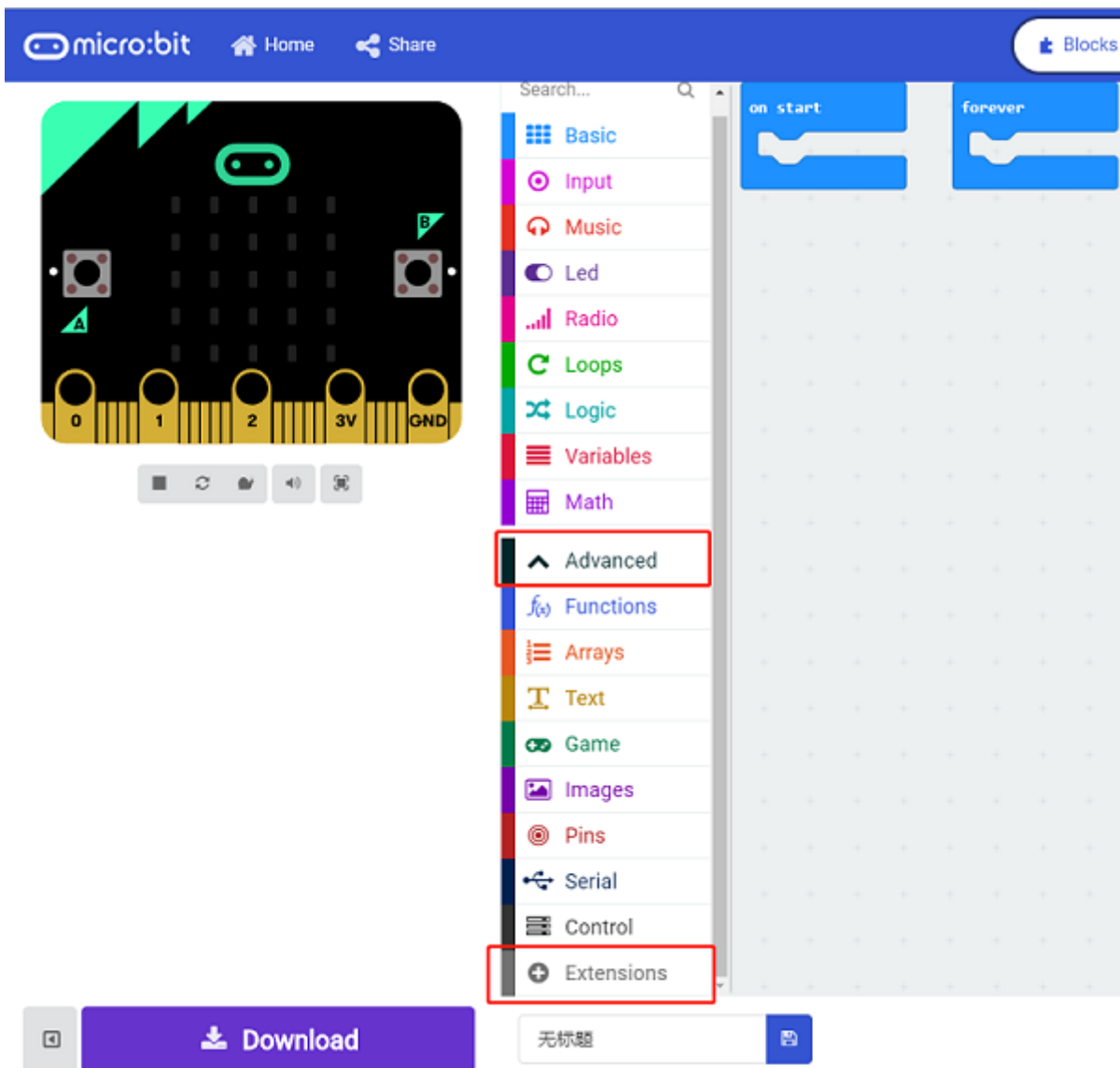
### 8.4. Programming

---

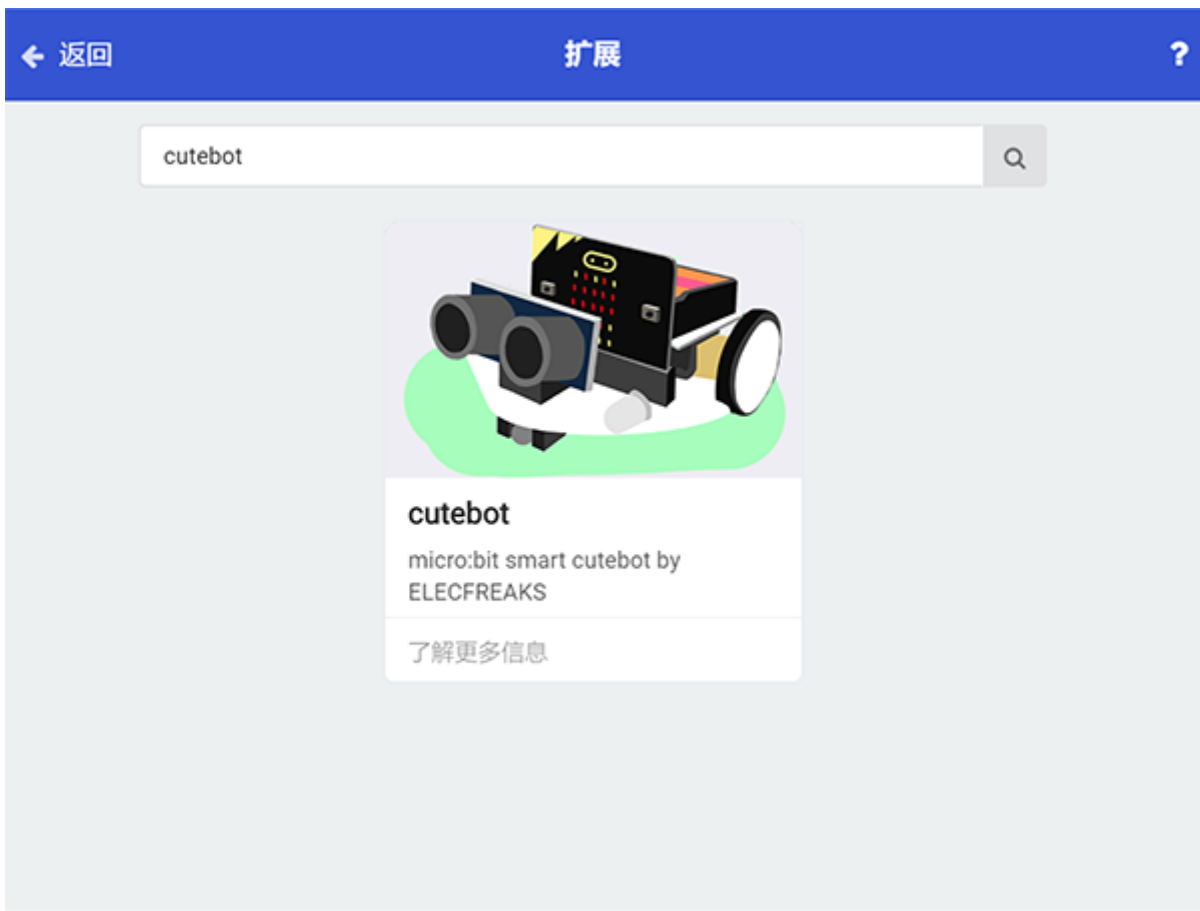
#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.





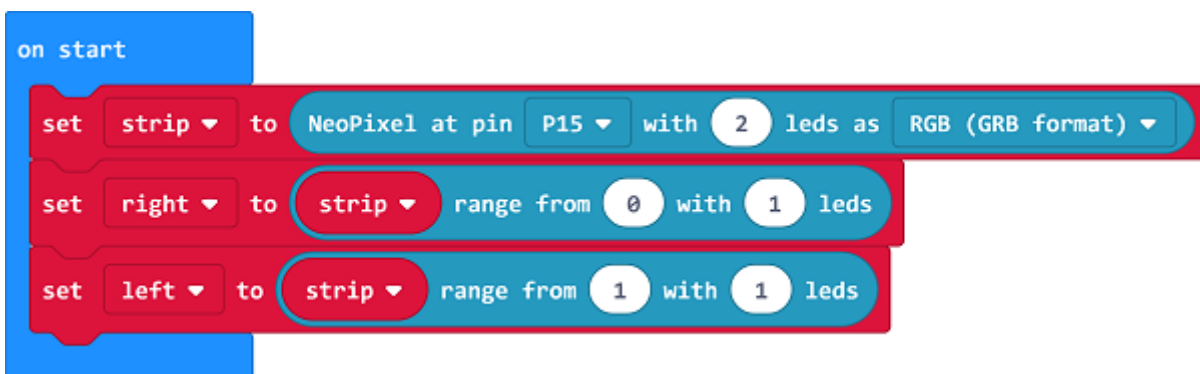
- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Set `P15` as the two Rainbow LEDs connection port in the `On start` brick.
- Set the `right` LED ranging from `0` with `1` LED.
- Set the `left` LED ranging from `0` with `1` LED.



## Step 3

- Drag the `repeat` brick to `on button A pressed` brick, set the color of the right clearance lamp in yellow and the left LED in yellow(Controlled by RGB), then pause 500ms and turn off the clearance lamp and LED on the left side to complete the first flashing.

```
on button A pressed
  repeat 5 times
    do
      right show color yellow
      LED Right_RGB R: 255 G: 255 B: 0
      pause (ms) 500
      right show color black
      LED Right_RGB R: 0 G: 0 B: 0
      pause (ms) 500
```

#### Step 4

- Program in the `on button B pressed` brick in the same way, please note the right side should be changed to the left side.

```
on button B pressed
repeat 5 times
do
  left show color yellow
  LED Left_RGB R: 255 G: 255 B: 0
  pause (ms) 500
  left show color black
  LED Left_RGB R: 0 G: 0 B: 0
  pause (ms) 500
```

## Programming

Links: [https://makecode.microbit.org/\\_FzDFDbCcHfVP](https://makecode.microbit.org/_FzDFDbCcHfVP)

You can also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▾    [🔗 Edit](#)

---

## **8.5. Result**

---

- After pressing button A, the LED and clearance lamp on the right side flashes 5 times.
- After pressing button B, the LED and clearance lamp on the left side flashes 5 times.

## **8.6. Exploration**

---

- How to program to press button A for turning on both lights while turning off by pressing button B ?

## **8.7. FAQ**

---

## **8.8. Relevant Files**

---

## 9. Case 07: Fall-arrest Cutebot

### 9.1. Purpose

---

- The Cutebot reverses quickly when detecting the edge of a table and goes forward after making a turn.

### 9.2. Materials

---

- 1 x Cutebot Kit

### 9.3. Software Platform

---

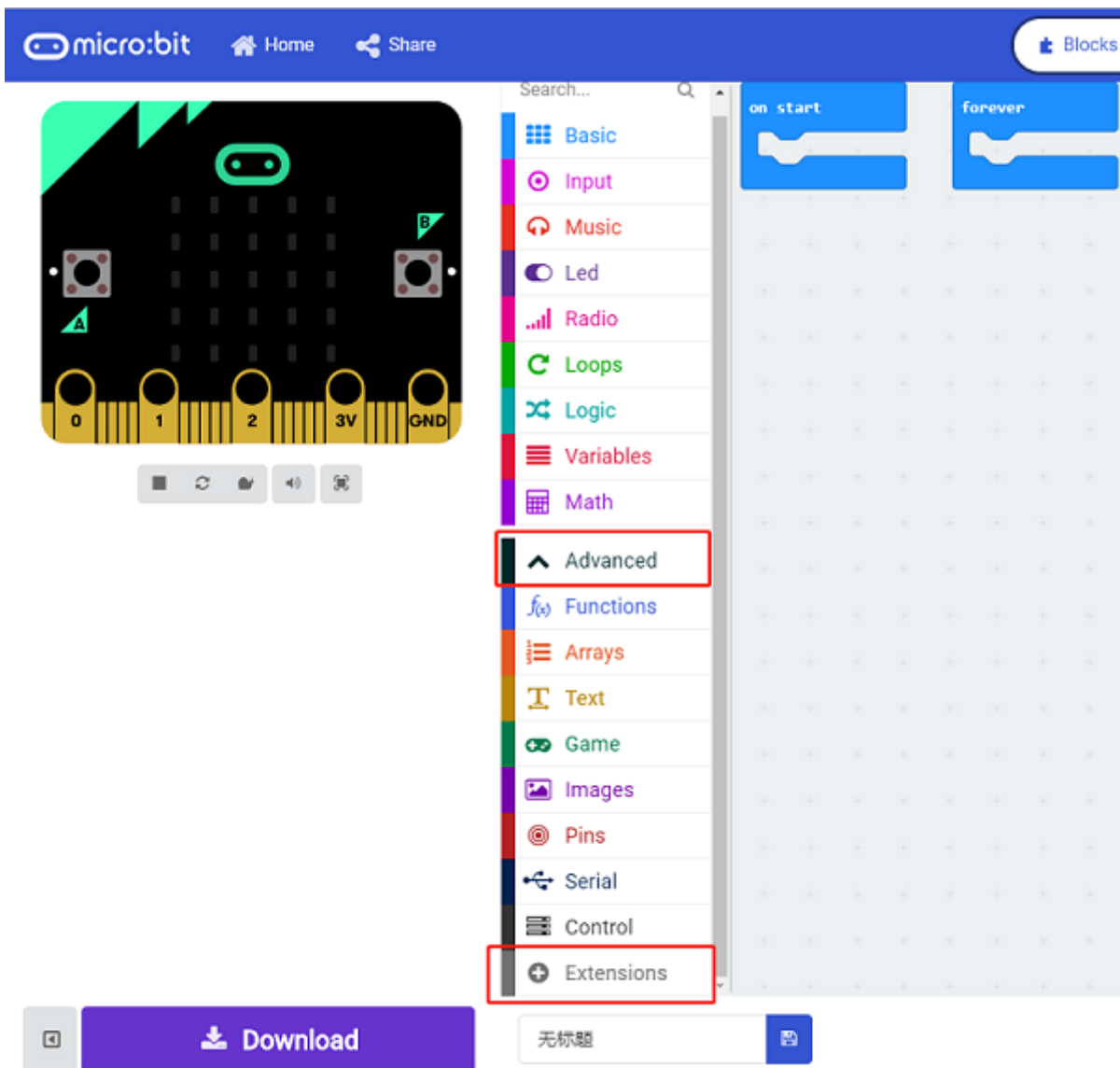
MicroSoft makecode

### 9.4. Programming

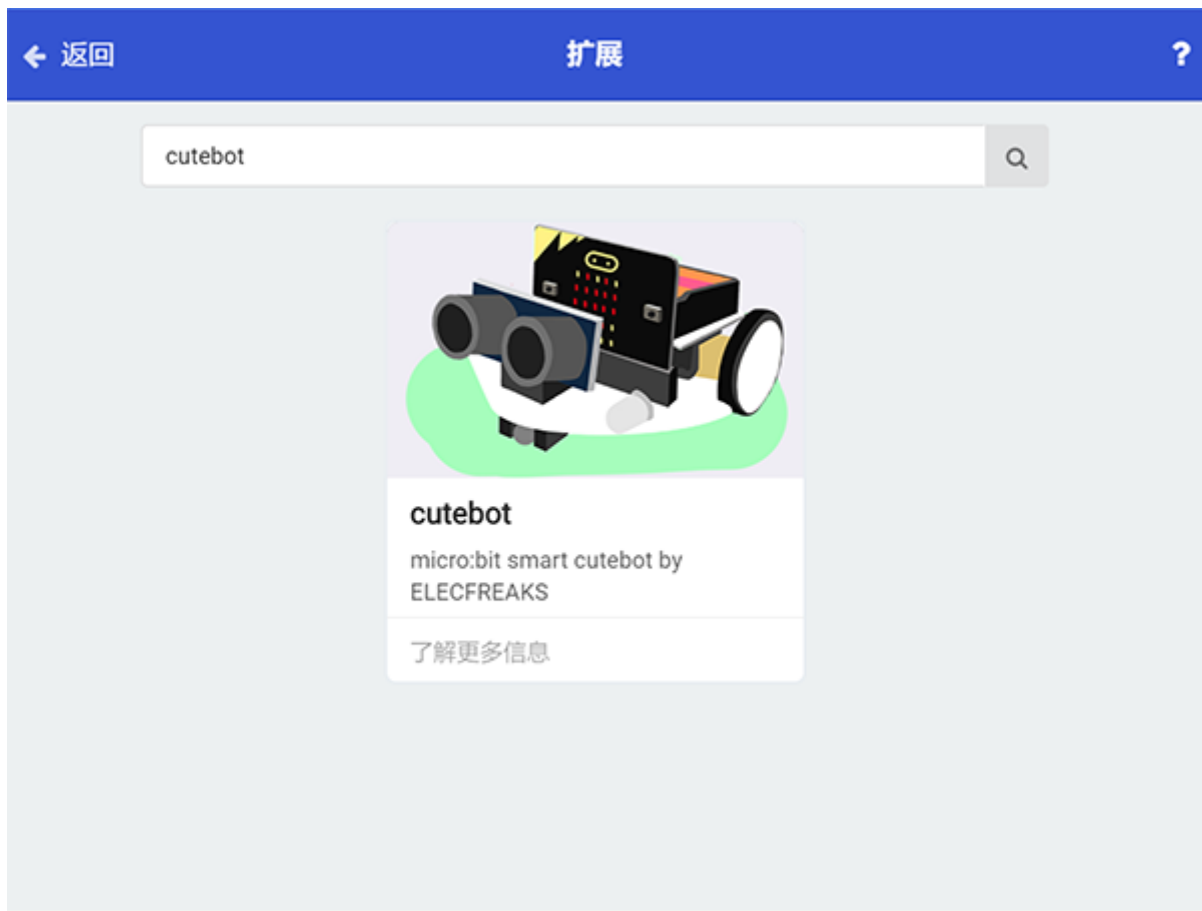
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.

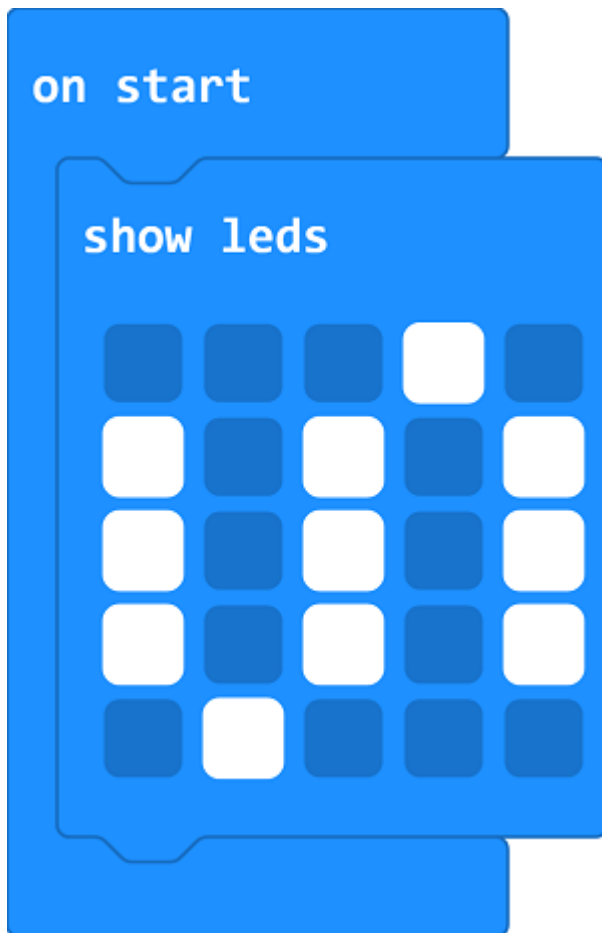


Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Choose “show icon” in the `On start` brick.





### Step 3

- Drag `if` brick into `forever` brick and judge if both of the line-tracking sensors are detecting the edges of the black line, then set the right wheel speed to `20`.
- If not, set the speed of both wheels as `-50` to reverse, pause `300ms` and keep the left wheel still but the right wheel moves at a random speed from 50~100 and lasts 100ms.
- Set the speed of both wheels to 0 and pause 1s to move forward again.

```
forever
  if not tracking state is [ ] then
    set left wheel speed 20 right wheel speed 20
  else
    set left wheel speed -50 right wheel speed -50
    pause (ms) 300
    set left wheel speed 0 right wheel speed pick random 50 to 100
    pause (ms) 100
    set left wheel speed 0 right wheel speed 0
    pause (ms) 1000
```

## Programming

Links: [https://makecode.microbit.org/\\_LvweWcb3J72u](https://makecode.microbit.org/_LvweWcb3J72u)

You can also download it directly below:

[▶ Simulator](#) [🧩 Blocks](#) [JS JavaScript](#) [Edit](#)

---

## **9.5. Result**

---

- The Cutebot reverses quickly when detecting the edge of a table and goes forward after making a turn.

## **9.6. Exploration**

## **9.7. FAQ**

---

## **9.8. Relevant Files**

---

## 10. Case 08: Run Along the Black Line

### 10.1. Purpose

---

- The Cutebot runs along the black line.

### 10.2. Materials

---

- 1 x Cutebot kit
- 1 x Line-tracking Map(Homemade or enclosed in the Cutebot Kit)

### 10.3. Software Platform

---

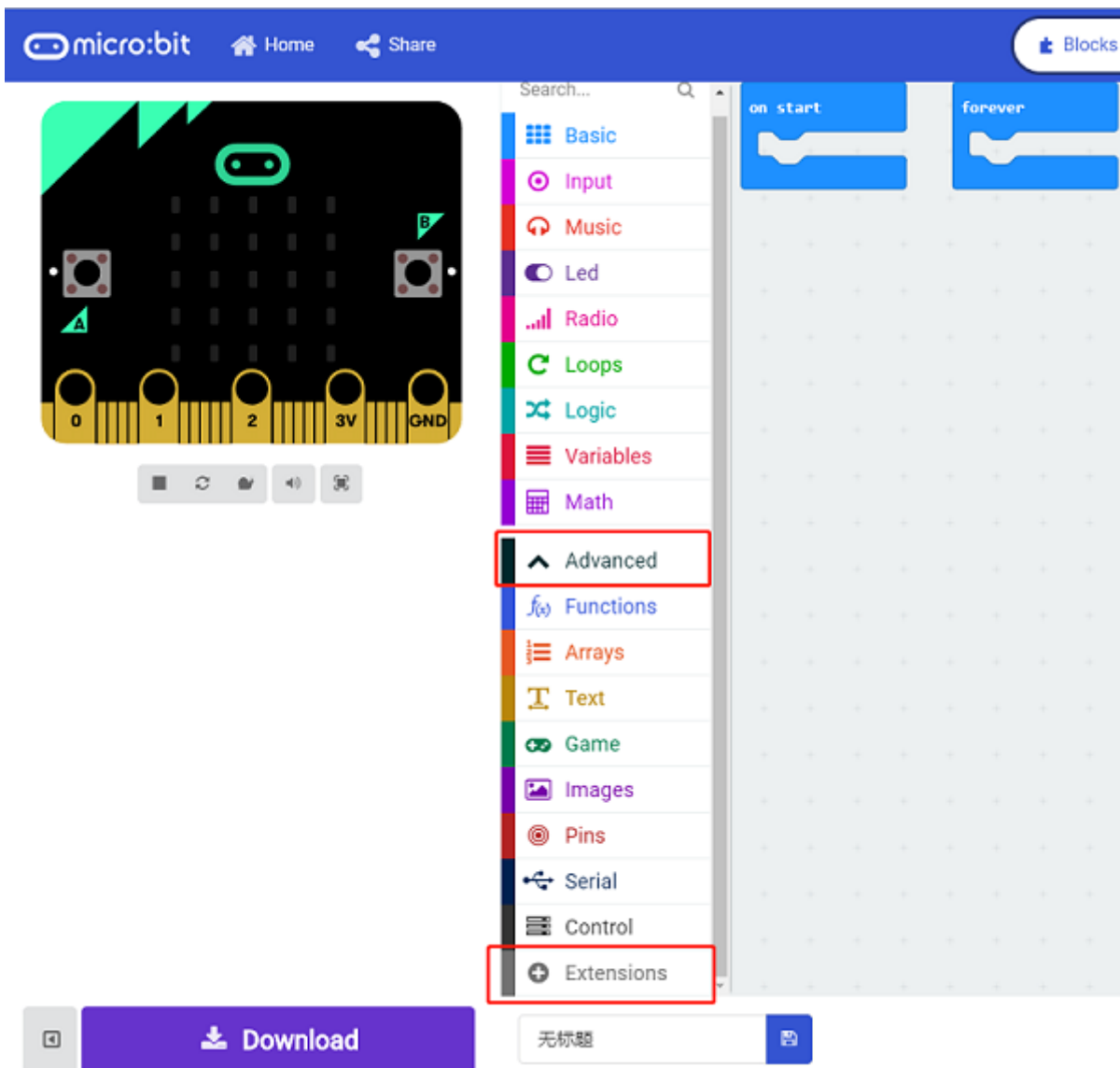
MicroSoft makecode

### 10.4. Programming

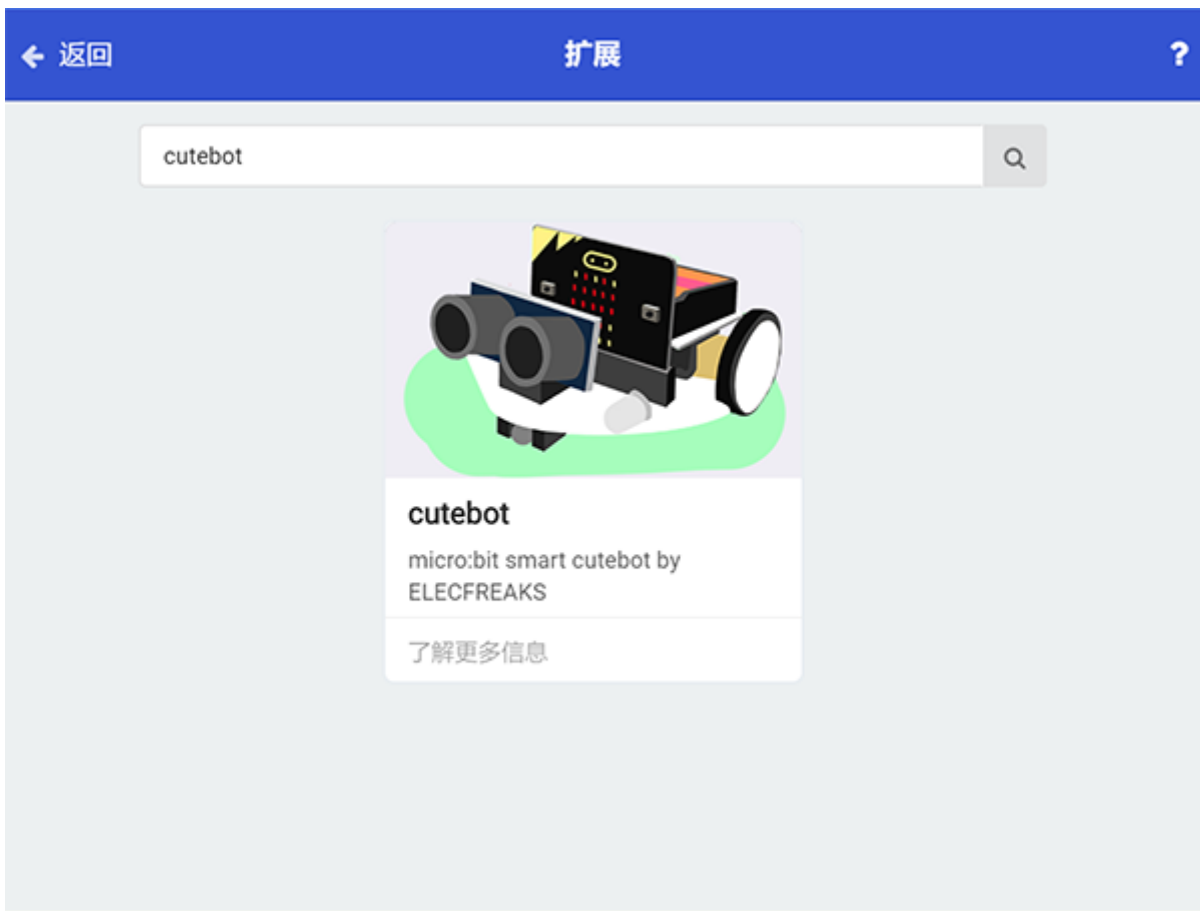
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Choose “show icon” in the `On start` brick.



## Step 3

- Drag three `if` bricks into the `Forever` brick.
- Judge if the status of line-tracking sensors is `○ ●`, saying the left probe doesn't detect the black line while the right probe detects the black line.
- Set the left wheel speed to `50` and right to `25`, make a right turn by the different speed of the two wheels and go back to the black line.
- Judge if the status of line-tracking sensors is `● ○` and make a left turn to go back to the black line.
- When the status is `● ●` that means the Cutebot runs along with the black line at the speed of `50`.

```

forever
  if tracking state is ○ ● then
    set left wheel speed 50 right wheel speed 25
  if tracking state is ● ○ then
    set left wheel speed 25 right wheel speed 50
  if tracking state is ● ● then
    set left wheel speed 50 right wheel speed 50

```

## Programming

Links: [https://makecode.microbit.org/\\_brF6tzUKwess](https://makecode.microbit.org/_brF6tzUKwess)

You can also download it directly below:

▶ Simulator
🧱 Blocks
JS JavaScript
▼
🔗 Edit

---

## **10.5. Result**

---

- The Cutebot runs along the black line and will adjust to run back to the black line if any deviation happens.

## **10.6. Exploration**

---

- How to program to make the Cutebot run in the white background of the map excluding the black line circle part?

## **10.7. FAQ**

---

## **10.8. Relevant Files**

---



# 11. Case 09: Autonomous Obstacle Avoidance

## 11.1. Purpose

---

- The Cutebot avoids the obstacles automatically to move forward.

## 11.2. Materials

---

- 1 x Cutebot Kit
- 1 x Ultrasonic Sensor

## 11.3. Software Platform

---

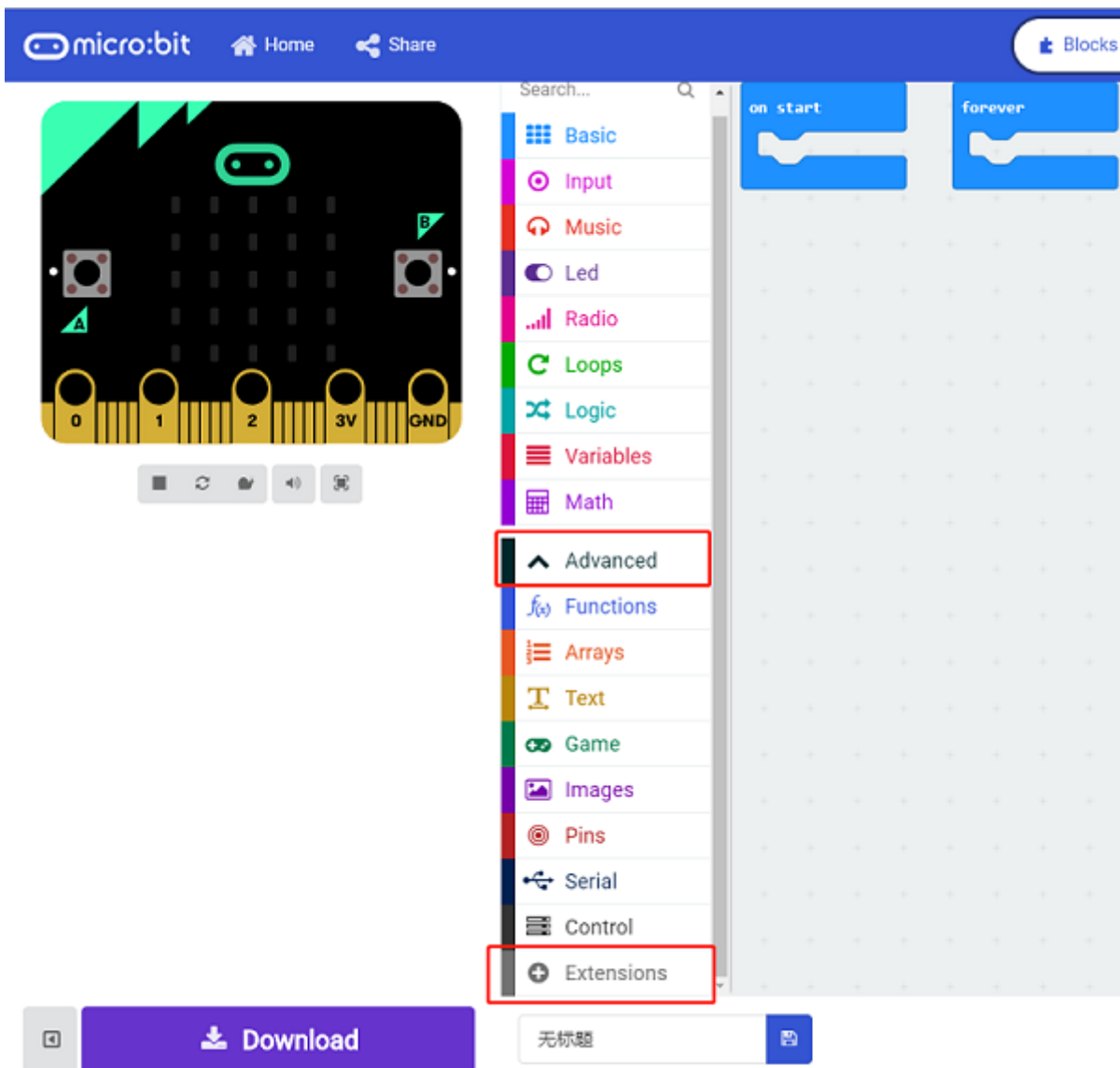
MicroSoft makecode

## 11.4. Programming

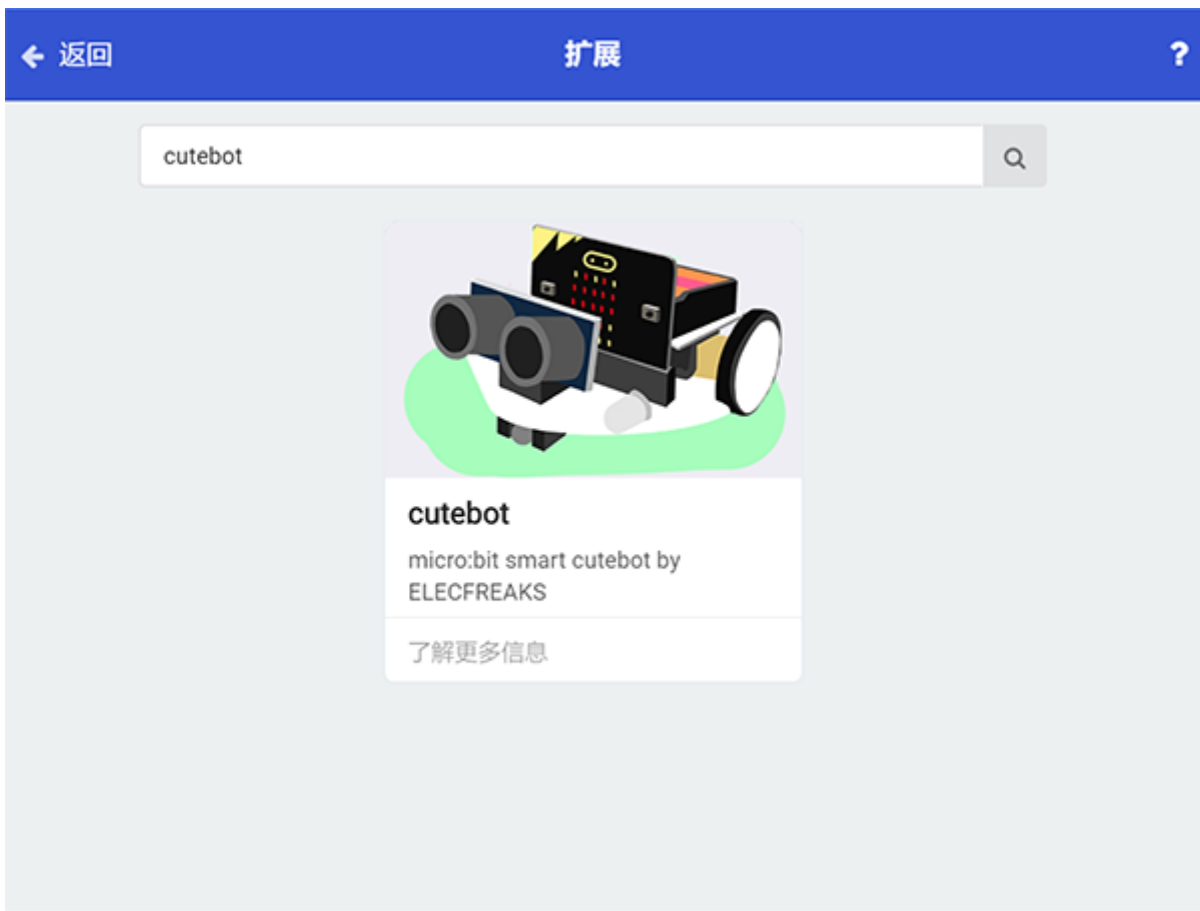
---

### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



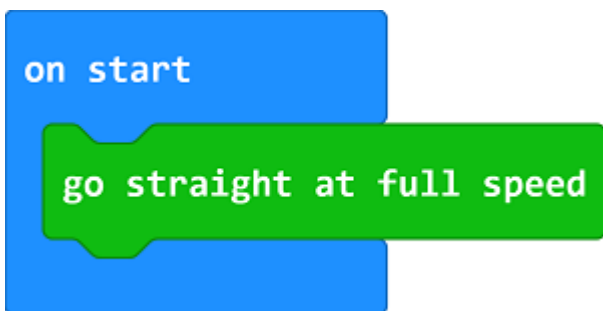
- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Drag “go straight at full speed” brick into the `on start` brick.



## Step 3

- Set a `Sonar` variable to save the detected `Cm` value in the `Forever` brick.
- If the detected value is between `2` and `20` which means there is obstacle being detected in the front 20cm far, set the left wheel speed to `0` and right to `-50`, make a right turn at a random time to complete an obstacle avoidance.
- If not, move forward at its full speed.

```
forever
  set sonar to HC-SR04 Sonar unit cm
  if sonar < 20 = sonar > 2 then
    set left wheel speed 0 right wheel speed -50
    pause (ms) pick random 100 to 200
  else
    go straight at full speed
```

## Programming

Links: [https://makecode.microbit.org/\\_hijb4L6ttgfc](https://makecode.microbit.org/_hijb4L6ttgfc)

You can also download it directly below:

[▶ Simulator](#) [🧩 Blocks](#) [JS JavaScript](#) [Edit](#)

---

## 11.5. Result

---

- The Cutebot moves forward at its full speed and will make a right turn to keep going if any obstacle being detected.

## **11.6. Exploration**

---

- Why should the detected value be over 2cm ?

## **11.7. FAQ**

---

## **11.8. Relevant Files**

---

## 12. Case 10: Car Following with A Fixed Distance

### 12.1. Purpose

---

- The Cutebot moves with a fixed distance between the car and your hands.

### 12.2. Materials

---

- 1 x Cutebot Kit
- 1 x Ultrasonic Sensor

### 12.3. Software Platform

---

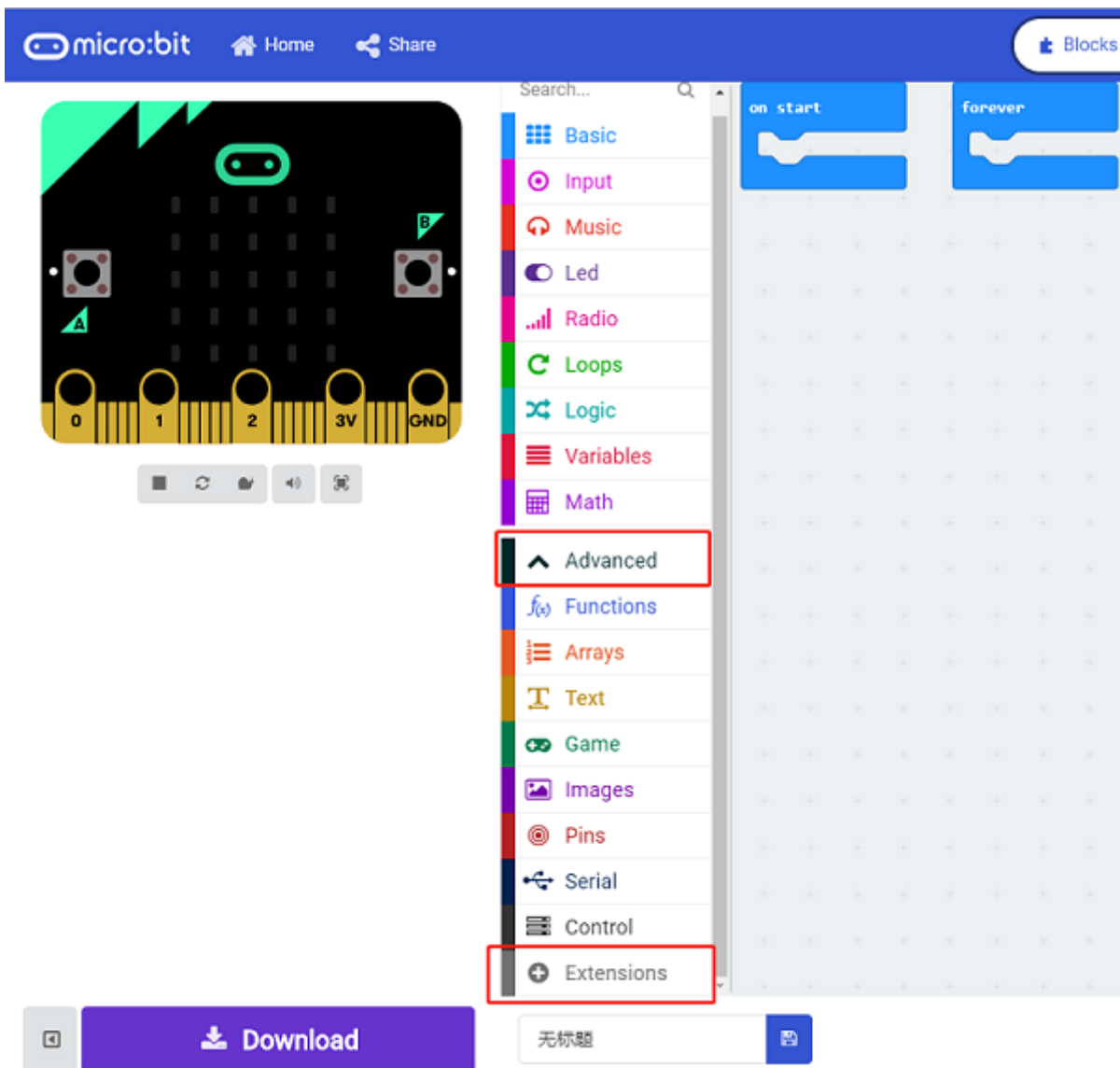
MicroSoft makecode

### 12.4. Programming

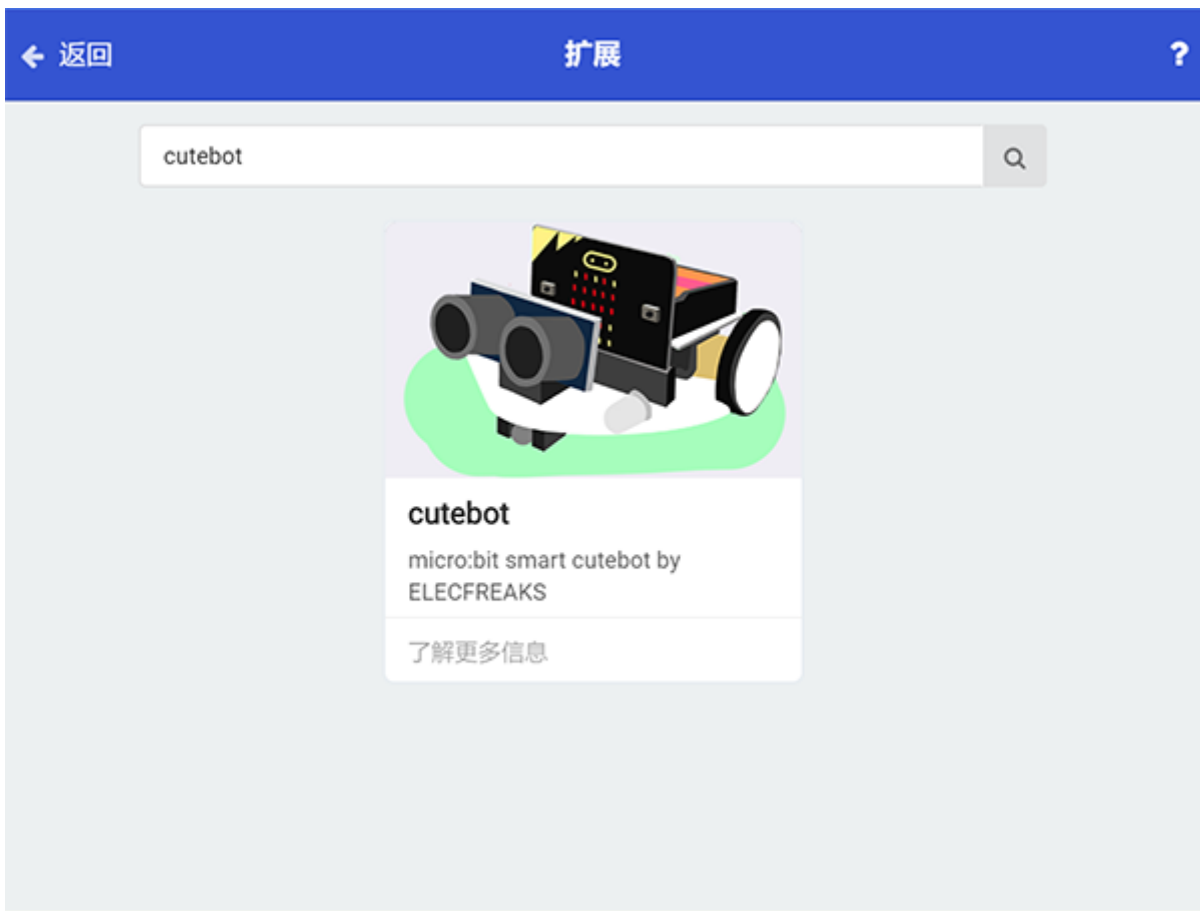
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2

- Drag the “set left wheel speed and right wheel speed” into the `On start` brick.



## Step 3

- Set a `Sonar` variable to save the detected `Cm` value in the `Forever` brick.
- If the detected value is between `5` and `10`, the car stops moving.
- If the detected value is below `5`, the car reverses because of the short distance with the hands.
- If not any, the car moves forward to catch up with the hands because of the far distance with the hands and then stay still .



```
forever
  set sonar to HC-SR04 Sonar unit cm
  if sonar > 5 and sonar < 10 then
    set left wheel speed 0 right wheel speed 0
  else if sonar < 5 then
    set left wheel speed -30 right wheel speed -30
  else
    set left wheel speed 30 right wheel speed 30
```

## Programming

Links: [https://makecode.microbit.org/\\_gRtPkmPOq0cM](https://makecode.microbit.org/_gRtPkmPOq0cM)

You can also download it directly below:

[▶ Simulator](#) [📌 Blocks](#) [JS JavaScript](#) [Edit](#)

---

## 12.5. Result

- 
- The Cutebot adjusts itself to keep a fixed distance with your hands.

## **12.6. Exploration**

## **12.7. FAQ**

---

## **12.8. Relevant Files**

---

## 13. Case 11: micro:bit Remote Control

### 13.1. Purpose

---

- Use another micro:bit as a remote control for your Cutebot.
- Both micro:bit needs to be programmed.

### 13.2. Materials

---

- 1 x Cutebot Kit
- 1 x micro:bit

### 13.3. Software Platform

---

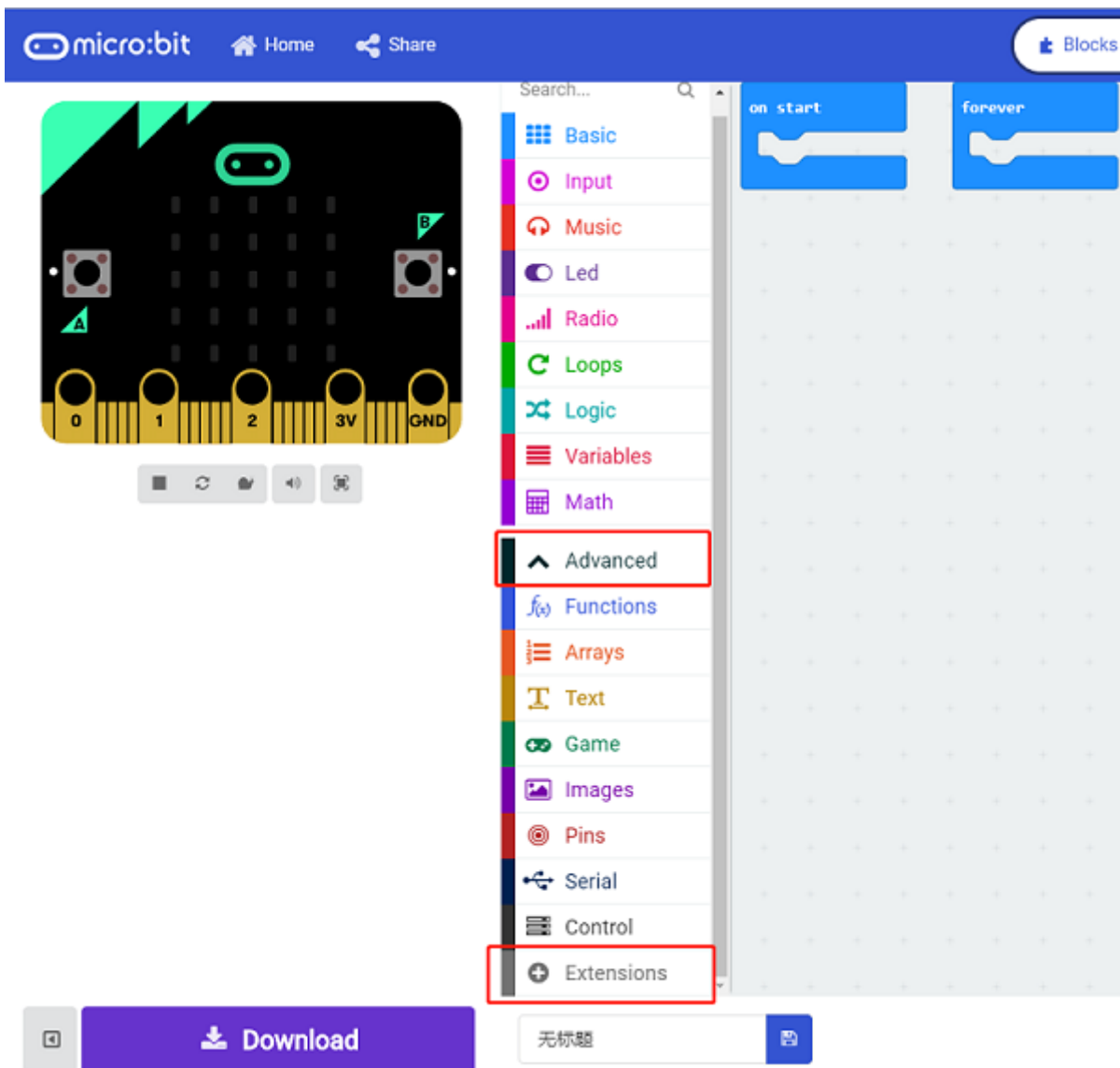
MicroSoft makecode

### 13.4. Programming

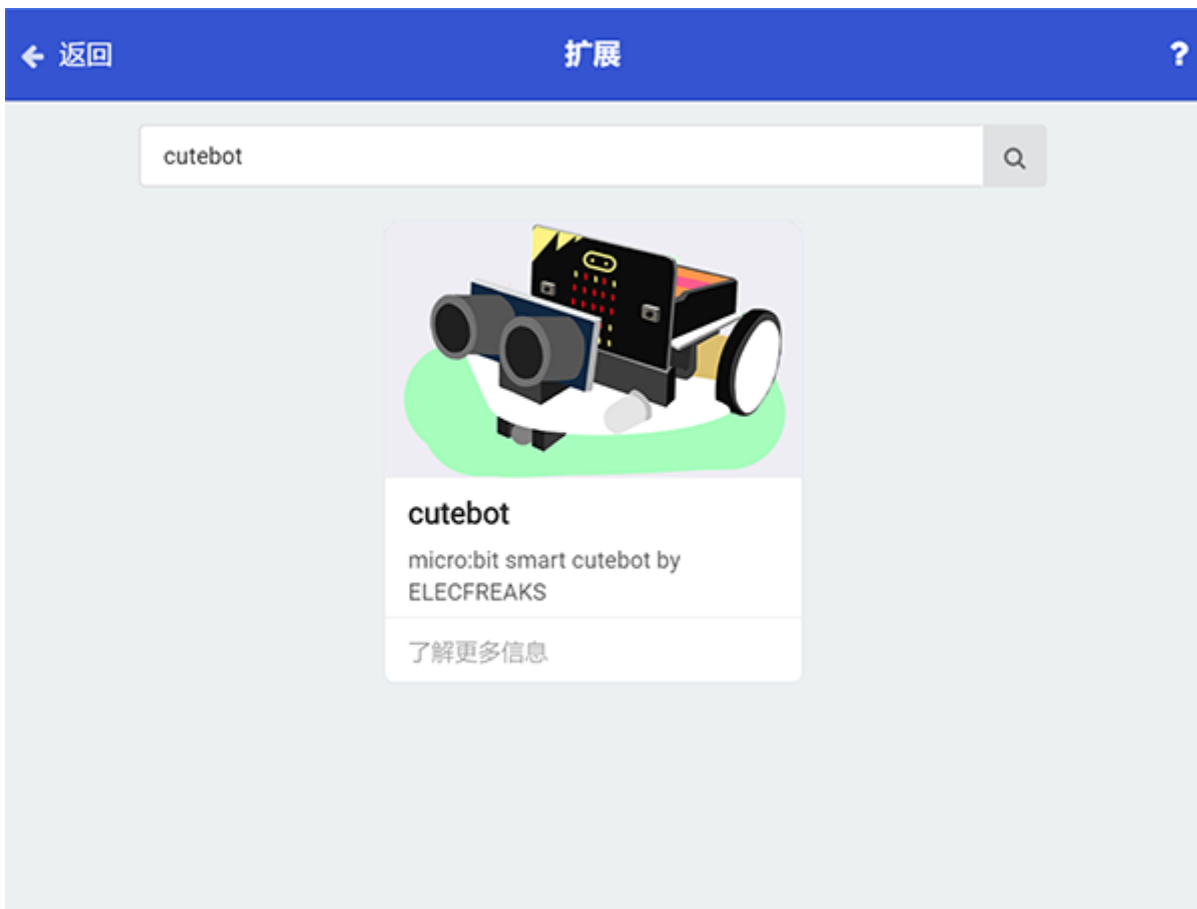
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



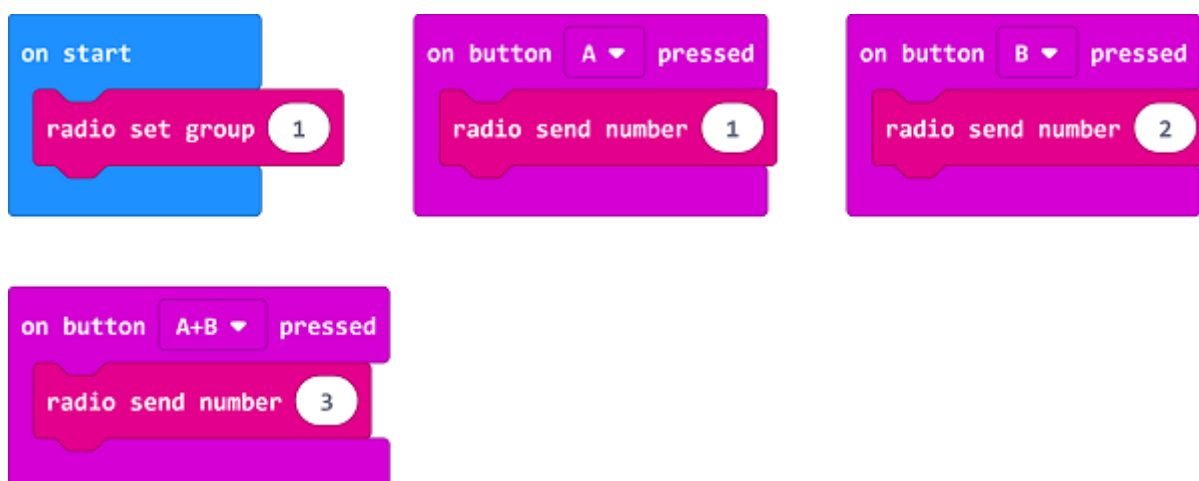
- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2: Remote Control Programming

- Set the “radio set group” to **1** in the **On start** brick.
- Send radio number in **1** when pressing button A.
- Send radio number in **2** when pressing button B.
- Send radio number in **3** when pressing button A+B.



## Programming

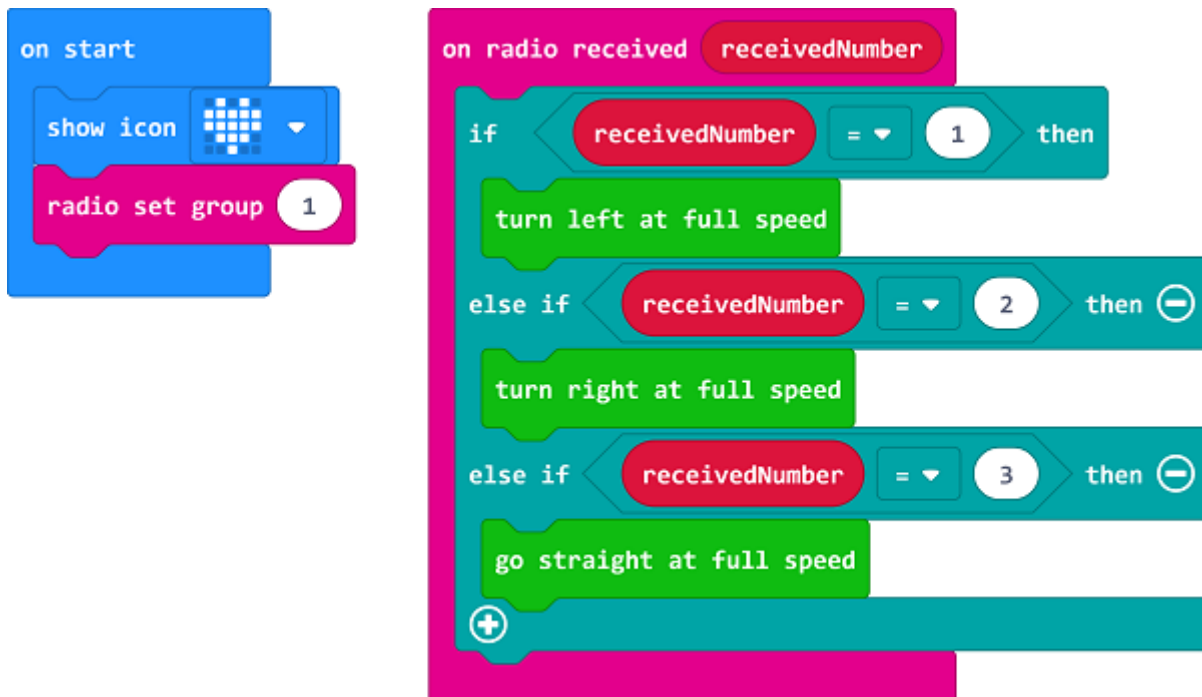
links: [https://makecode.microbit.org/\\_VbpXHCC4jW3T](https://makecode.microbit.org/_VbpXHCC4jW3T)

You can also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼    ↗ Edit

### Step 3: Cutebot Programming

- Drag “show icon” brick into the `On start` brick and set the “radio set group” to `1`. Items must be the same with the remote control for the correct match.
- Drag three “if” bricks into the `on radio received` brick and judge if the received number is `1`, `2` or `3`.
- When the received number is `1`, turn left.
- When the received number is `2`, turn right.
- When the received number is `3`, go straight.



## Programming

Links: [https://makecode.microbit.org/\\_MreDFh8se1LK](https://makecode.microbit.org/_MreDFh8se1LK)

You can also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼    [Edit](#)

[Microsoft MakeCode](#) | [Terms of Use](#) | [Privacy](#) | [Download](#)

## 13.5. Result

- When button A+B being pressed on the remote control, the Cutebot goes straight.

- When button A being pressed on the remote control, the Cutebot turns left.
- When button B being pressed on the remote control, the Cutebot turns right.

## **13.6. Exploration**

---

## **13.7. FAQ**

---

## **13.8. Relevant Files**

---



## 14. Case 12: Remote Control the Cutebot with micro:bit Accelerometer

### 14.1. Purpose

---

- Use the accelerometer in another micro:bit to remote control the Cutebot for the direction and speed.
- Both of the micro:bit need to be programmed.

### 14.2. Materials

---

- 1 x Cutebot Kit
- 1 x micro:bit

### 14.3. Software Platform

---

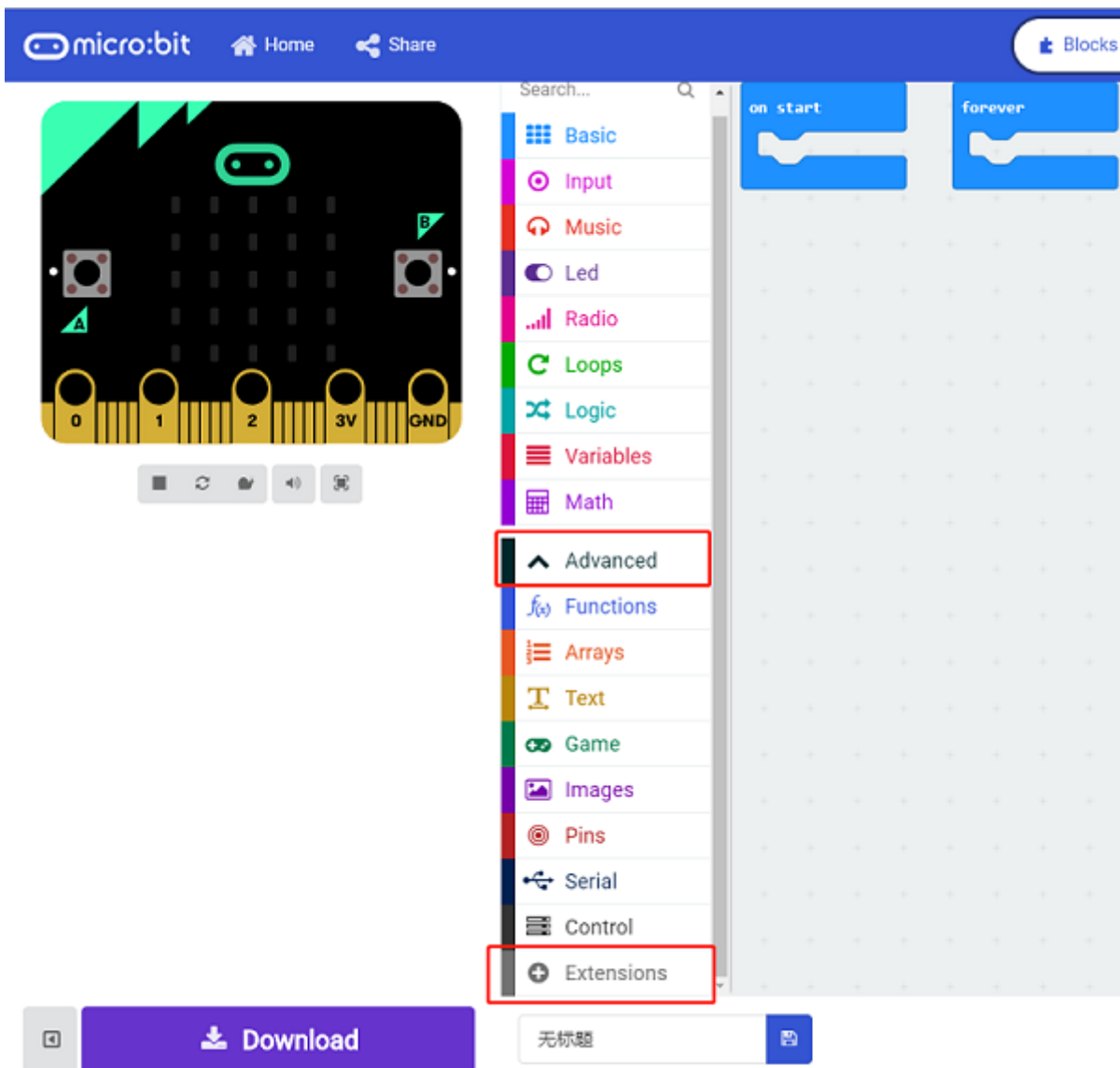
MicroSoft makecode

### 14.4. Programming

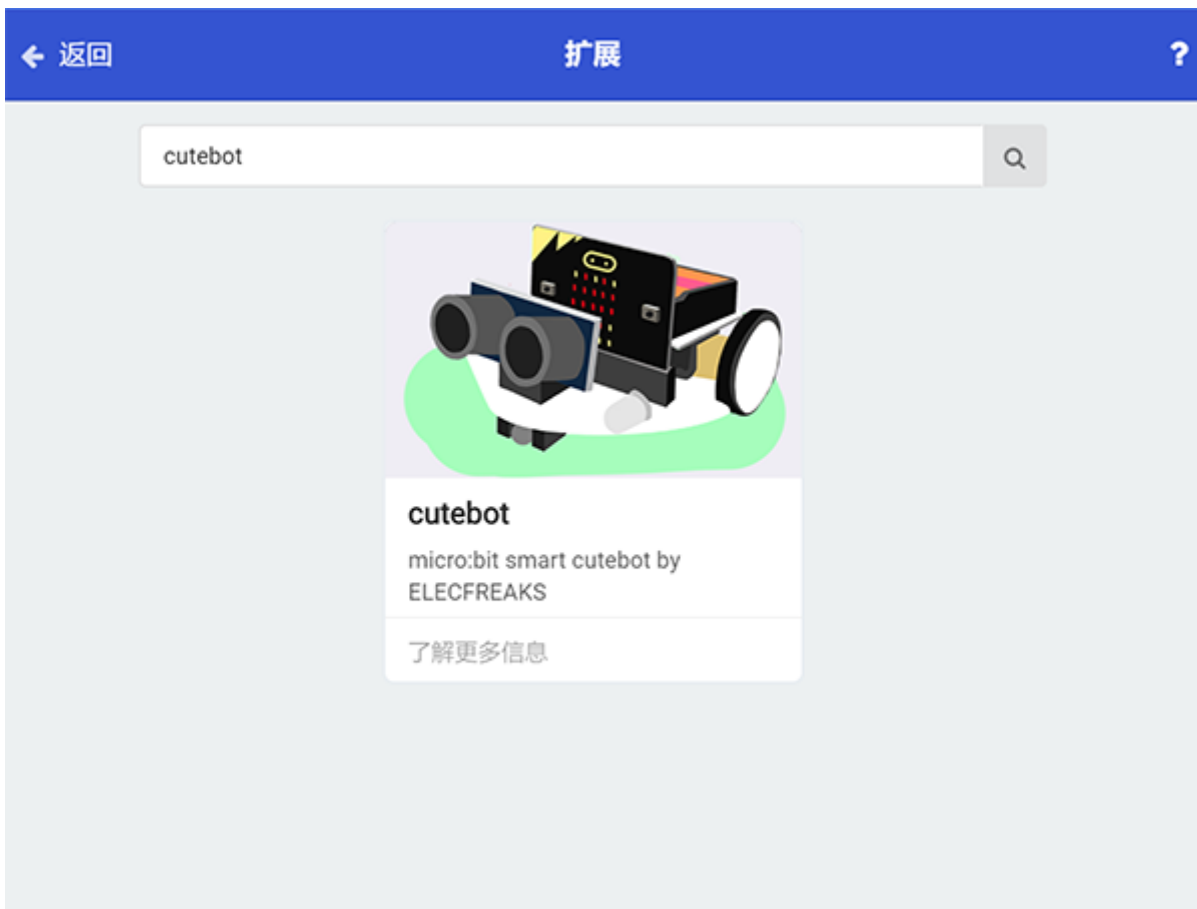
---

#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.



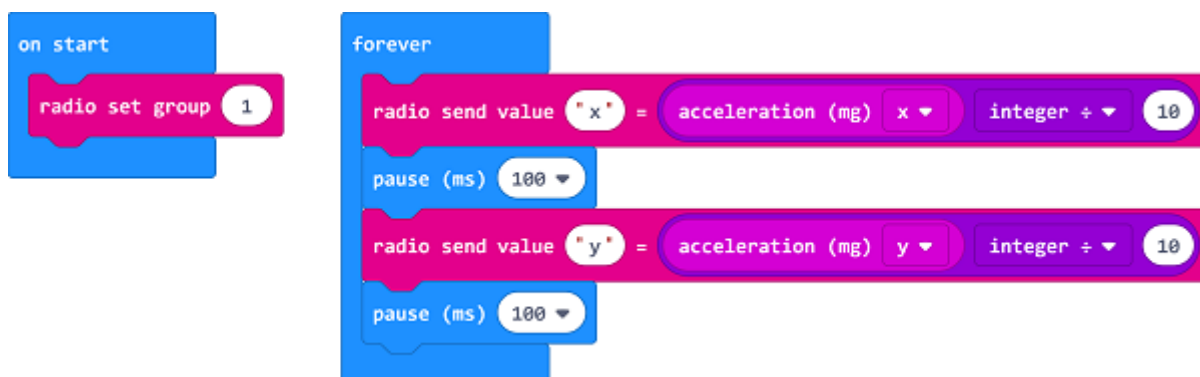
- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2: Remote Control Programming

- Set “radio set group” to `1` in the `On start` brick.
- Set `x` whose value is given by “acceleration (mg) x” exactly divides `10` to the radio value in `forever` brick.
- Set `y` whose value is given by “acceleration (mg) y” exactly divides `10` to the radio value in `forever` brick.
- The scope of the acceleration value is `0 ~ 1024`, which can be regarded roughly as the speed value in `0 ~ 100` after dividing `10`.



## Programming

Links: [https://makecode.microbit.org/\\_0Xo6EX0weMVF](https://makecode.microbit.org/_0Xo6EX0weMVF)

You can also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼    ↗ Edit

### Step 3: Cutebot Programming

- Set the “radio set group” to `1` in the `On start` brick. Items must be the same with the remote control for the correct match.
- Drag two “if” bricks into the `on radio received` brick and judge if the radio received value `name` is `x` or `y`.
- If the radio received value `name` is `x`, it is the data for `X` and then save the `value` in the variable `xValue`.
- If the radio received value `name` is `y`, it is the data for `y` and then save the `value` in the variable `yValue`.
- In `forever` brick, set the left wheel speed to `yValue + xValue` and right wheel speed to `yValue - xValue`.

```
on start
  radio set group 1

on radio received name value
  if name = "x" then
    set xValue to value
  if name = "y" then
    set yValue to value

forever
  set left wheel speed yValue + xValue
  set right wheel speed yValue - xValue
```

## Programming

Links: [https://makecode.microbit.org/\\_6ExC8oRz3i6U](https://makecode.microbit.org/_6ExC8oRz3i6U)

You can also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼    [🔗 Edit](#)

## 14.5. Result

- The moving direction of the Cutebot is controlled by the tilt degree of the micro:bit.
- The tilt angle of the controlling micro:bit controls the speed of the Cutebot.

## 14.6. Exploration

---

## 14.7. FAQ

---

## 14.8. Relevant Files [🔗](#)

---

## 15. Case 13: Remote Control with Joystick:bit

### 15.1. Purpose

---

- Use the joystick:bit to control the Cutebot.

### 15.2. Materials

---

- 1 x Cutebot Kit
- 1 x Joystick:bit2

### 15.3. Software Platform

---

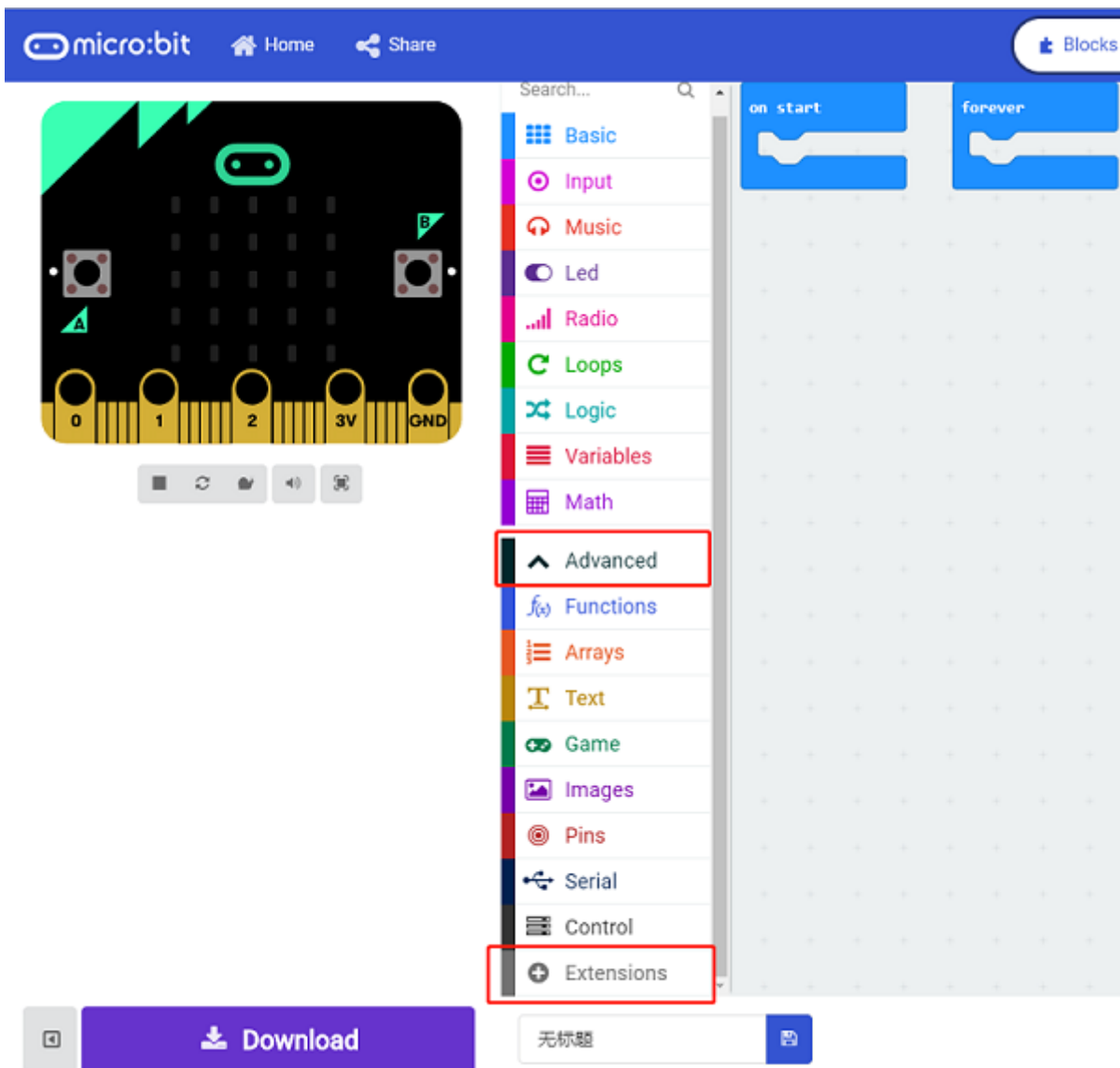
MicroSoft makecode

### 15.4. Programming

---

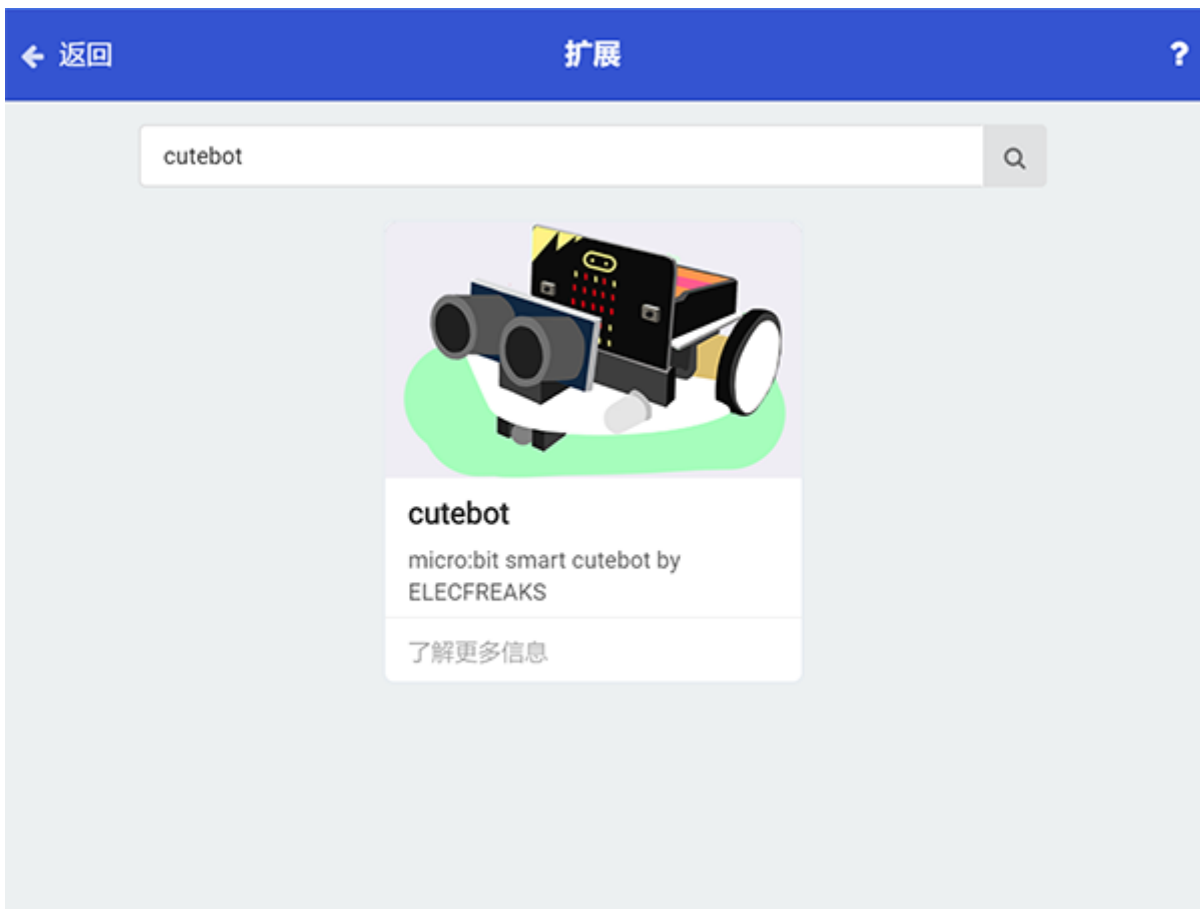
#### Step 1

- Click the “Advanced” to see more choices in the MakeCode drawer.

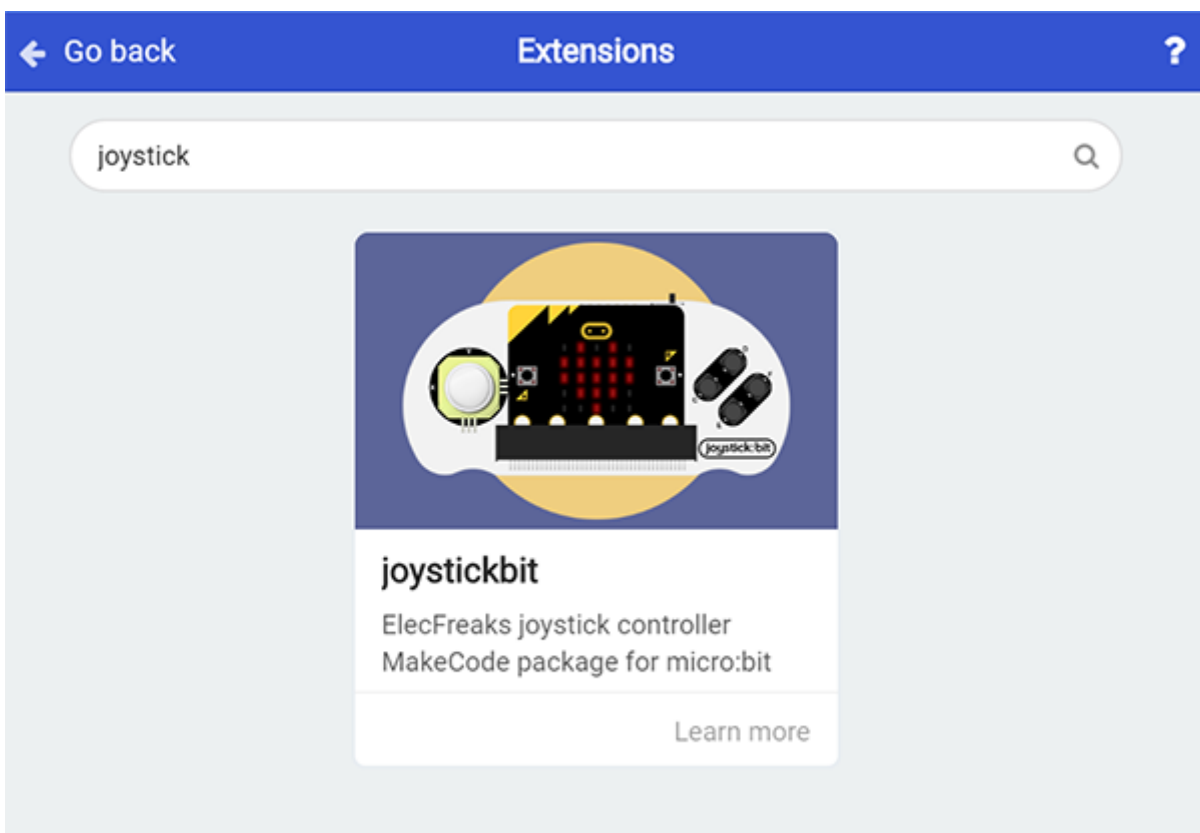


- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `Cutebot` in the dialogue box and download it.





- A codebase is required for Cutebot programming, click “Add Package” at the bottom of the drawer, search `joystick` in the dialogue box and download it.
- 



Note: If you met a tip indicating incompatibility of the codebase, you can continue with the tips or build a new project there.

## Step 2: Joystick:bit Programming

- Set “radio set group” to `1` in the `On start` brick.
- The scope for `X` and `Y` is `0~1023`, the theoretical value is `512` if the rocker is in the middle place, in this way we need to make `0~1023` `map` in the scope of `-100~100`.
- Set `x` whose value is given by “acceleration (mg) x” exactly divides `10` to the radio value in `forever` brick.
- Set `y` whose value is given by “acceleration (mg) y” exactly divides `10` to the radio value in `forever` brick.
- The scope of the acceleration value is `0 ~ 1024`, which can be regarded roughly as the speed value in `0 ~ 100` after dividing `10`.

```
on start
  radio set group 1

forever
  set x to map rocker value of X from low 1023 high 0 to low -100 high 100
  set y to map rocker value of Y from low 1023 high 0 to low -100 high 100
  radio send value "x" = x
  radio send value "y" = y
```

## Programming

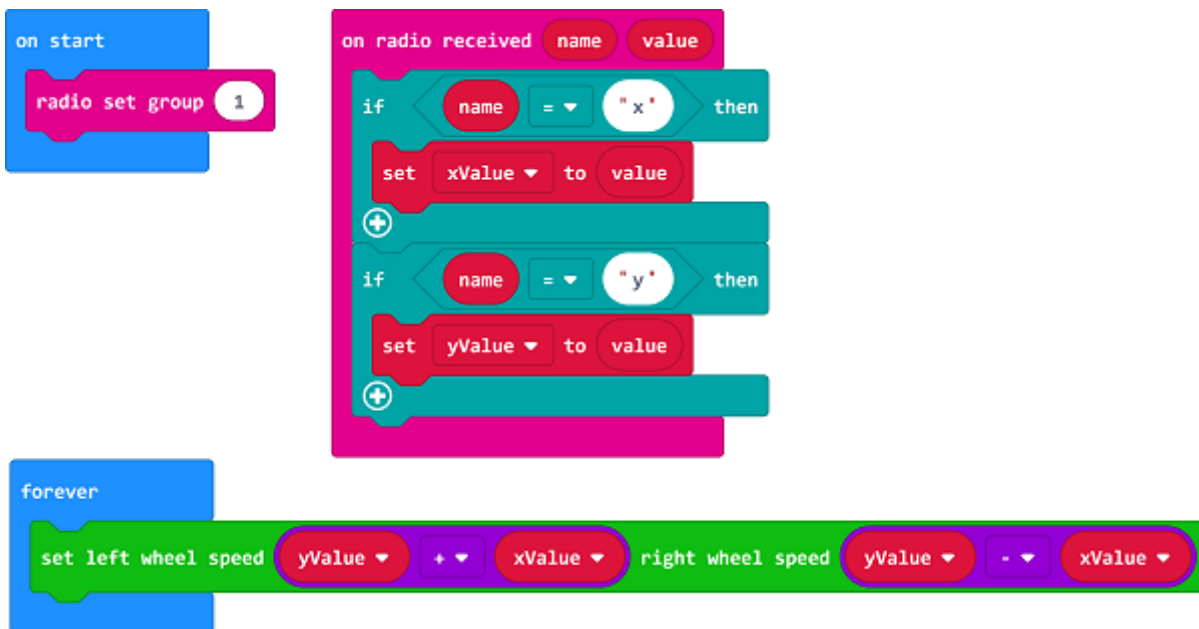
Links: [https://makecode.microbit.org/\\_ag2g2sc22hUp](https://makecode.microbit.org/_ag2g2sc22hUp)

You can also download it directly below:

▶ Simulator    🧱 Blocks    📄 JavaScript    ▼    ↗ Edit

## Step 3: Cutebot Programming

- Set the “radio set group” to `1` in the `On start` brick. Items must be the same with the remote control for the correct match.
- Drag two “if” bricks into the `on radio received` brick and judge if the radio received value `name` is `x` or `y`
- If the radio received value `name` is `x`, it is the data for `x` and then save the `value` in the variable `xValue`.
- If the radio received value `name` is `y`, it is the data for `y` and then save the `value` in the variable `yValue`.
- In `forever` brick, set the left wheel speed to `yValue + xValue` and right wheel speed to `yValue - xValue`.



```

on start
  radio set group 1

on radio received name value
  if name = "x" then
    set xValue to value
  if name = "y" then
    set yValue to value

forever
  set left wheel speed yValue + xValue right wheel speed yValue - xValue
  
```

## Programming

Links: [https://makecode.microbit.org/\\_6ExC8oRz3i6U](https://makecode.microbit.org/_6ExC8oRz3i6U)

You can also download it directly below:

 Simulator
  Blocks
  JavaScript
 
 Edit

## 15.5. Result

---

- The rocker controls the movement of the Cutebot.

## 15.6. Exploration

---

## 15.7. FAQ

---

## 15.8. Relevant Files

---

## 16. Case 14: IR Remote Control Car

### 16.1. Purpose

---

- Use an infrared remote control to give orders to the Cutebot.

### 16.2. Materials

---

- 1 x Cutebot Kit
- 1 x IR Remote Control

### 16.3. Software Platform

---

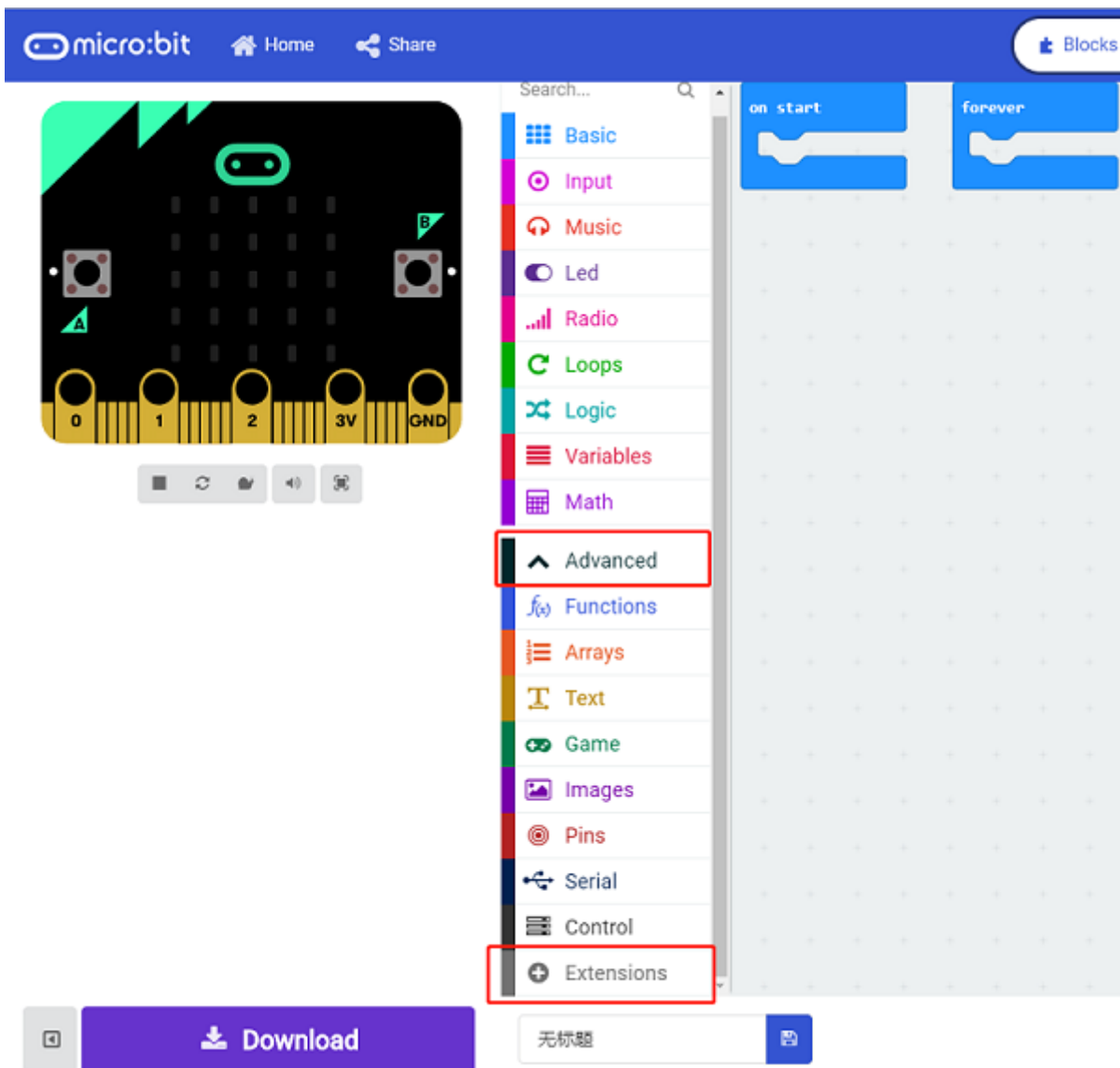
MicroSoft makecode

### 16.4. Programming

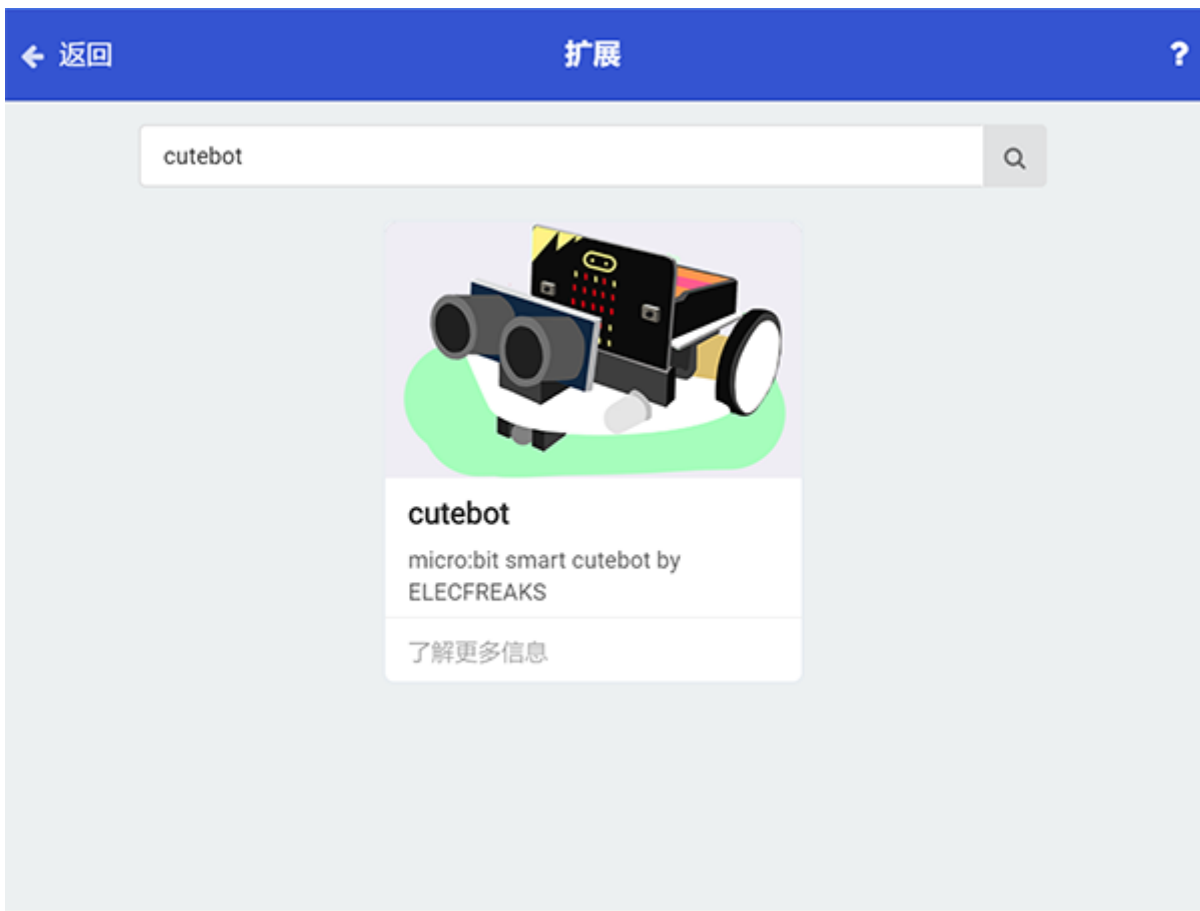
---

#### Step 1

- Click “Advanced” to see more choices in the MakeCode drawer.



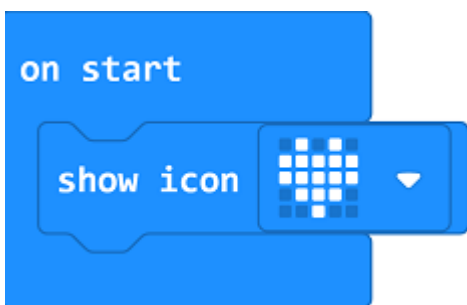
- We need to add a package for programming, click “Extensions” on the bottom of the drawer and search with “Cutebot” in the dialogue box to download it.



Note: If you met a tip indicating codebase will be deleted due to incompatibility, you may continue as the tips tell or build a new project.

## Step 2

- Show an icon when on start.



## Step 3

- Set the car to move at its full speed while button “up” being pressed on the remote controller; to reverse at its full speed while button “down” being pressed; to turn left at its full speed while button “Left” being pressed; to turn right at its full speed while button “Right” being pressed and to stop immediately while button “OK” being pressed.

on **Up ▼** button pressed

Go straight at full speed

on **Down ▼** button pressed

Reverse at full speed

on **Left ▼** button pressed

Turn left at full speed

on **Right ▼** button pressed

Turn right at full speed

on **OK ▼** button pressed

Stop car immediatly

## Link

Link: [https://makecode.microbit.org/\\_2z1a3APfPLT3](https://makecode.microbit.org/_2z1a3APfPLT3)

You can also download it directly below:

▶ Simulator

🧩 Blocks

JS JavaScript



🔗 Edit



---

## **16.5. Conclusion**

---

- Programme to use the IR Remote Control to give orders of moving forward, reversing, turning left/right and stopping to the car.

## **16.6. Exploration**

---

## **16.7. FAQ**

---

## **16.8. Relevant File**

---

## 17. Case15: Seeking the Light

### 17.1. Purpose

---

- Programme to make the Cutebot seek the light source automatically.

### 17.2. Materials

---

- 1 x Cutebot Kit

### 17.3. Software Platform

---

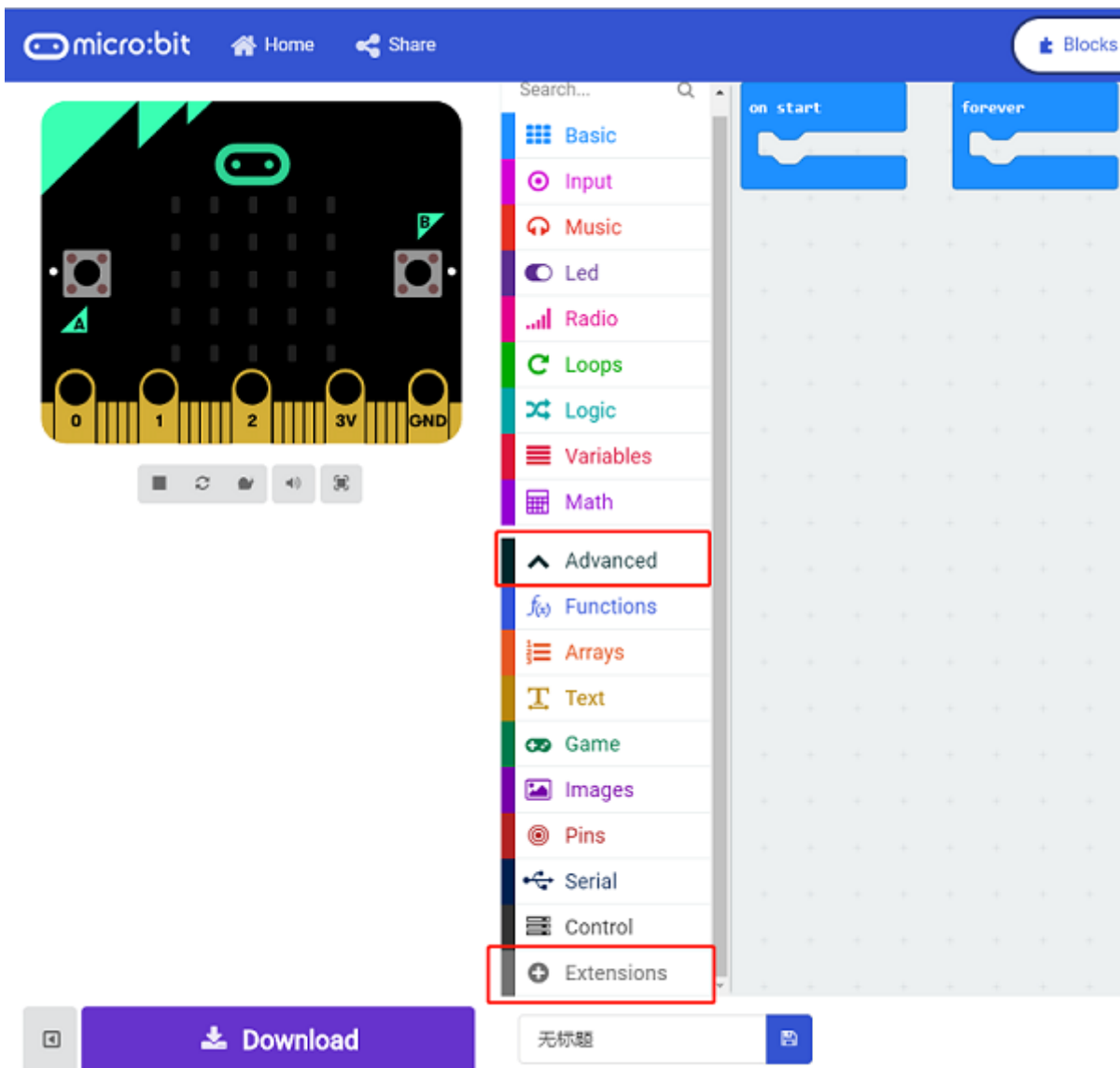
MicroSoft makecode

### 17.4. Programming

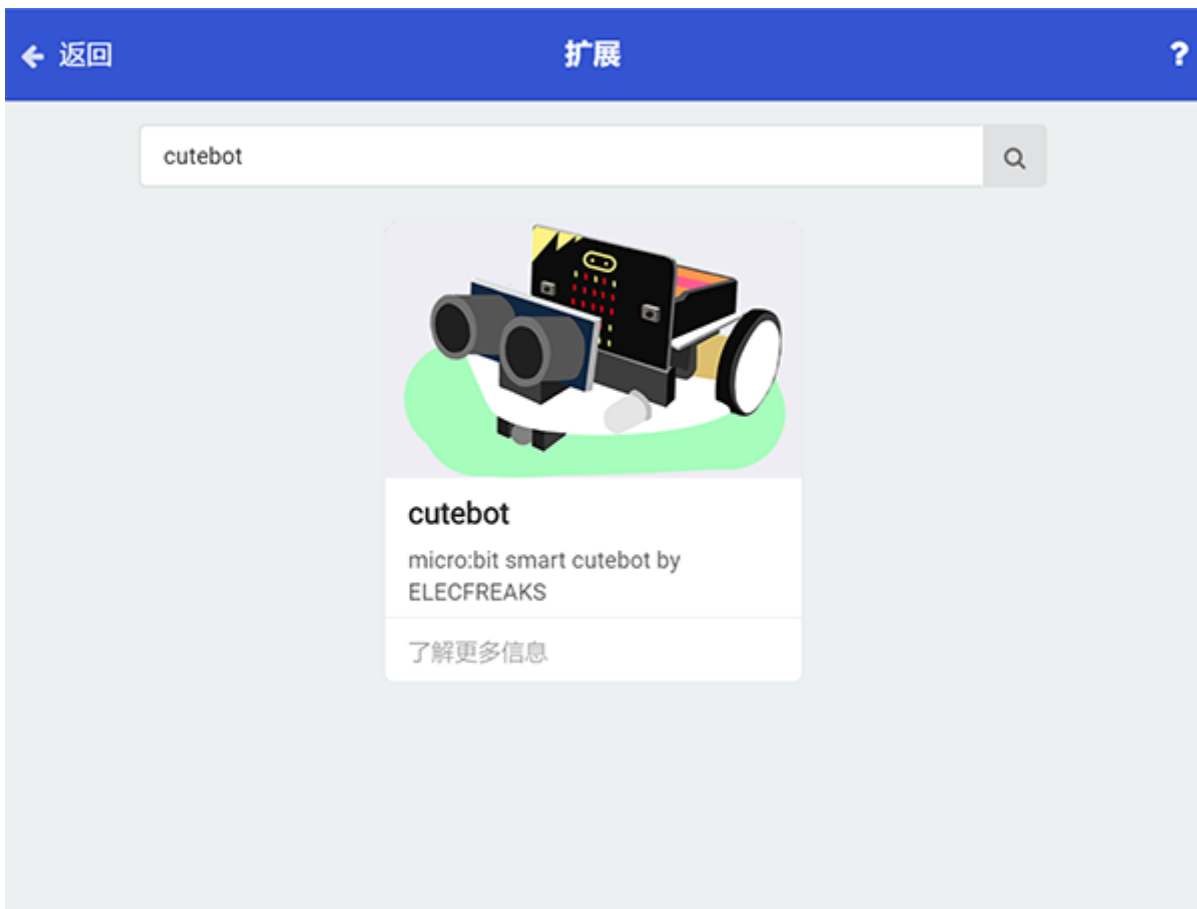
---

#### Step 1

- Click “Advanced” in the MakeCode drawer to see more choices.



- We need to add a package for programming, click “Extensions” on the bottom of the drawer and search with “Cutebot” in the dialogue box to download it.



Note: If you met a tip indicating the codebase will be deleted due to incompatibility, you may continue as the tips tell or build a new project.

## Step 2

- Judge the luminous intensity with the block “light level “ in “forever” ; if the value is below the setting point, set the Cutebot turn left at its full speed; Or it moves forward at its full speed.



**Link**

Link: [https://makecode.microbit.org/\\_UatK2a6cgc7u](https://makecode.microbit.org/_UatK2a6cgc7u)

You can also download it directly below:

 Simulator  Blocks  JavaScript   Edit

---

## 17.5. Result

---

- The Cutebot spins if there is no light being detected or it drives forward to it at its full speed.

## 17.6. Exploration

---

## 17.7. FAQ

---

## 17.8. Relevant Files

---

## 18.1. Cutebot & AI Lens Line-tracking Kit

### Purpose

---

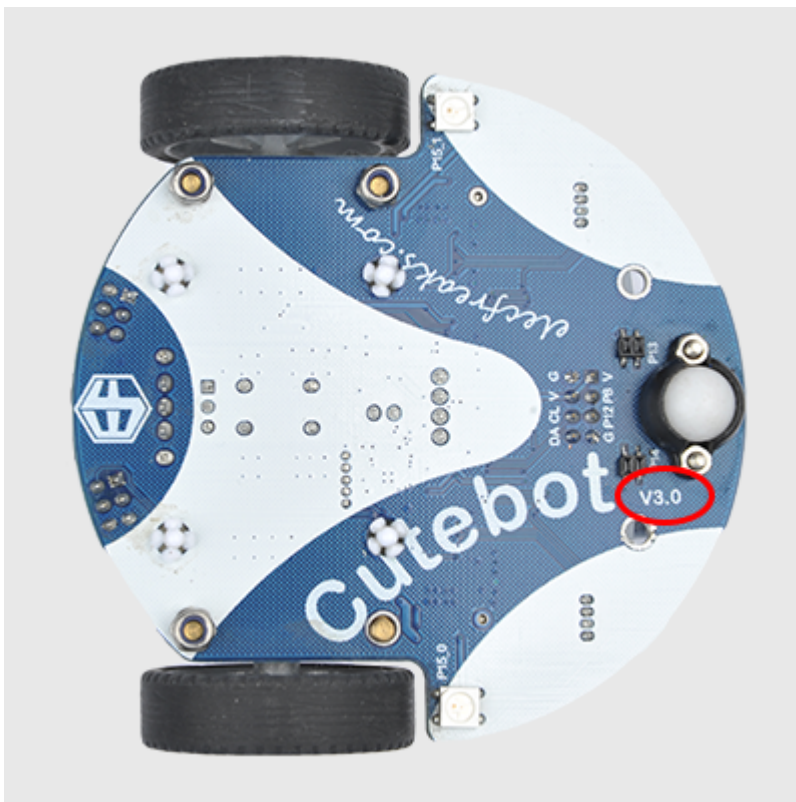
- Use the Cutebot and the AI lens to achieve the line tracking function.

### Materials required

---

- 1 × Cutebot V3.0
- 1 × Cutebot lithium battery pack
- 1 × AI Lens Kit

*Note: The AI Lens kit works with Cutebot V3.0 only(You can see the version number printed on the baseboard).*



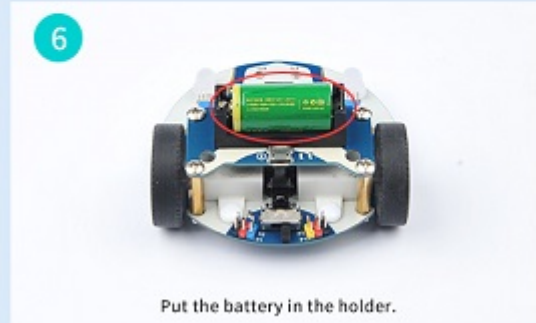
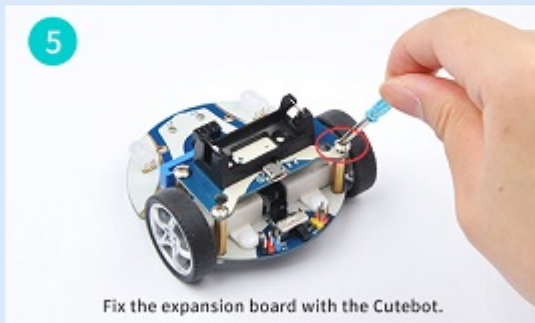
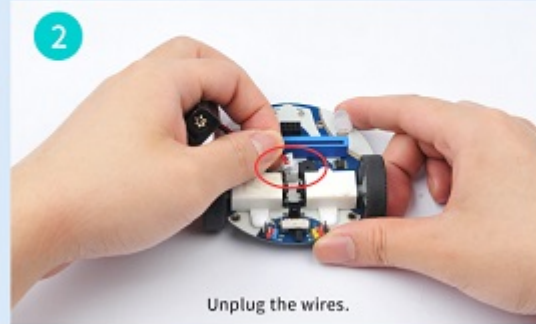
### Connections:

---

### Steps to install the lithium battery pack:

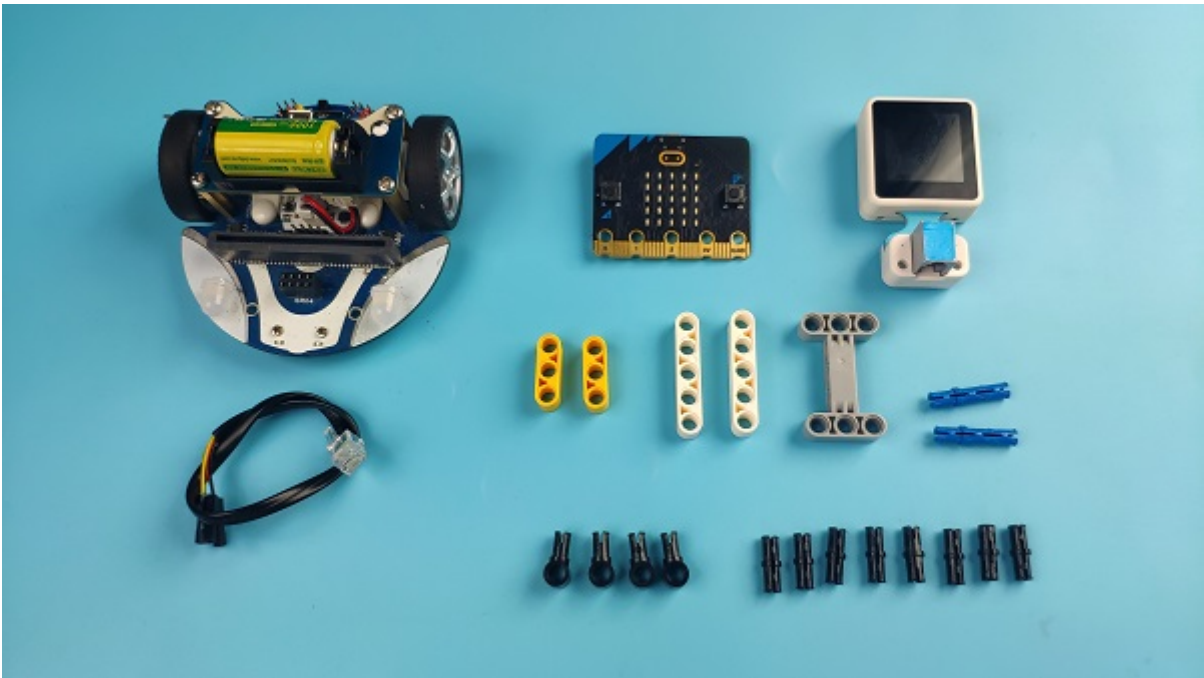
## Assembly steps for the battery pack

Follow the below assembly figures



Assembly steps for bricks:

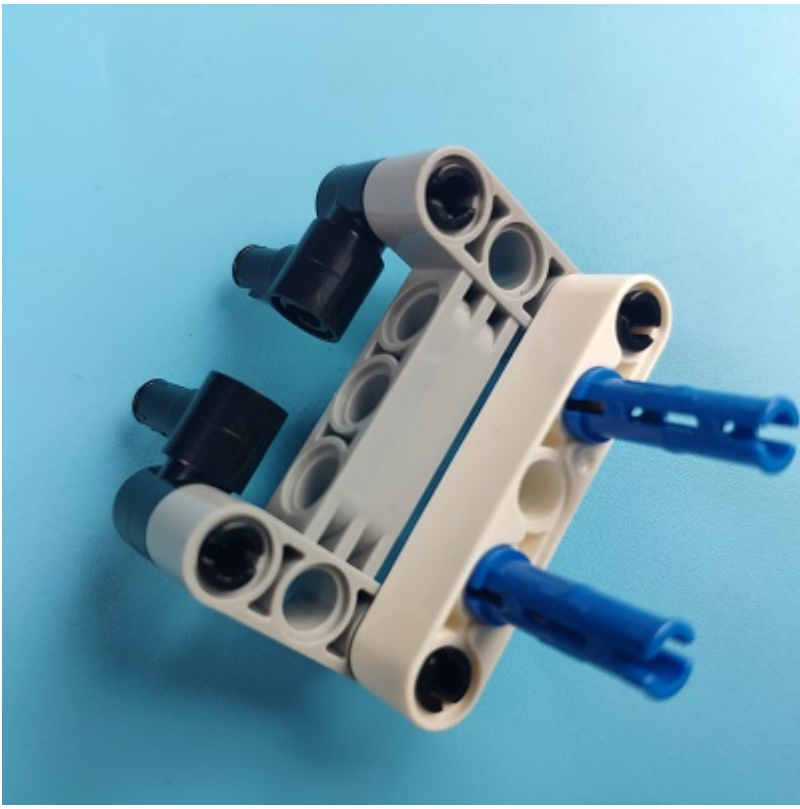
Parts list:

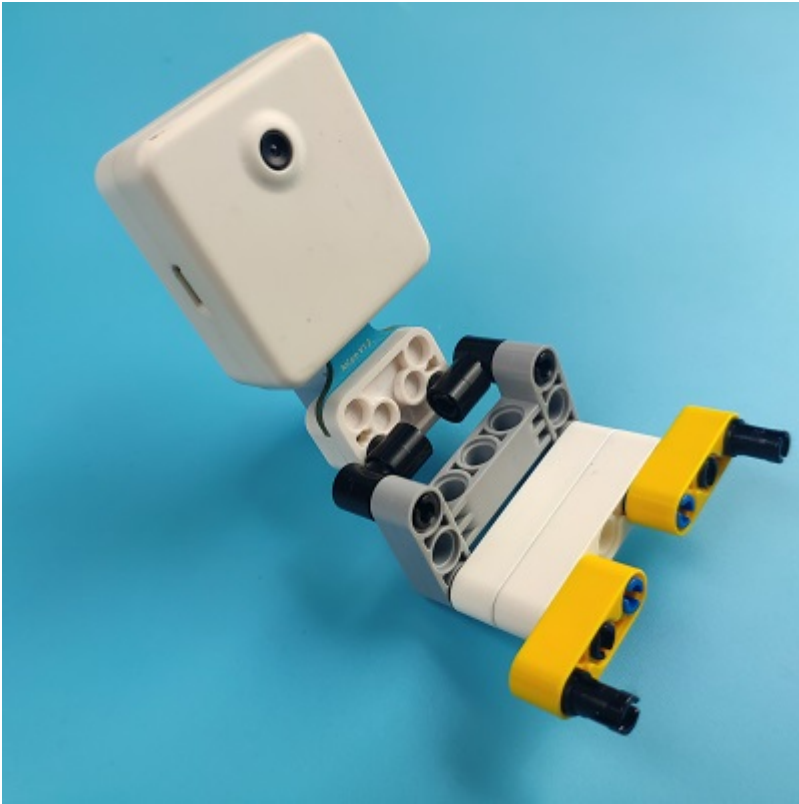
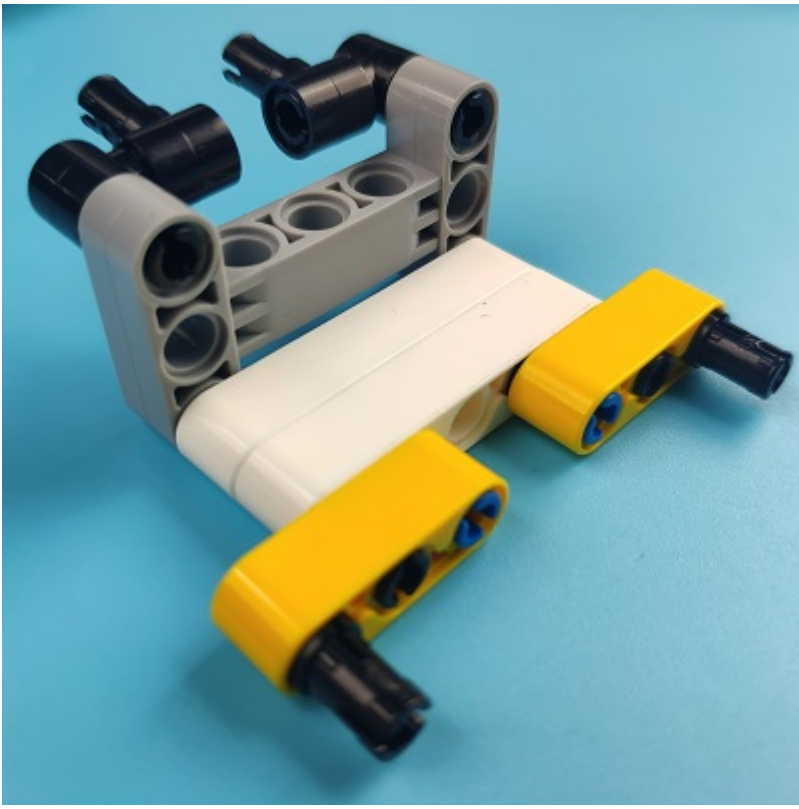


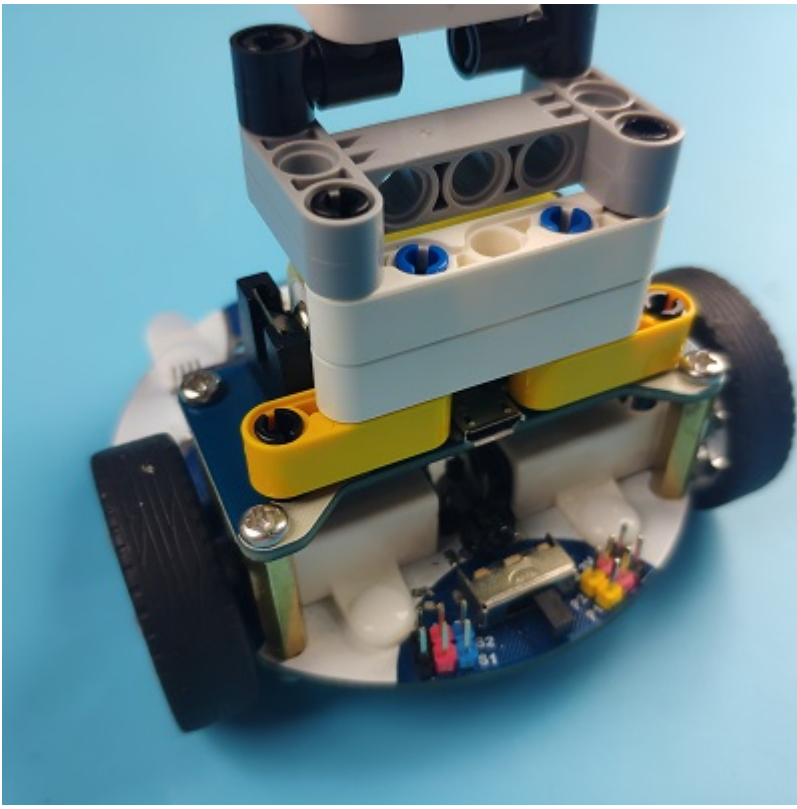
Steps of build-up:





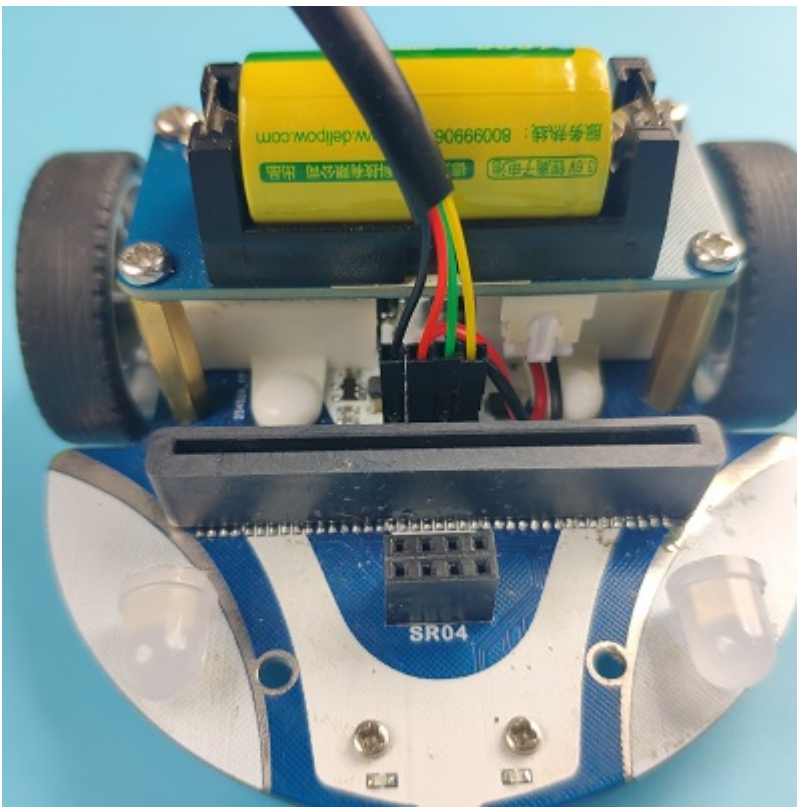






### **Connections of the AI Lens:**

Connect the RJ11 cable with the AI Lens and the other end in Dupont connection to the circled place in the below picture (make sure you connect to the right connections).



*Tips: the bricks holder here is flexible to be adjusted, we may manually adjust the angles of the AI lens to meet the requirements of the functions that you want to achieve.*

## **Software Platform:**

---

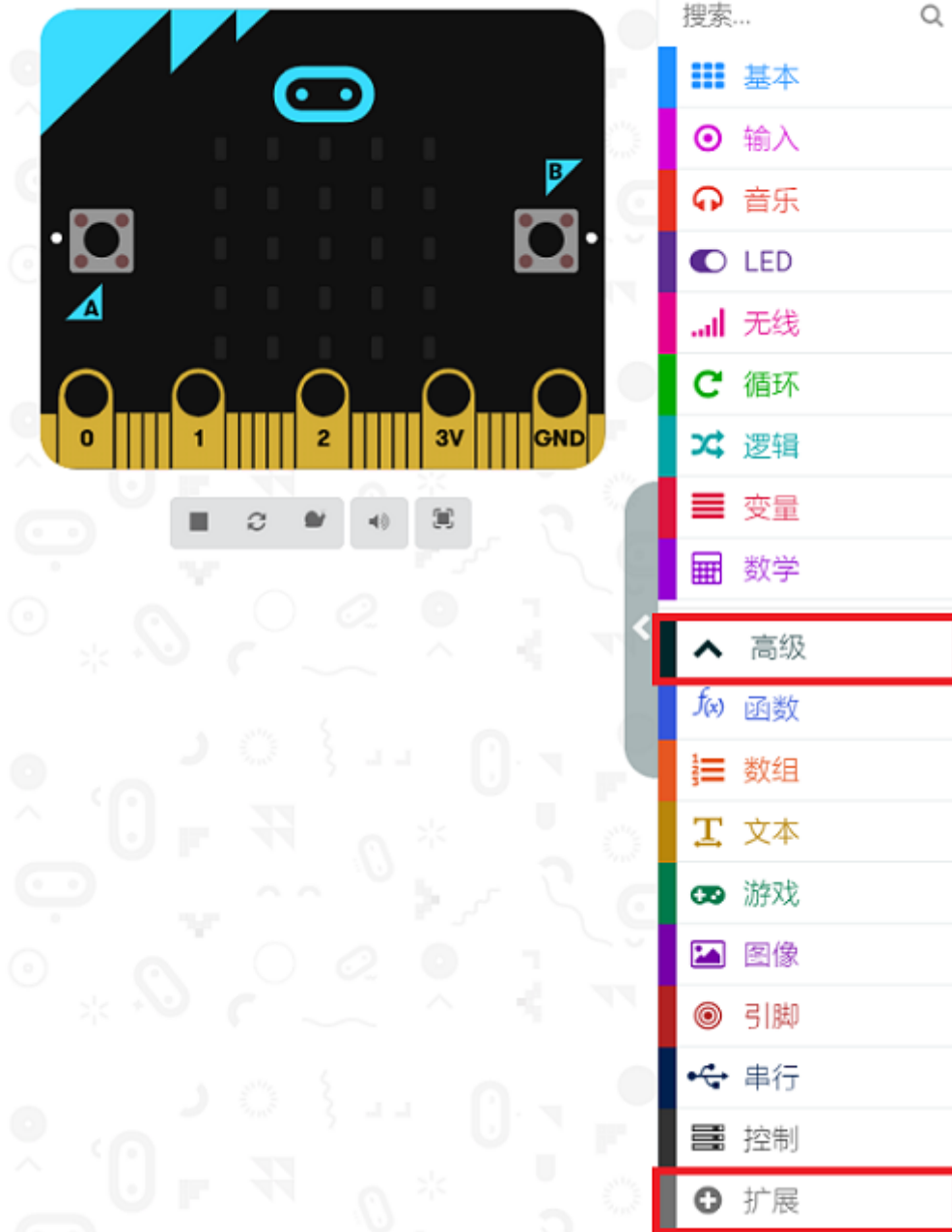
MicroSoft MakeCode

## **Programming**

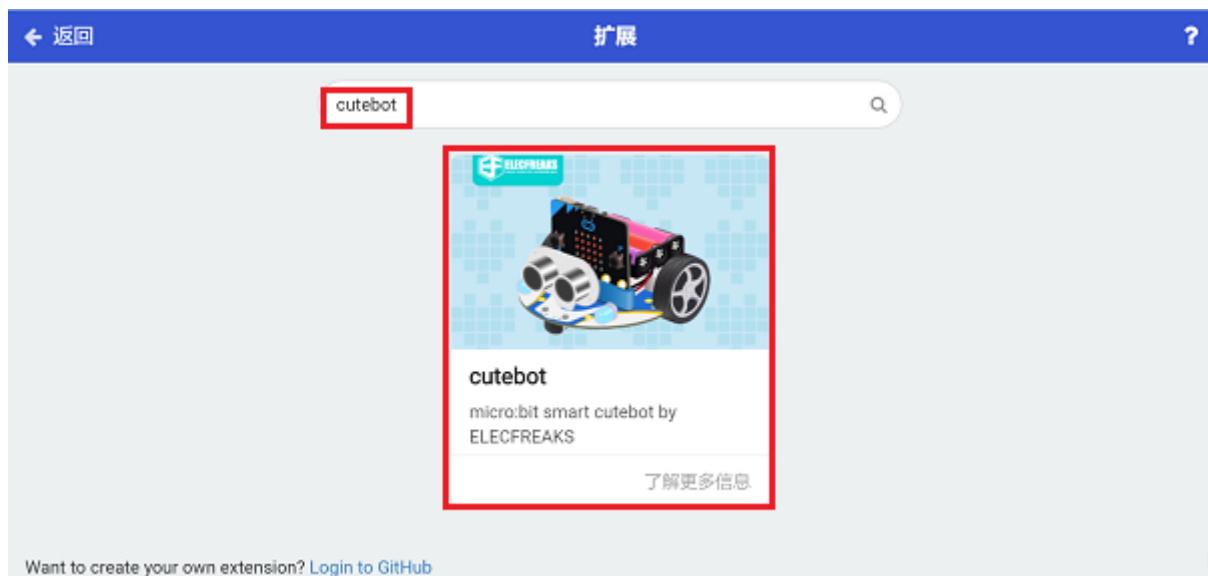
---

### **Step 1**

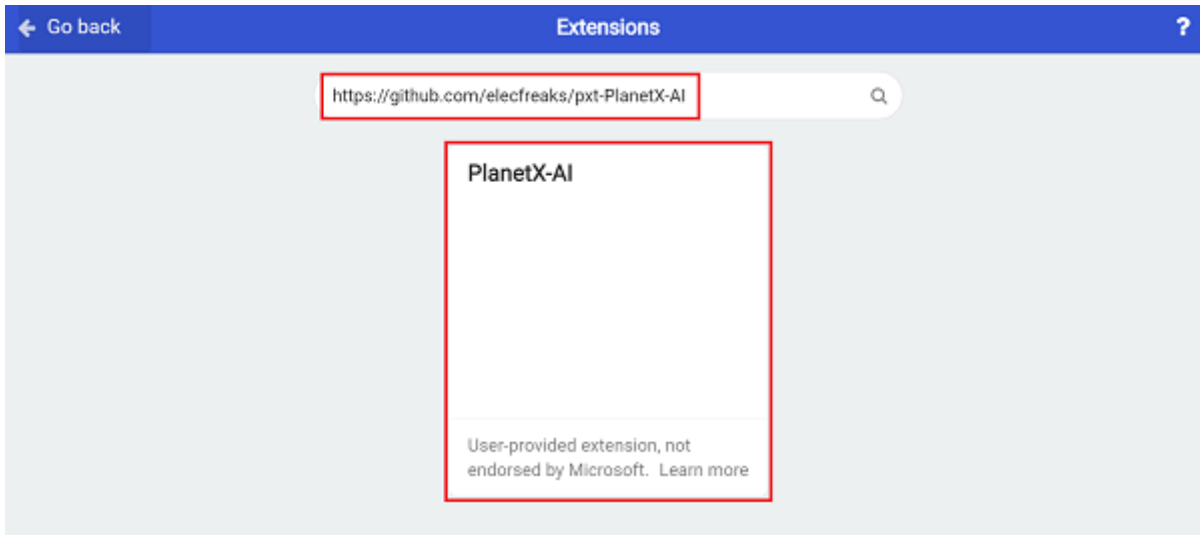
- Click “Advanced” in the drawer to see more choices.



- We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “cutebot” in the dialogue box to download it.



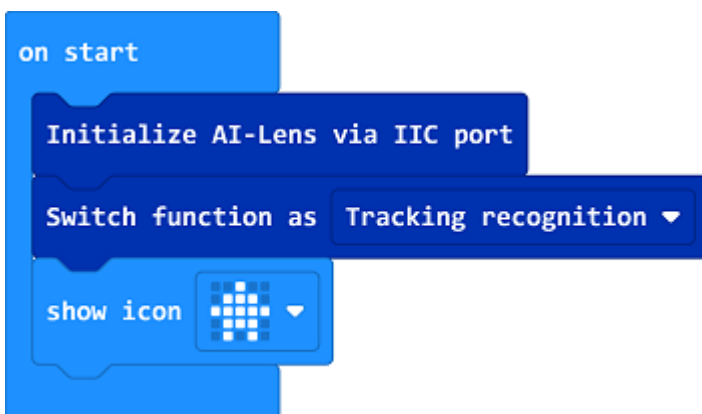
- We need to add a package for programming the AI lens kit. Click “Extensions” in the bottom of the drawer and search with “<https://github.com/electfreaks/pxt-PlanetX-AI>” in the dialogue box to download it.



*Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.*

## Step 2

- In the “on start” brick, initialize the AI lens and switch the function to the line tracking mode, set the micro:bit to display the appointed icon.



- In the “forever” brick, set to get one image form the AI lens and judge the deviation direction of the line on the image. If it deviates to the left side, it means the car deviates to the right, we should set the speed of the left wheel at the speed of 10% and the right at 40% to make the car turn left and go to the right way; if the line deviates to the right side, it means the car deviates to the left side, now we should set the speed of the right wheel at the speed of 10% and the left at 40% to make the car turn right and go to the right way; or we may set the speed of both wheels at 20% and the car moves forward with the line.



```
forever
  Get one image from AI-Lens
  if Image contains line's direction towards Left then
    Set left wheel speed 10 % right wheel speed 40 %
  else if Image contains line's direction towards Right then
    Set left wheel speed 40 % right wheel speed 10 %
  else
    Set left wheel speed 20 % right wheel speed 20 %
```

## Code

```
on start
  Initialize AI-Lens via IIC port
  Switch function as Tracking recognition
  show icon [grid icon]
forever
  Get one image from AI-Lens
  if Image contains line's direction towards Left then
    Set left wheel speed 10 % right wheel speed 40 %
  else if Image contains line's direction towards Right then
    Set left wheel speed 40 % right wheel speed 10 %
  else
    Set left wheel speed 20 % right wheel speed 20 %
```

Link: [https://makecode.microbit.org/\\_hcy8YeM87AVd](https://makecode.microbit.org/_hcy8YeM87AVd)

You may also download it directly below:

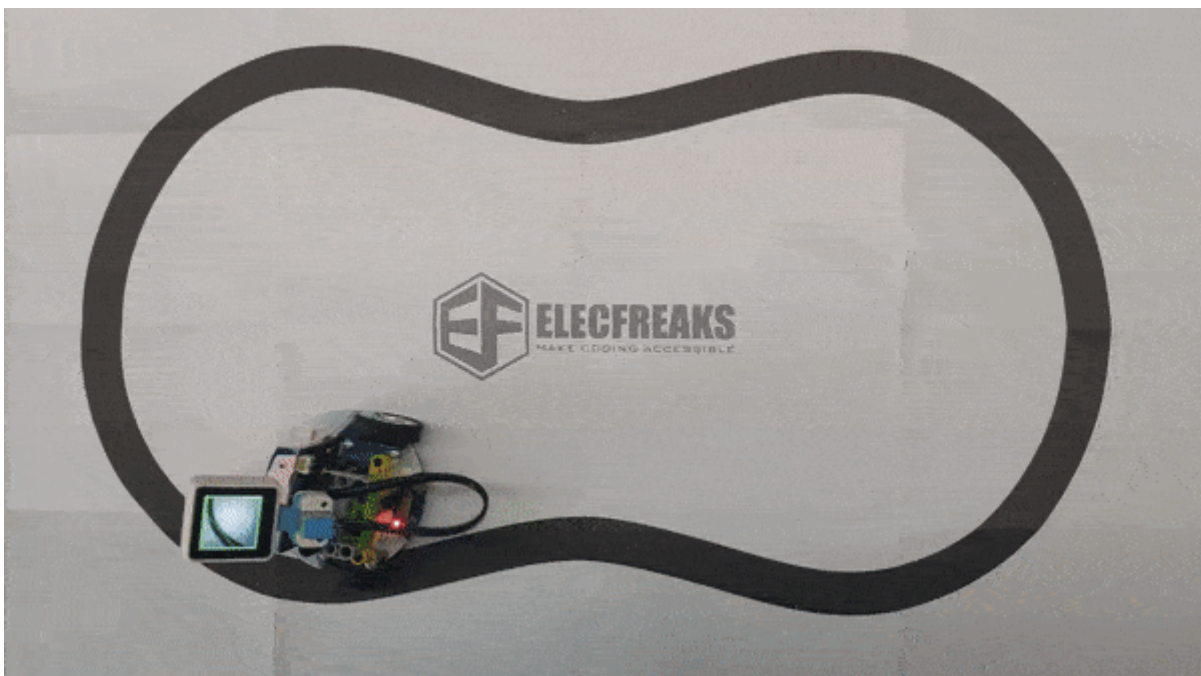
▶ Simulator    🧩 Blocks    JS JavaScript    ▼    ↗ Edit

---

## Result

---

- The Cutebot car moves along with the black line.





## **Exploration**

---

## **FAQ**

---

## **Relevant Files**

---

## 18.2. Cutebot & AI Lens Signpost Car

### Purpose

---

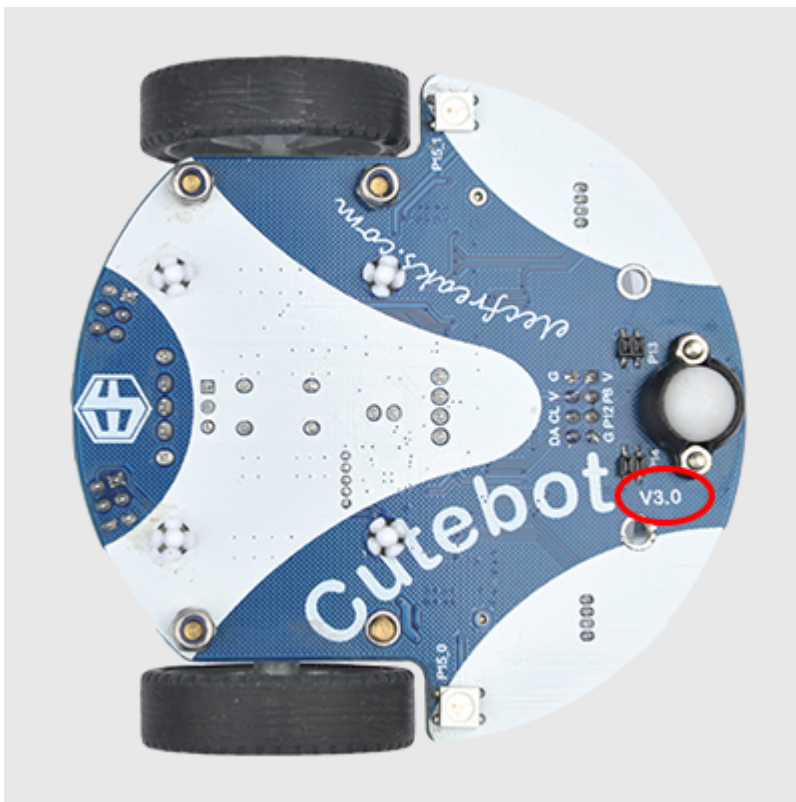
- Control the Cutebot with the AI Lens by recognizing the signpost cards.

### Materials required

---

- 1 × Cutebot V3.0
- 1 × Cutebot lithium battery pack
- 1 × AI Lens Kit

*Note: The AI Lens kit works with Cutebot V3.0 only(You can see the version number printed on the baseboard).*



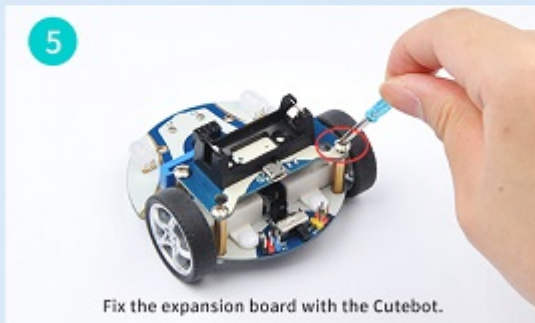
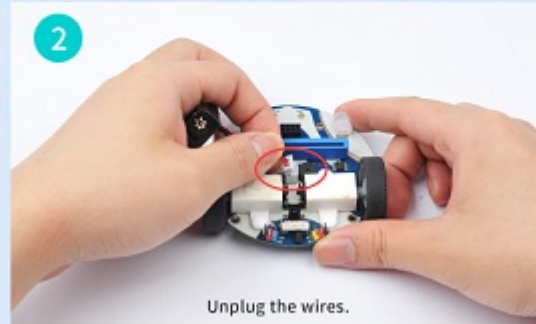
### Connections:

---

### Steps to install the lithium battery pack:

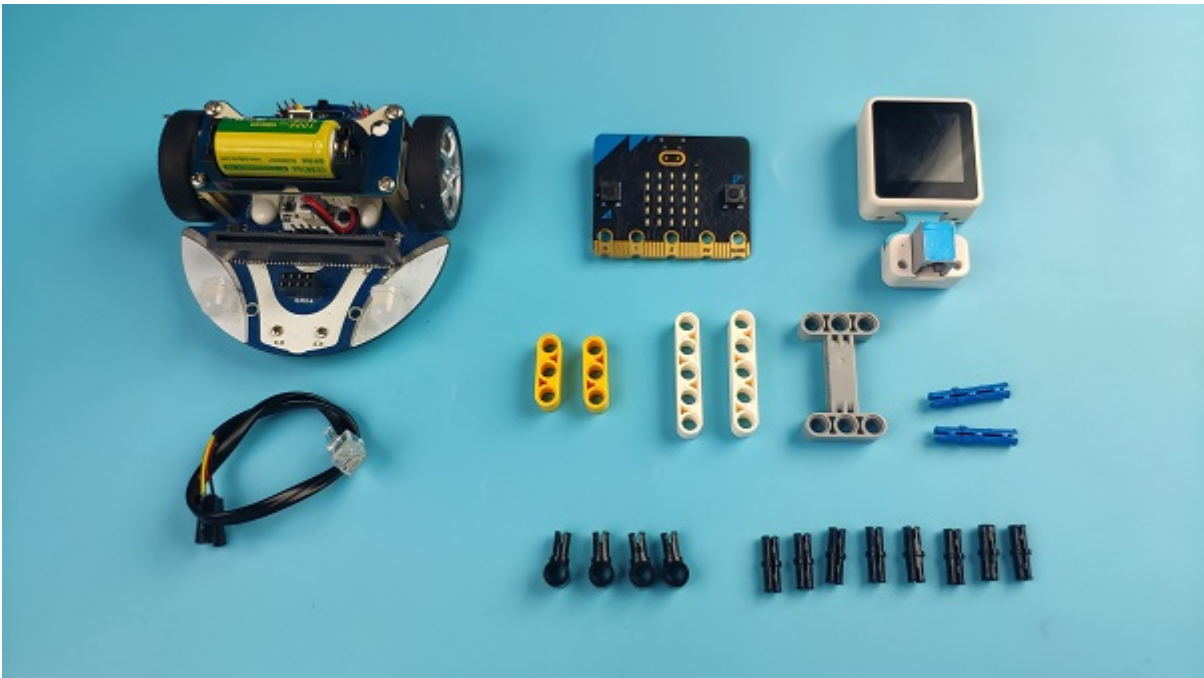
## Assembly steps for the battery pack

Follow the below assembly figures



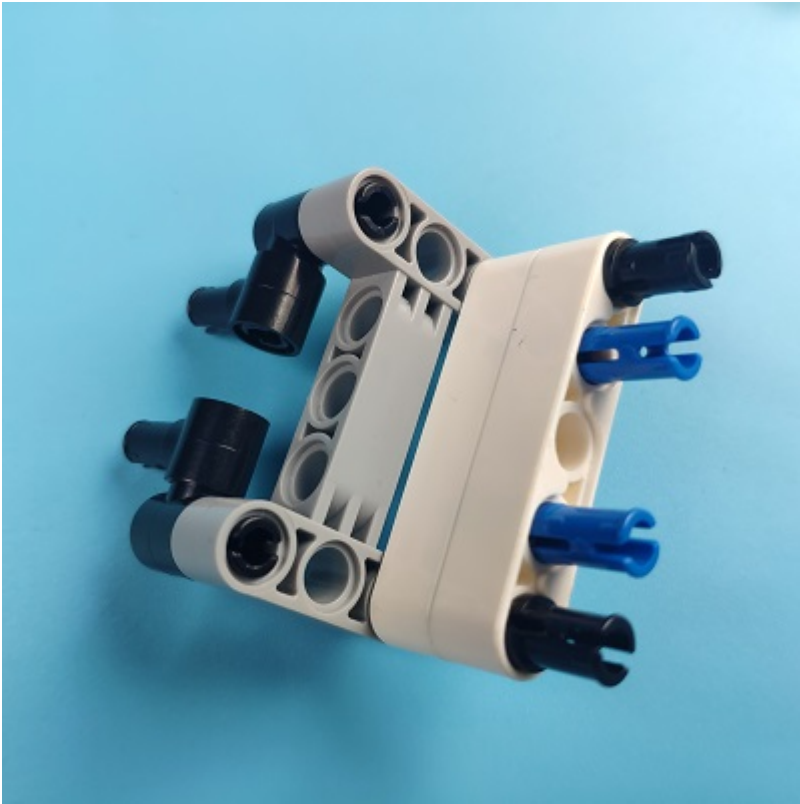
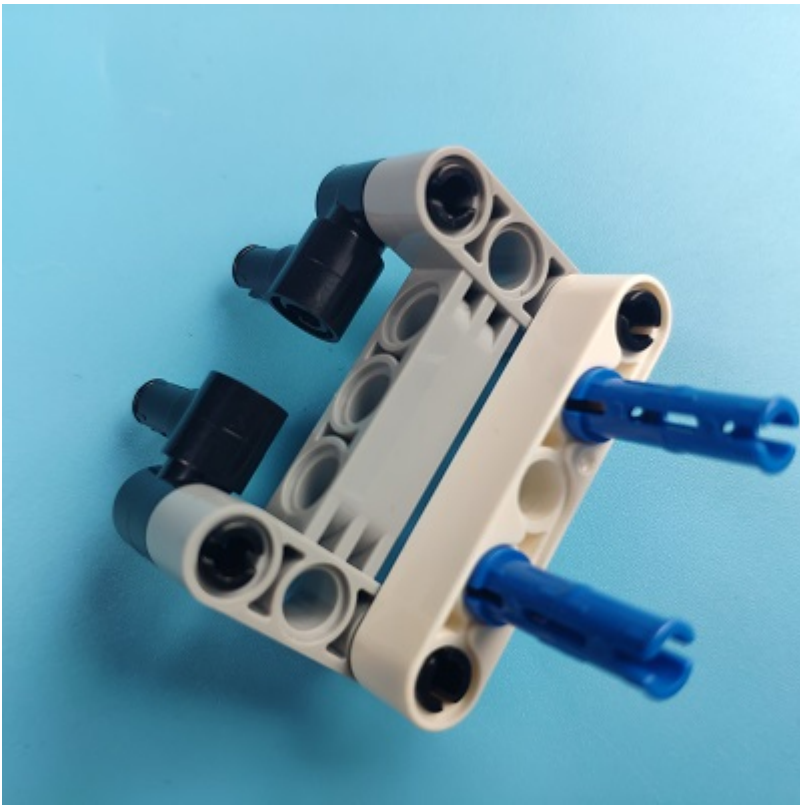
Assembly steps for bricks:

Parts list:

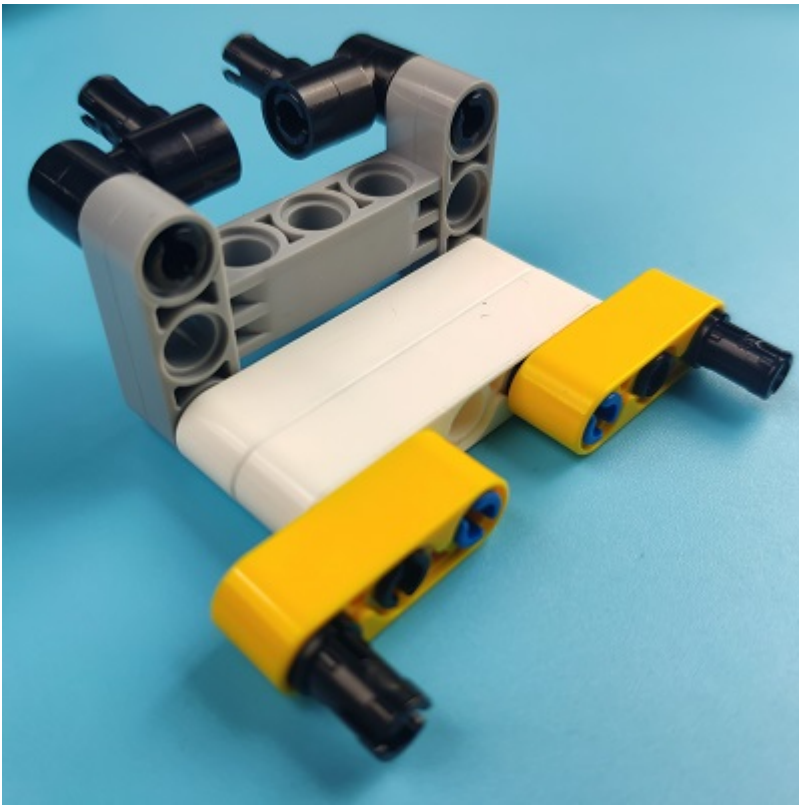


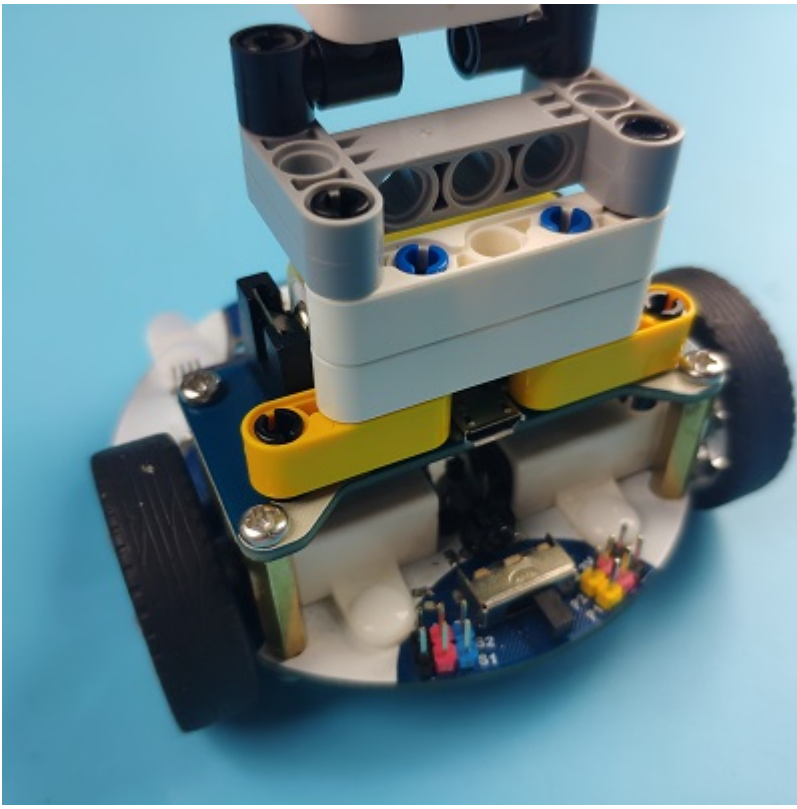
Steps of build-up:





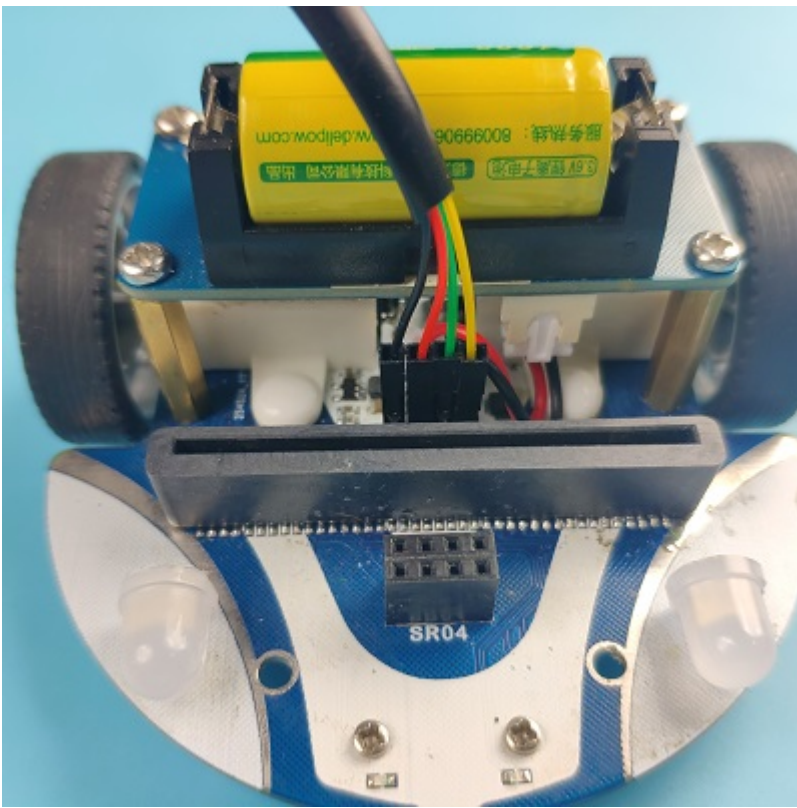






### **Connections of the AI Lens:**

Connect the RJ11 cable with the AI Lens and the other end in Dupont connection to the circled place in the below picture (make sure you connect to the right connections).



*Tips: the bricks holder here is flexible to be adjusted, we may manually adjust the angles of the AI lens to meet the requirements of the functions that you want to achieve.*

## Software Platform:

---

MicroSoft MakeCode

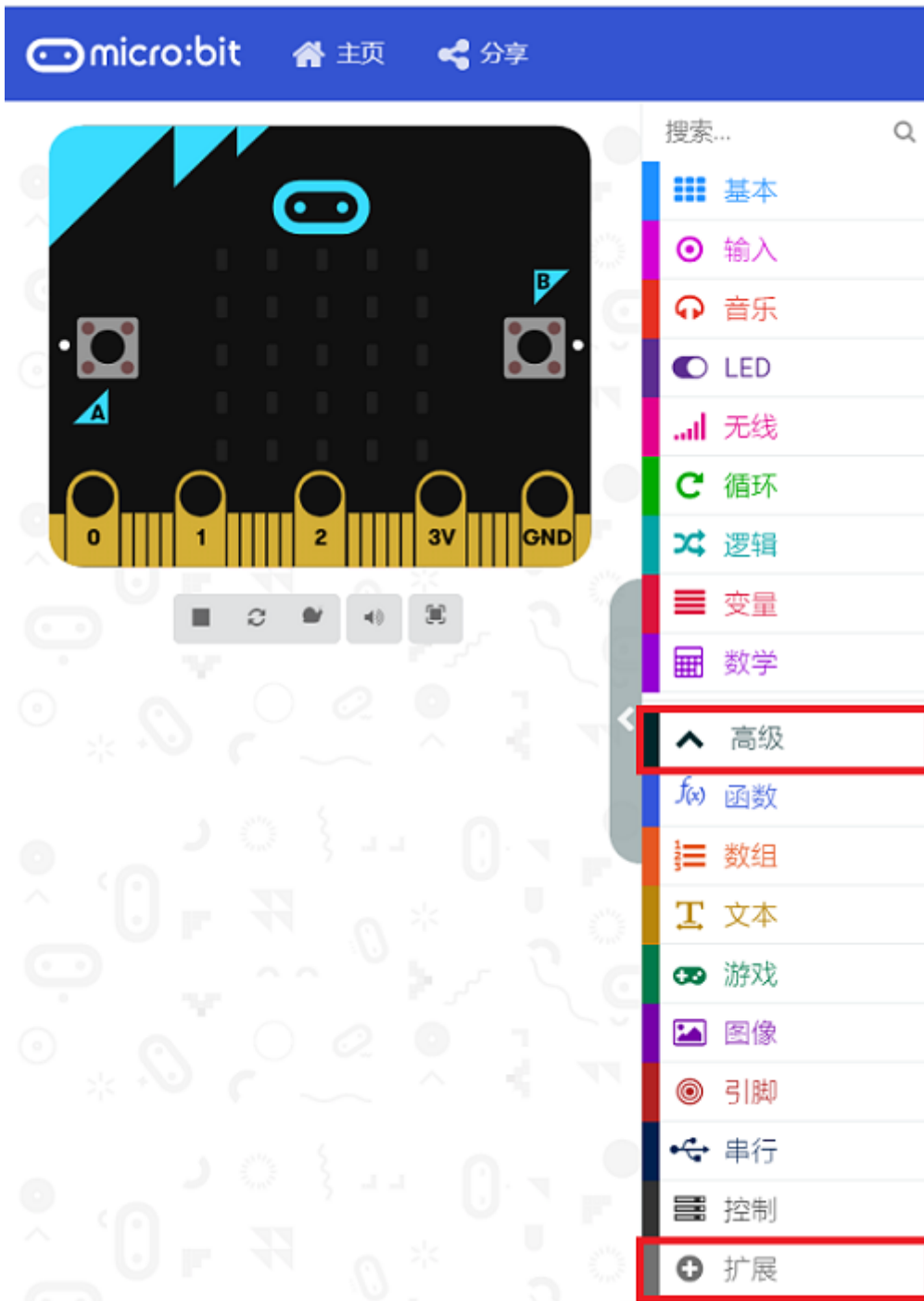
## Programming

---

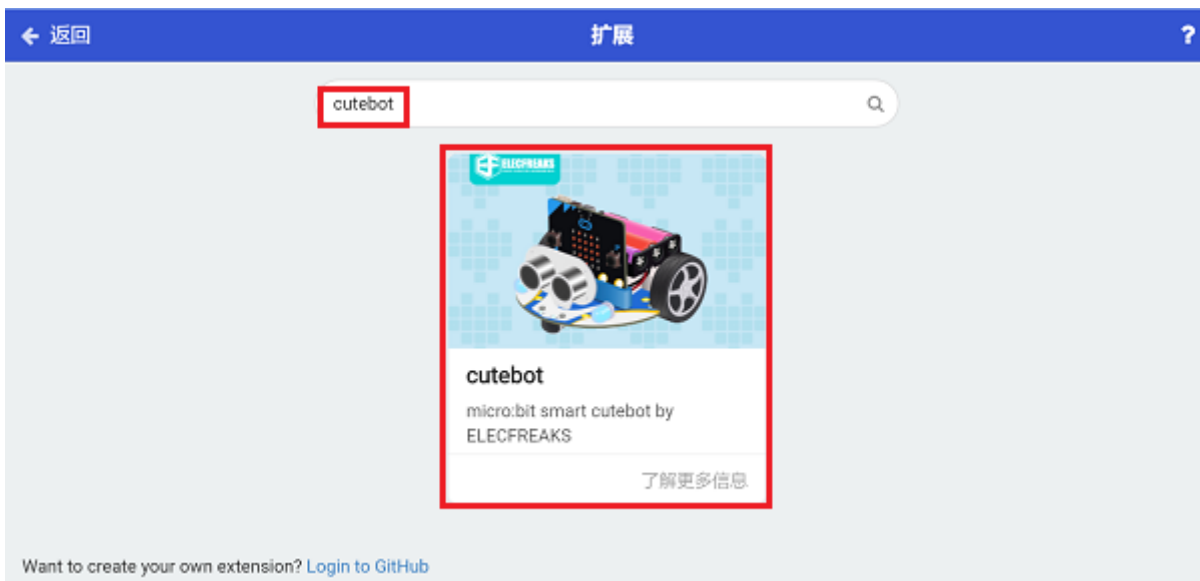
### Step 1

- Click “Advanced” in the drawer to see more choices.

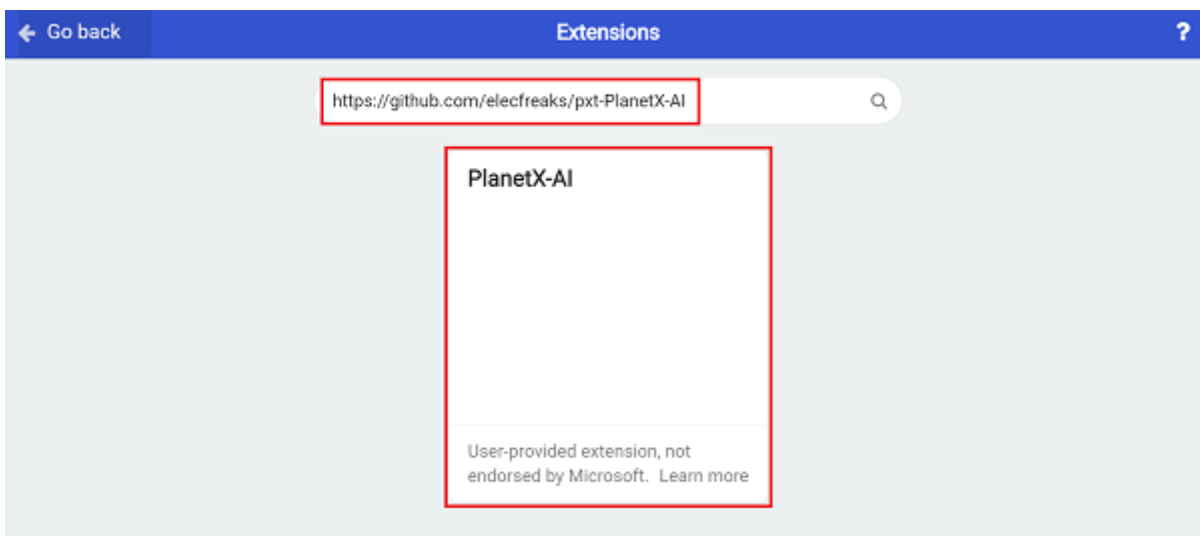




- We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “cutebot” in the dialogue box to download it.



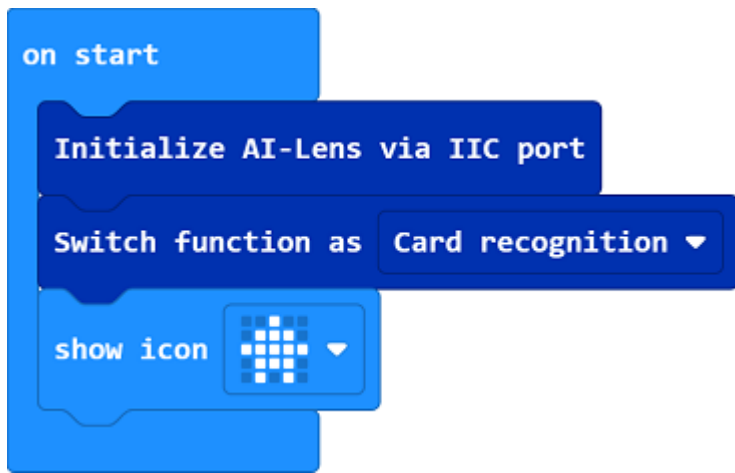
We need to add a package for programming the AI lens kit. Click “Extensions” in the bottom of the drawer and search with “<https://github.com/electfreaks/pxt-PlanetX-AI>” in the dialogue box to download it.



*Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.*

## Step 2

- In the “on start” brick, initialize the AI lens and switch the function to the cards recognition mode, set the micro:bit to display the appointed icon.



- In the “forever” brick, set to get one image form the AI lens and judge the cards icon on the image. If it reconizes the “forward” icon, we set the speed of the left wheel at the speed of 50% and the right at the same to make the car go forward; if it recognizes a “turn-left” icon, we set the car to turn left for 0.5s at the speed of 30% and then move forward, note the AI Lens has three buffers and we need get three images to clear the buffer after the execution. If it recognizes a “turn-right” icon, we set the car to turn right for 0.5s at the speed of 30% and then move forward. Note the AI Lens has three buffers and we need get three images to clear the buffer after the execution. If it recognizes the “stop” icon, the car stops moving accordingly.

```
forever
  Get one image from AI-Lens
  if Image contains traffic card(s): Forward then
    Set left wheel speed 50 % right wheel speed 50 %
  +
  if Image contains traffic card(s): Turn left then
    Go Left at speed 30 % for 0.5 seconds
    Set left wheel speed 50 % right wheel speed 50 %
    repeat 3 times
    do Get one image from AI-Lens
  +
  if Image contains traffic card(s): Turn right then
    Go Right at speed 30 % for 0.5 seconds
    Set left wheel speed 50 % right wheel speed 50 %
    repeat 3 times
    do Get one image from AI-Lens
  +
  if Image contains traffic card(s): Stop then
    Stop car immediatly
  +
```

## Code

```
on start
```

Initialize AI-Lens via IIC port

Switch function as

show icon 

forever

Get one image from AI-Lens

if  then

Set left wheel speed  % right wheel speed  %



if  then

Go  at speed  % for  seconds

Set left wheel speed  % right wheel speed  %

repeat  times

do



if  then

Go  at speed  % for  seconds

Set left wheel speed  % right wheel speed  %

repeat  times

do



if  then

Stop car immediatly



Link: [https://makecode.microbit.org/\\_1v3bWhet40hr](https://makecode.microbit.org/_1v3bWhet40hr)

You may also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼    [↗ Edit](#)

---

## Result

---

If the AI Lens recognizes the “forward” card, the car moves forward; if it recognizes the “turn-left” card, the car turns left and then goes forward; if it recognizes the “turn-right” card, the car turns right and then goes forward; if it recognizes the “stop” card, the car stops moving.

## Exploration

---

## FAQ

---

## Relevant Files

---



## 18.3. Cutebot & AI Lens Discoloring Lights

### Purpose

---

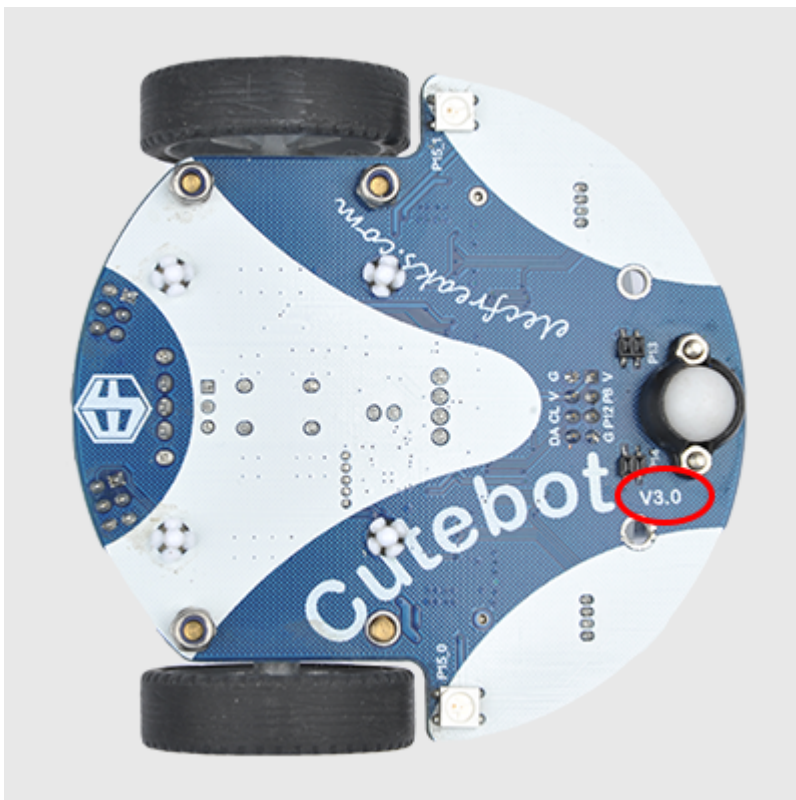
- Change the color of the lights via the color of the cards with the Smart AI Lens.

### Materials required

---

- 1 × Cutebot V3.0
- 1 × Cutebot lithium battery pack
- 1 × AI Lens Kit

*Note: The AI Lens kit works with Cutebot V3.0 only(You can see the version number printed on the baseboard).*



### Connections:

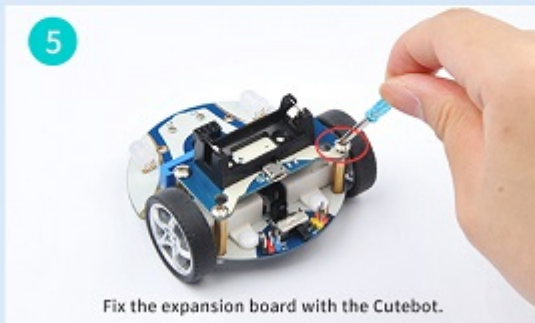
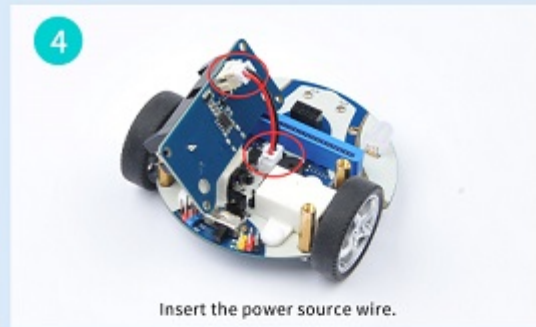
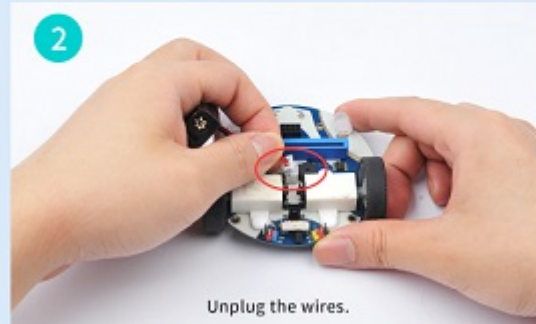
---

### Steps to install the lithium battery pack:



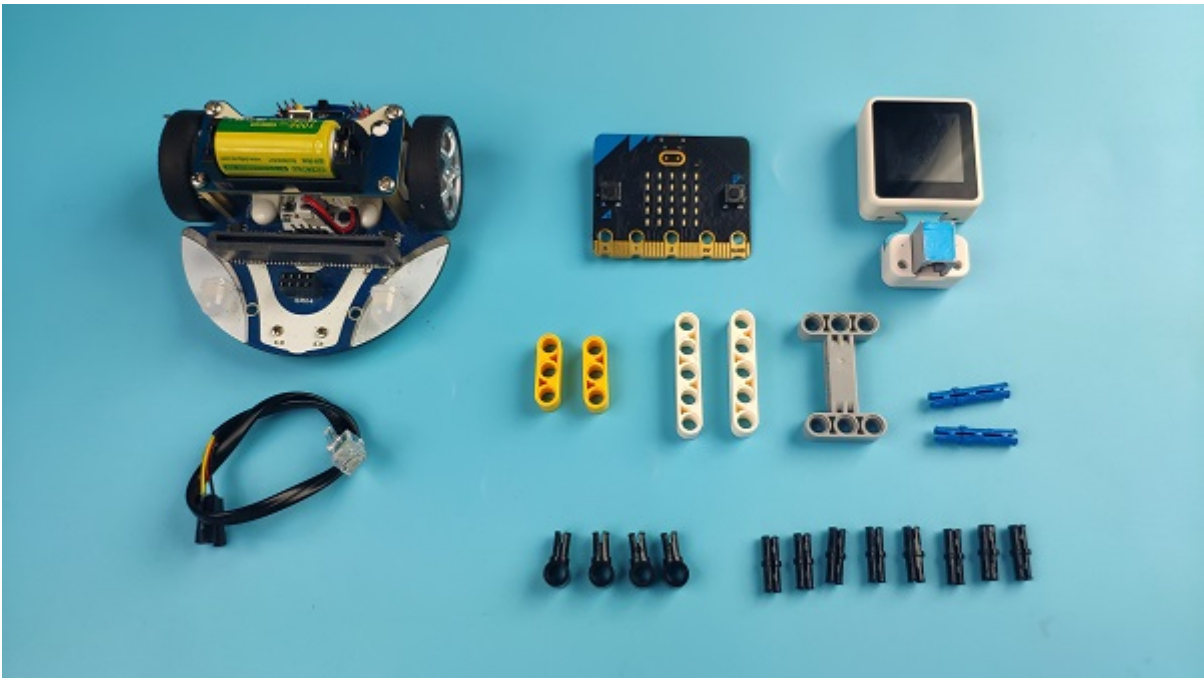
## Assembly steps for the battery pack

Follow the below assembly figures



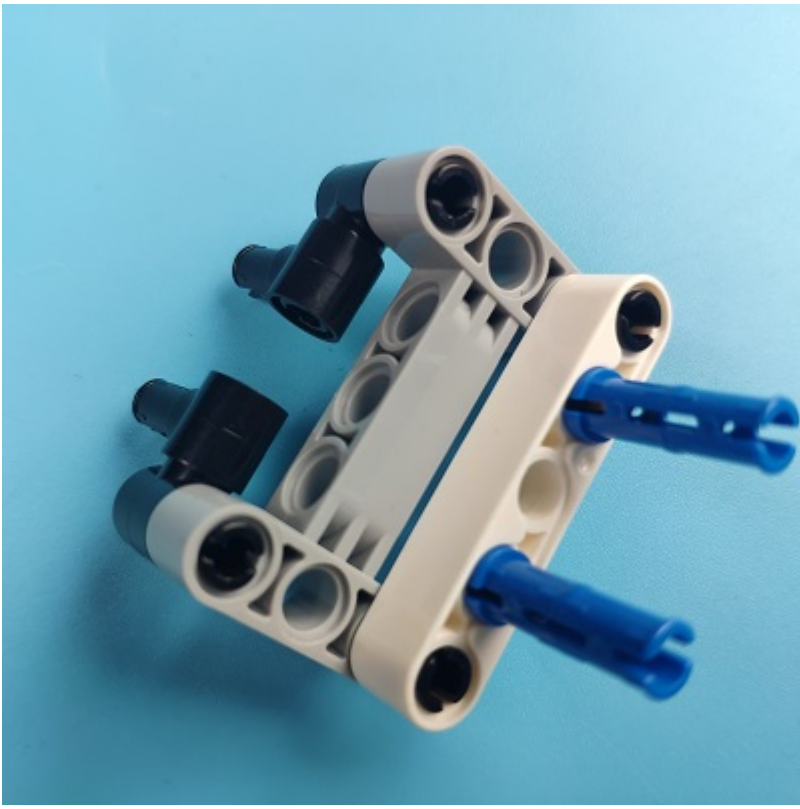
Assembly steps for bricks:

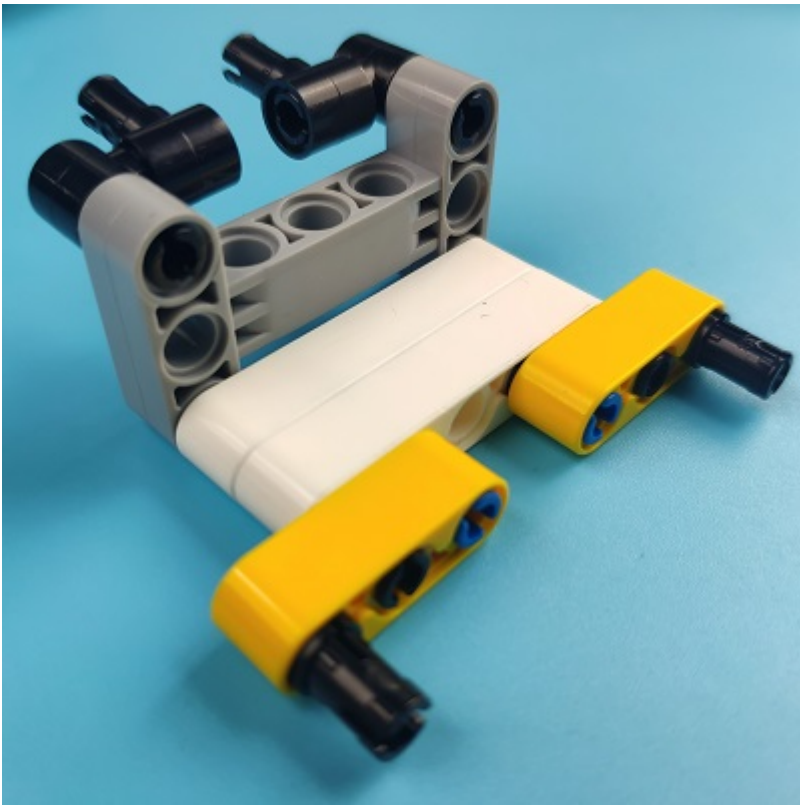
Parts list:



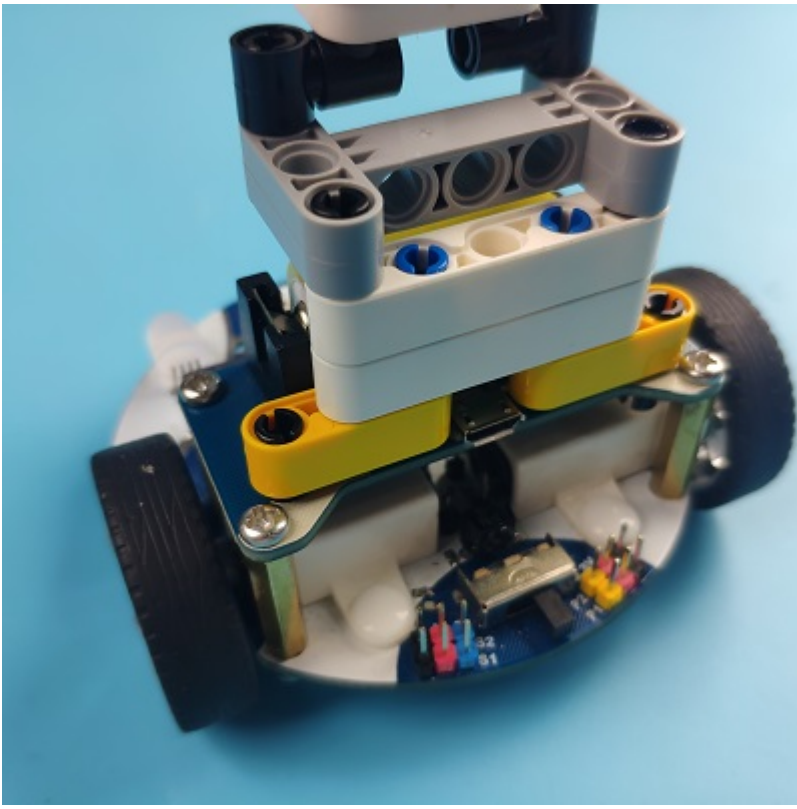
Steps of build-up:





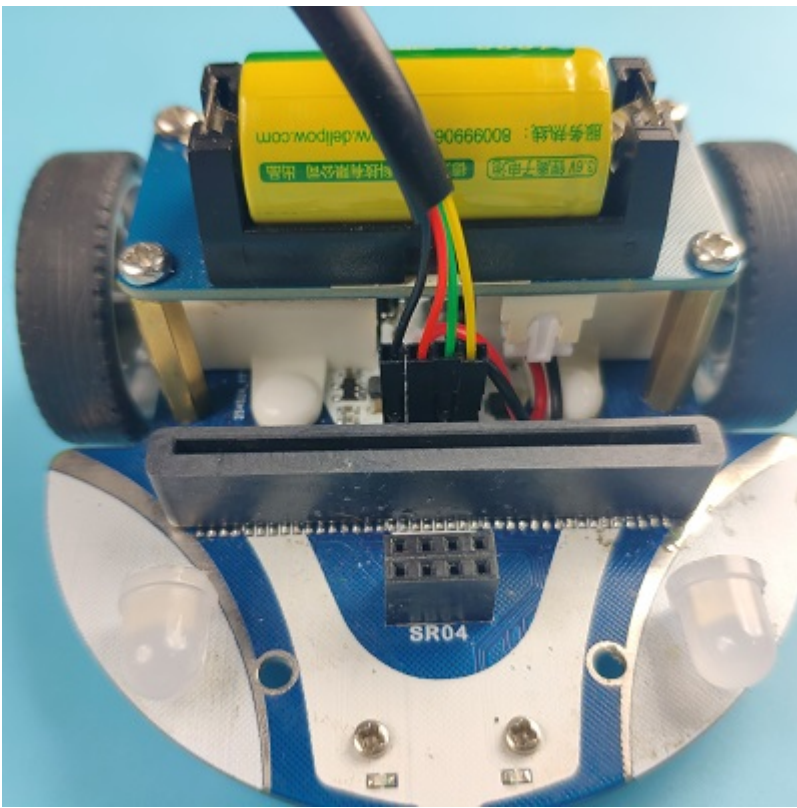






### **Connections of the AI Lens:**

Connect the RJ11 cable with the AI Lens and the other end in Dupont connection to the circled place in the below picture (make sure you connect to the right connections).



*Tips: the bricks holder here is flexible to be adjusted, we may manually adjust the angles of the AI lens to meet the requirements of the functions that you want to achieve.*

## **Software Platform:**

---

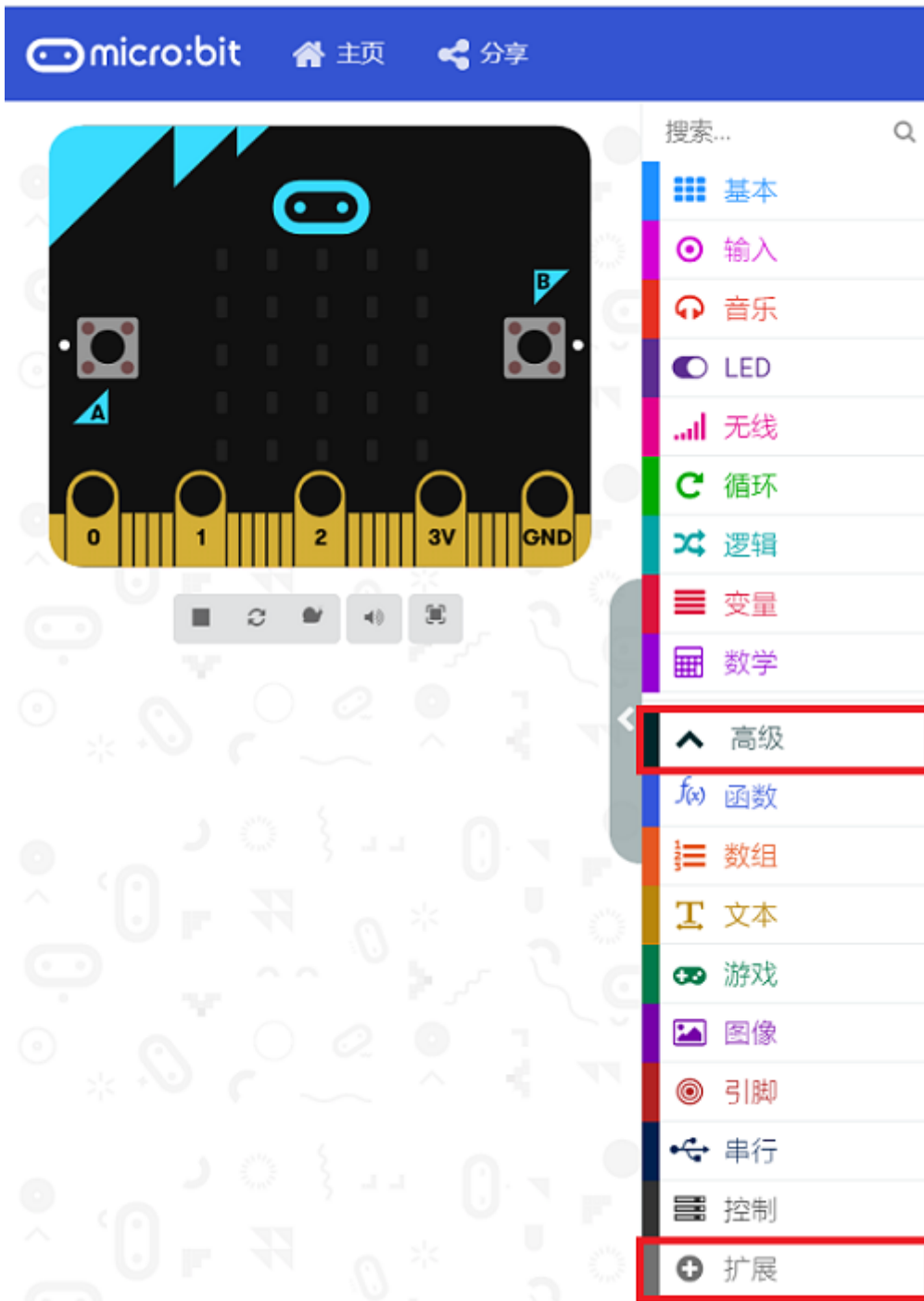
MicroSoft MakeCode

## **Programming**

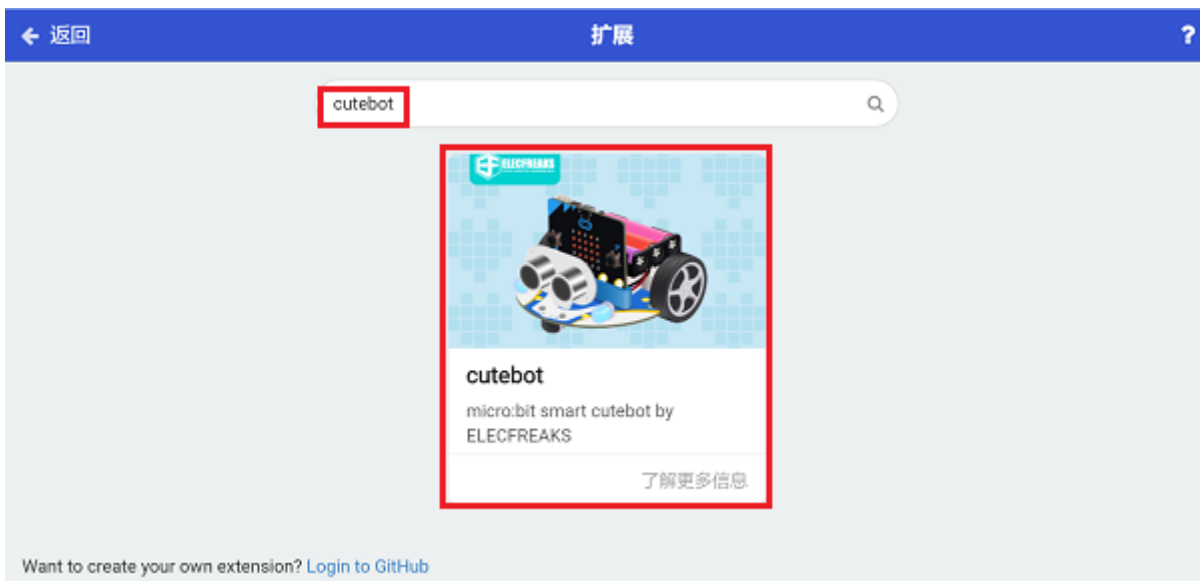
---

### **Step 1**

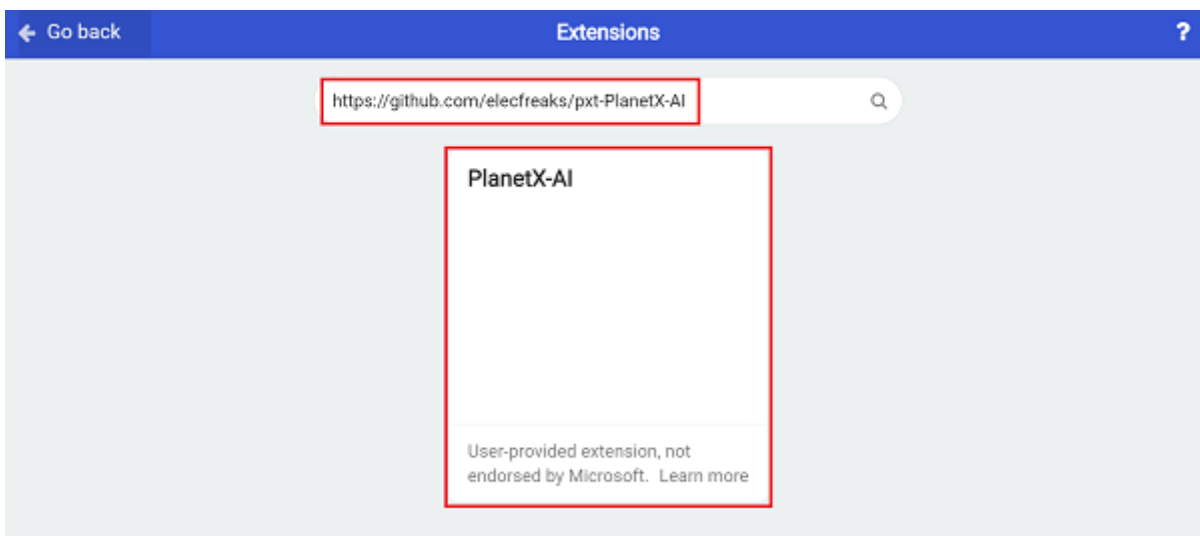
Click "Advanced" in the drawer to see more choices.



- We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “cutebot” in the dialogue box to download it.



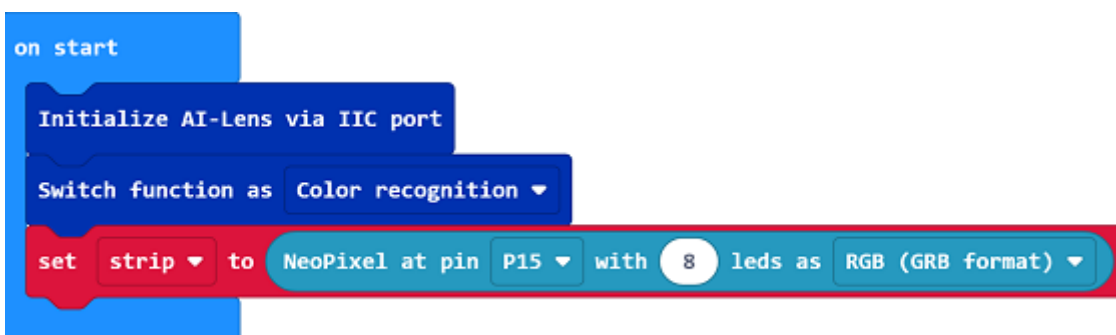
We need to add a package for programming the AI lens kit. Click “Extensions” in the bottom of the drawer and search with “https://github.com/electfreaks/pxt-PlanetX-AI” in the dialogue box to download it.



*Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.*

## Step 2

- In the “on start” brick, initialize the AI lens and switch the function to the color recognition mode, set the neopixel lights connect to P15 port.





- In the “forever” brick, set to get one image from the AI lens and judge the cards color on the image. If it recognizes the white color, we set the LED headlights and signal lights in white; if we recognizes the blue color, we set the LED headlights and signal lights in blue, by analogy, we programme with green, red, yellow and black card in the same way.

```
forever
  Get one image from AI-Lens
  if Image contains color card(s): White then
    Set LED headlights ALL color
    strip show color white
  +
  if Image contains color card(s): Blue then
    Set LED headlights ALL color
    strip show color blue
  +
  if Image contains color card(s): Green then
    Set LED headlights ALL color
    strip show color green
  +
  if Image contains color card(s): Red then
    Set LED headlights ALL color
    strip show color red
  +
  if Image contains color card(s): Yellow then
    Set LED headlights ALL color
    strip show color yellow
  +
  if Image contains color card(s): Black then
    Set LED headlights ALL color
    strip show color black
  +
```

Code

on start

Initialize AI-Lens via IIC port

Switch function as Color recognition

set strip to NeoPixel at pin P15 with 8 leds as RGB (GRB format)

forever

Get one image from AI-Lens

if Image contains color card(s): White then

Set LED headlights ALL color

strip show color white

+

if Image contains color card(s): Blue then

Set LED headlights ALL color

strip show color blue

+

if Image contains color card(s): Green then

Set LED headlights ALL color

strip show color green

+

if Image contains color card(s): Red then

Set LED headlights ALL color

strip show color red

+

if Image contains color card(s): Yellow then

Set LED headlights ALL color

strip show color yellow

+

if Image contains color card(s): Black then

Set LED headlights ALL color

strip show color black

+

Link: [https://makecode.microbit.org/\\_EL876k2ykeaW](https://makecode.microbit.org/_EL876k2ykeaW)

You may also download it directly below:

[▶ Simulator](#)   [🧩 Blocks](#)   [JS JavaScript](#)   [⌵](#)   [🔗 Edit](#)

[Microsoft MakeCode](#) | [Terms of Use](#) | [Privacy](#) | [📄 Download](#)

---

## Result

---

- The lights change the color in accordance with the color of the cards.

## Exploration

---

## FAQ

---

## Relevant Files

---

## 18.4. Cutebot & AI Lens Balls Tracking

### Purpose

---

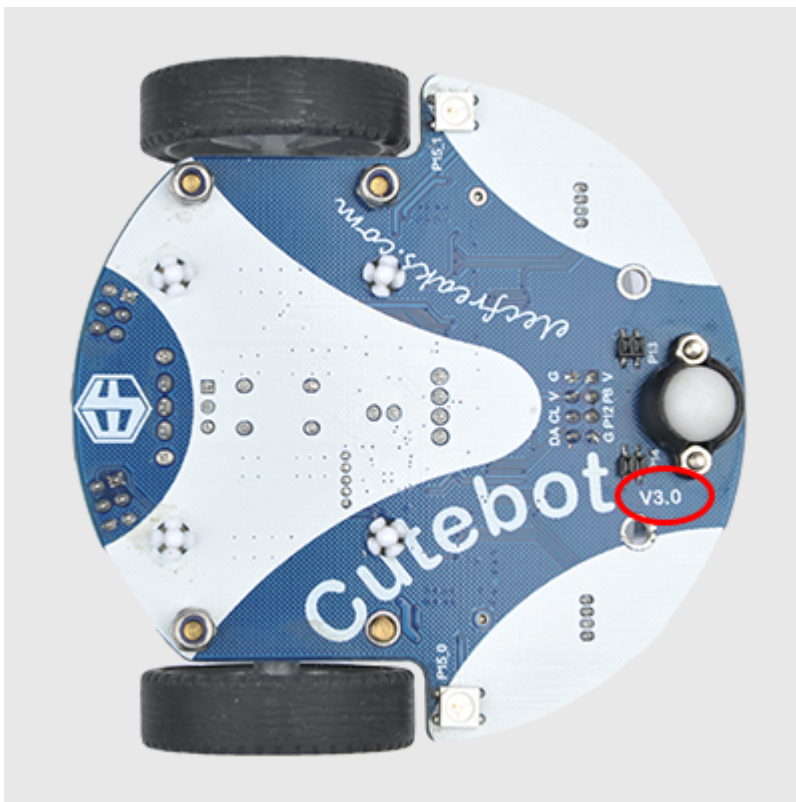
- To make a ball-tracking Cutebot with the AI Lens.

### Materials required

---

- 1 × Cutebot V3.0
- 1 × Cutebot lithium battery pack
- 1 × AI Lens Kit

*Note: The AI Lens kit works with Cutebot V3.0 only(You can see the version number printed on the baseboard).*



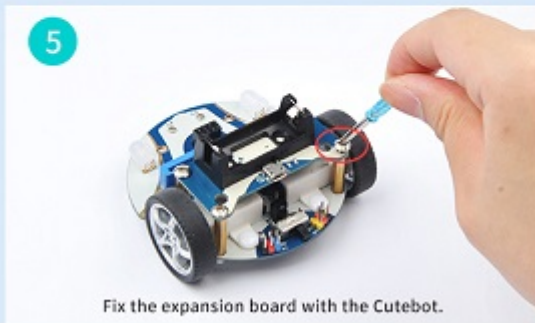
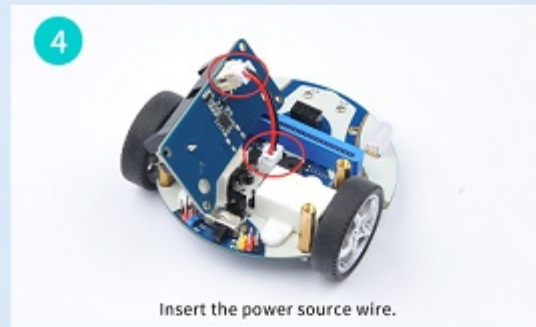
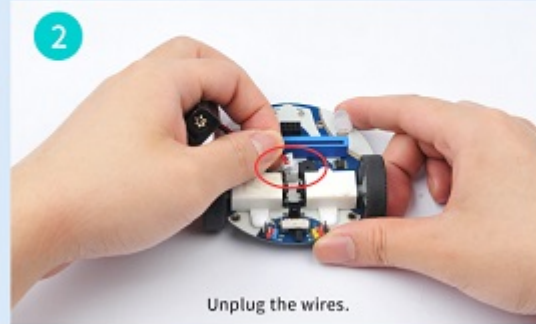
### Connections:

---

### Steps to install the lithium battery pack:

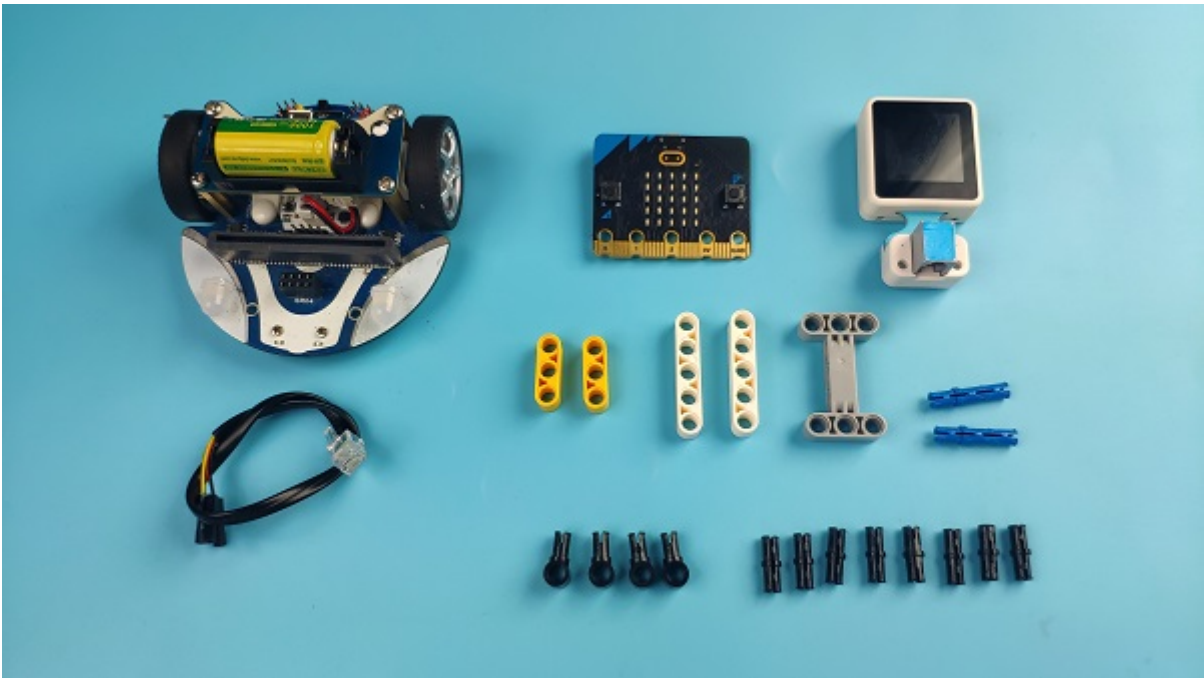
## Assembly steps for the battery pack

Follow the below assembly figures



Assembly steps for bricks:

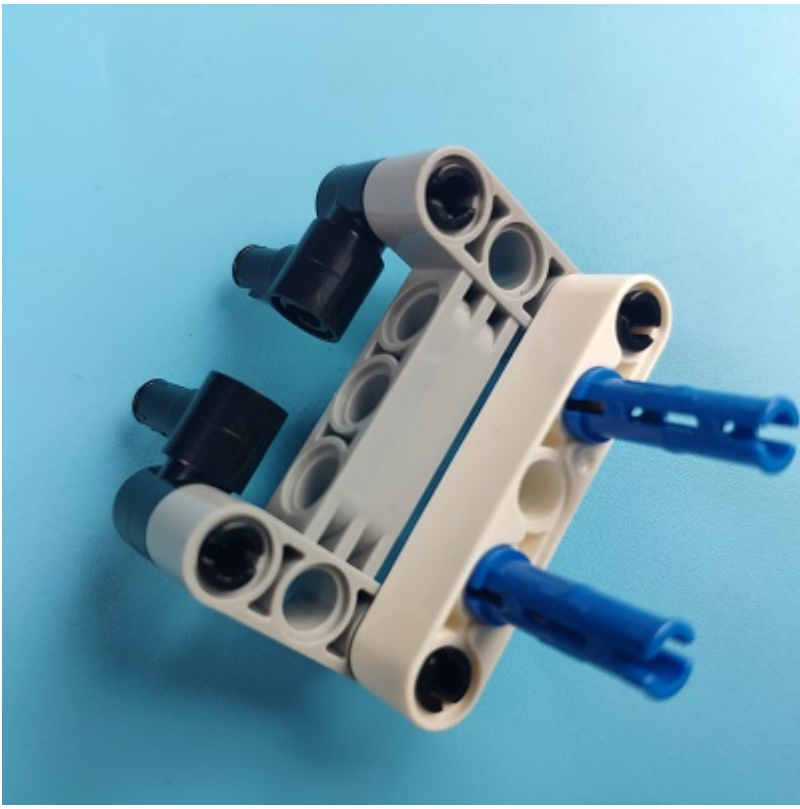
Parts list:

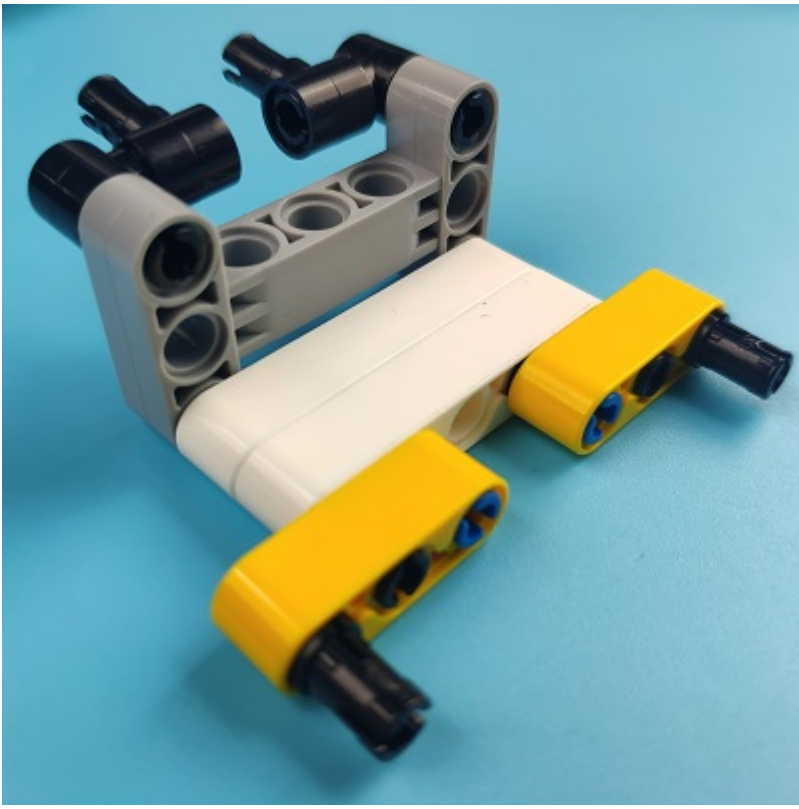


Steps of build-up:

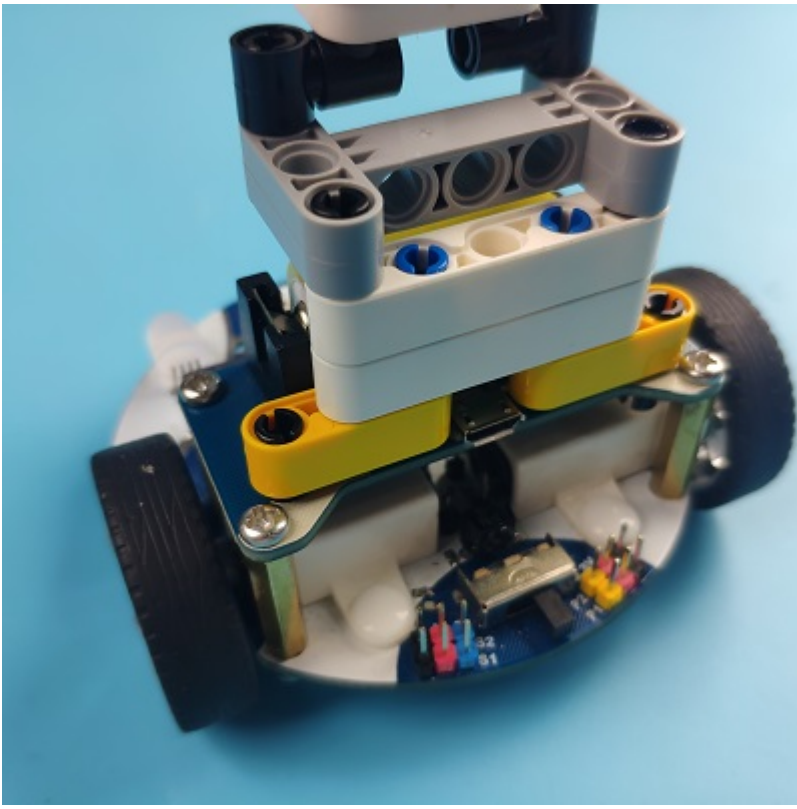






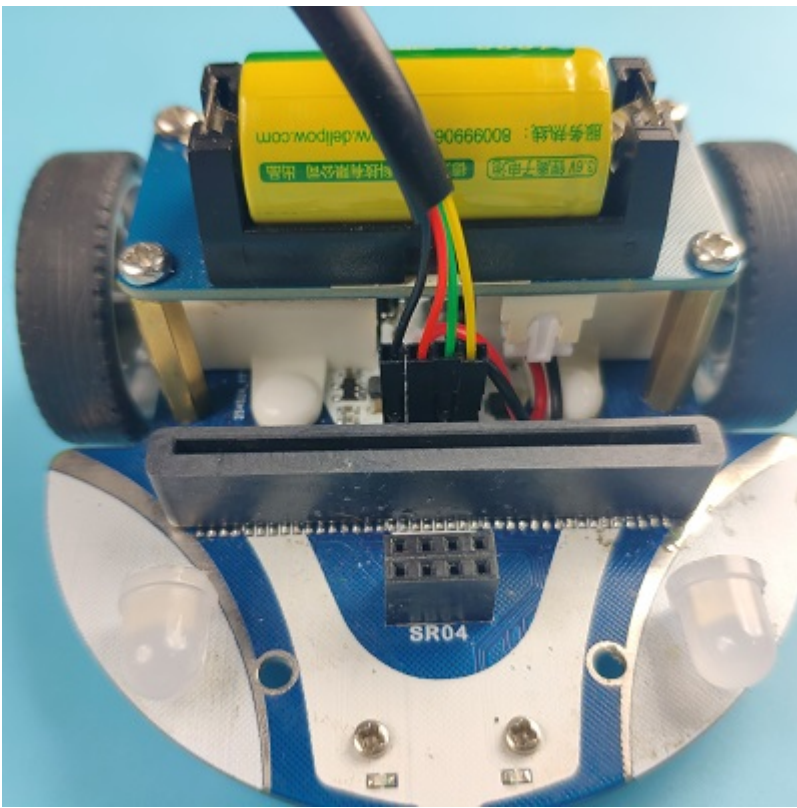






### **Connections of the AI Lens:**

Connect the RJ11 cable with the AI Lens and the other end in Dupont connection to the circled place in the below picture (make sure you connect to the right connections).



*Tips: the bricks holder here is flexible to be adjusted, we may manually adjust the angles of the AI lens to meet the requirements of the functions that you want to achieve.*

## **Software Platform:**

---

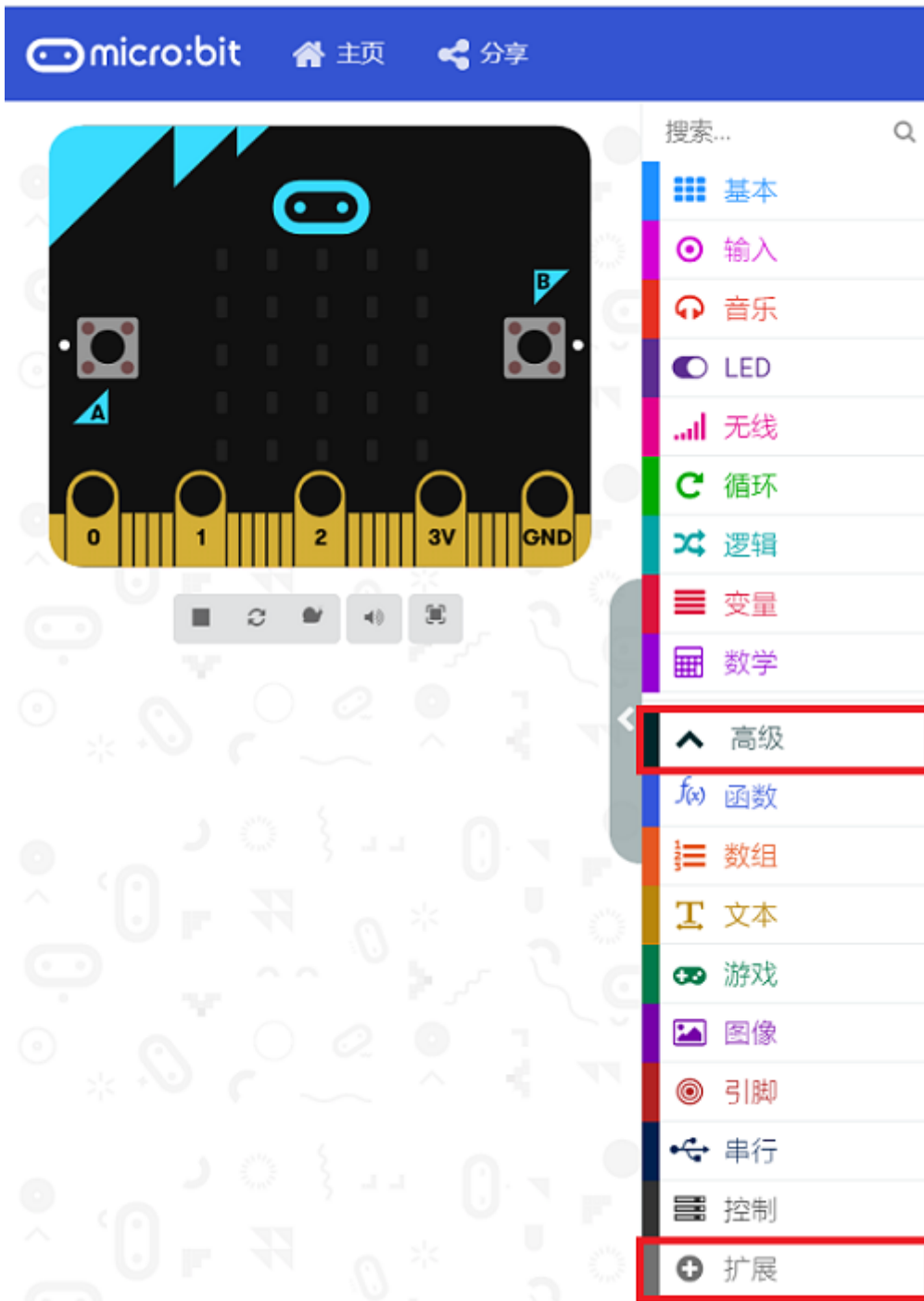
MicroSoft MakeCode

## **Programming**

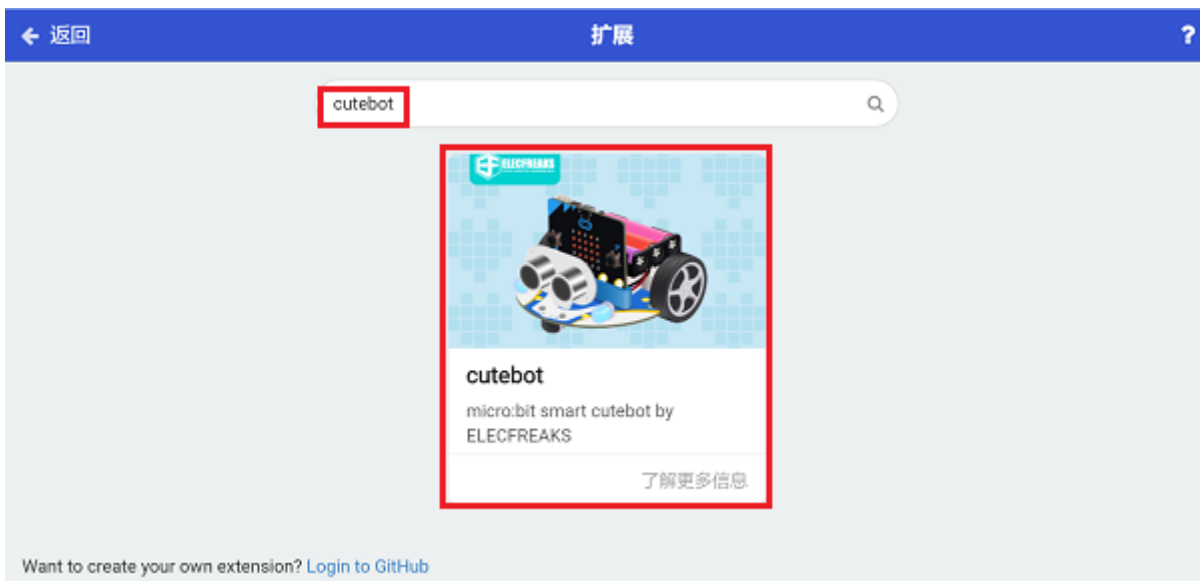
---

### **Step 1**

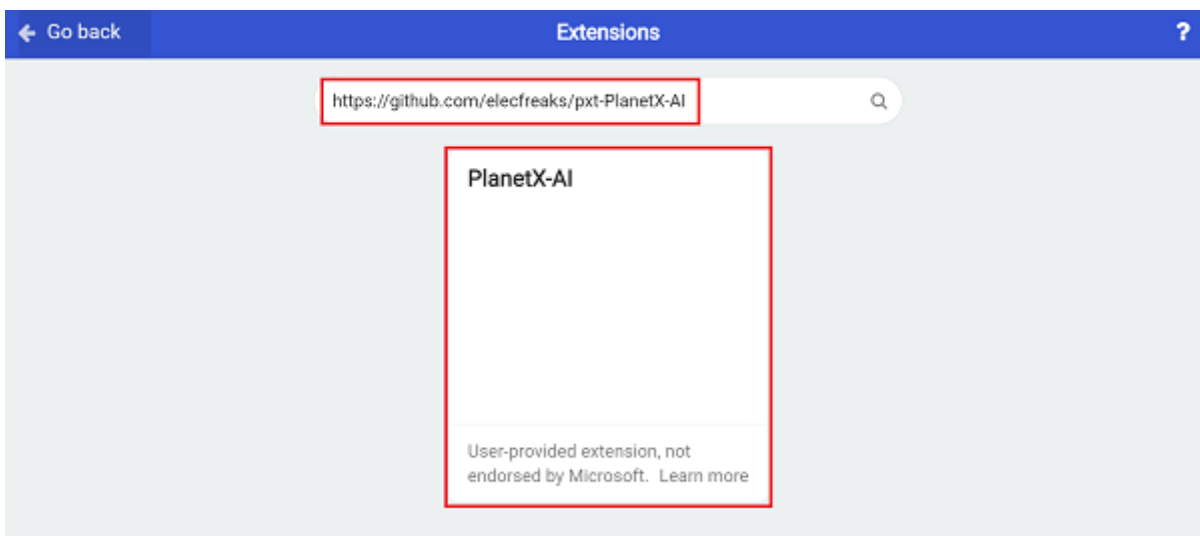
Click “Advanced” in the drawer to see more choices.



- We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “cutebot” in the dialogue box to download it.



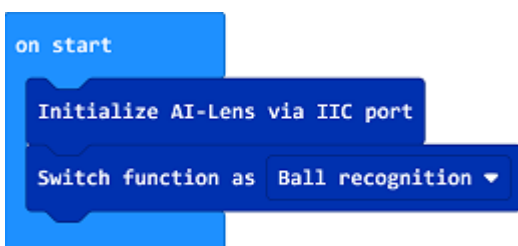
We need to add a package for programming the AI lens kit. Click “Extensions” in the bottom of the drawer and search with “<https://github.com/electfreaks/pxt-PlanetX-AI>” in the dialogue box to download it.



*Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.*

## Step 2

- In the “on start” brick, initialize the AI lens and switch the function to the ball tracking mode.



- In the “forever” brick, set to get one image form the AI lens.

- If there is a ball in the image, judge the size of the ball. If the size detected is below 100, it means the ball is far from the Cutebot, and then get the place of the ball with a value from the X axis, if the value is below 80, it means the ball is on the left front side of the Cutebot, we need set the speed of the left wheel at 0% and the right wheel at 20% to make the car turn left. If the value of the X axis is over 144, it means the ball is on the right front side of the Cutebot, we need set the speed of the left wheel at 20% and the right at 0% to make the car turn right; or we set both of the wheels at 25%; If the size of the ball is not less than 100, it means the ball is near the Cutebot, now we set the Cutebot to stop moving.

```
forever
  Get one image from AI-Lens
  if Image contains ball(s) then
    if In the image get ball(s)' info: Size < 100 then
      if In the image get ball(s)' info: X < 80 then
        Set left wheel speed 0 % right wheel speed 20 %
      else if In the image get ball(s)' info: X > 144 then
        Set left wheel speed 20 % right wheel speed 0 %
      else
        Set left wheel speed 25 % right wheel speed 25 %
    else
      Stop car immediatly
```

## Code

```
on start
  Initialize AI-Lens via IIC port
  Switch function as Ball recognition

forever
  Get one image from AI-Lens
  if Image contains ball(s) then
    if In the image get ball(s)' info: Size < 100 then
      if In the image get ball(s)' info: X < 80 then
        Set left wheel speed 0 % right wheel speed 20 %
      else if In the image get ball(s)' info: X > 144 then
        Set left wheel speed 20 % right wheel speed 0 %
      else
        Set left wheel speed 25 % right wheel speed 25 %
    else
      Stop car immediatly
```

Link: [https://makecode.microbit.org/\\_FWL7247fyfCk](https://makecode.microbit.org/_FWL7247fyfCk)

You may also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼

🔗 Edit

## Result

---

- The Cutebot goes for the ball if the AI Lens detects the ball and if the distance gets smaller enough, the Cutebot stops moving.

## Exploration

---

## FAQ

---

## Relevant Files

---

## 18.5. Cutebot & AI Lens-One Button to Learn

### Purpose

---

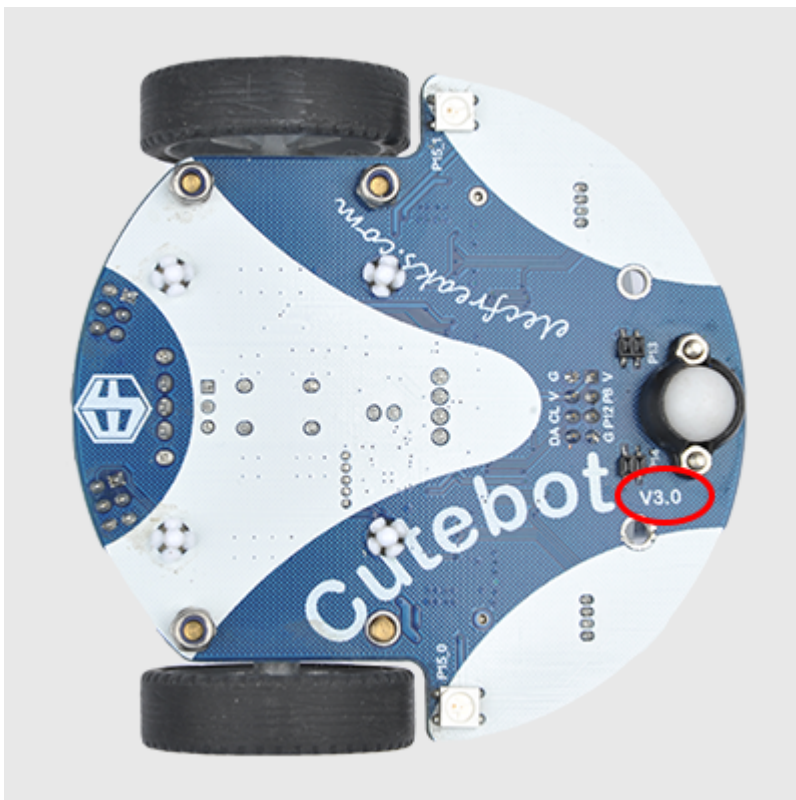
- Use the Cutebot and AI Lens to achieve the one button to learn function.

### Materials required

---

- 1 × Cutebot V3.0
- 1 × Cutebot lithium battery pack
- 1 × AI Lens Kit

*Note: The AI Lens kit works with Cutebot V3.0 only(You can see the version number printed on the baseboard).*



### Connections:

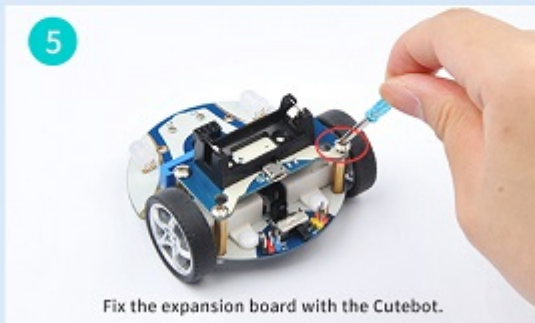
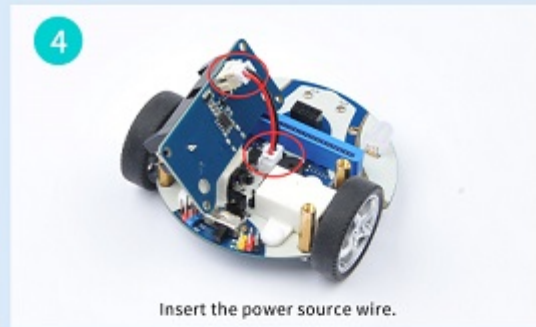
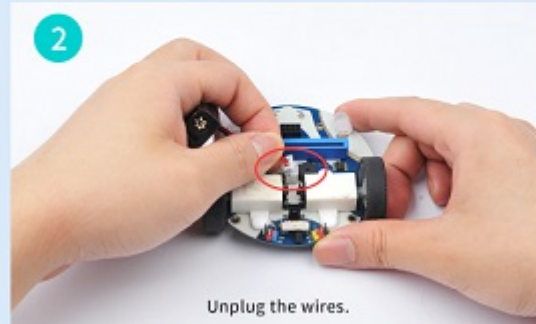
---

### Steps to install the lithium battery pack:



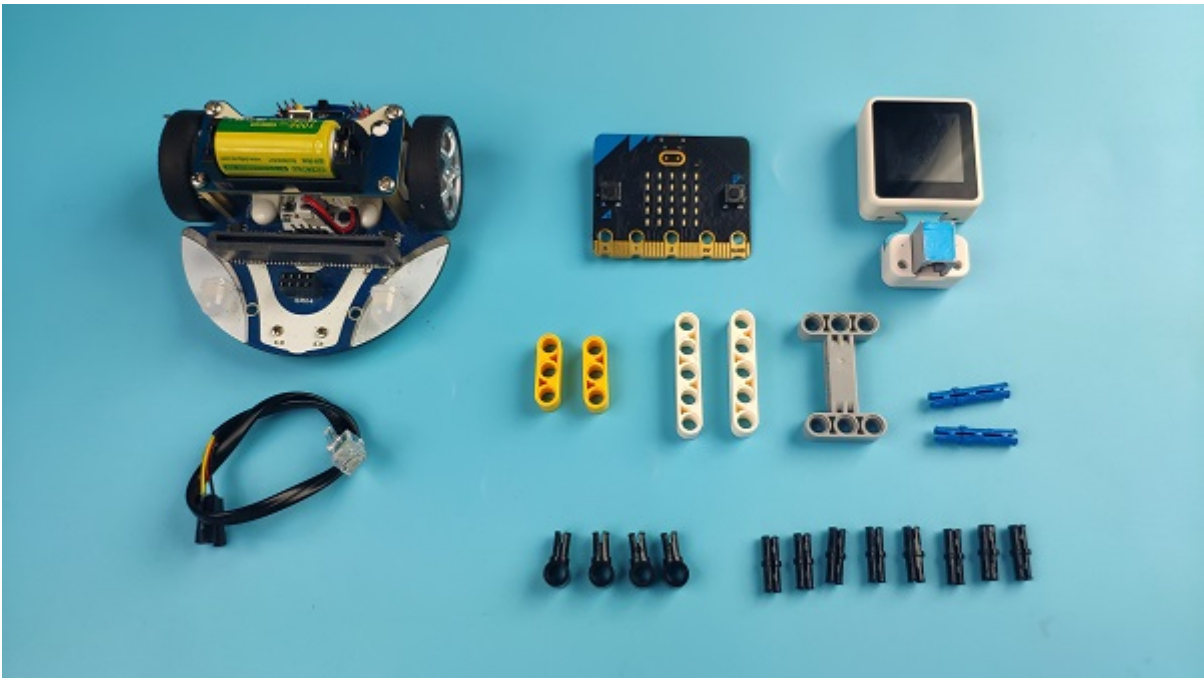
## Assembly steps for the battery pack

Follow the below assembly figures



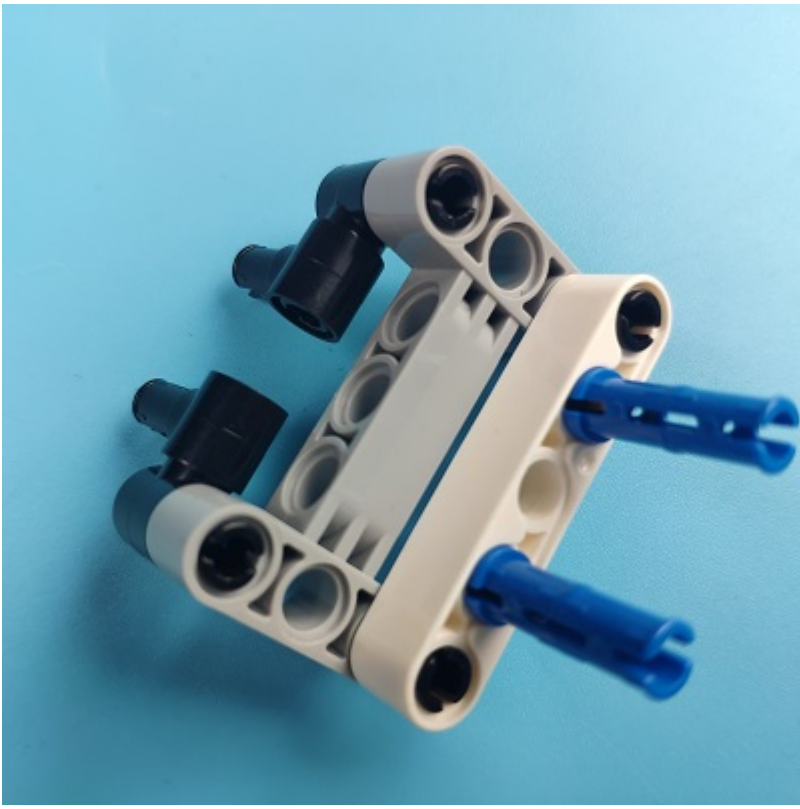
Assembly steps for bricks:

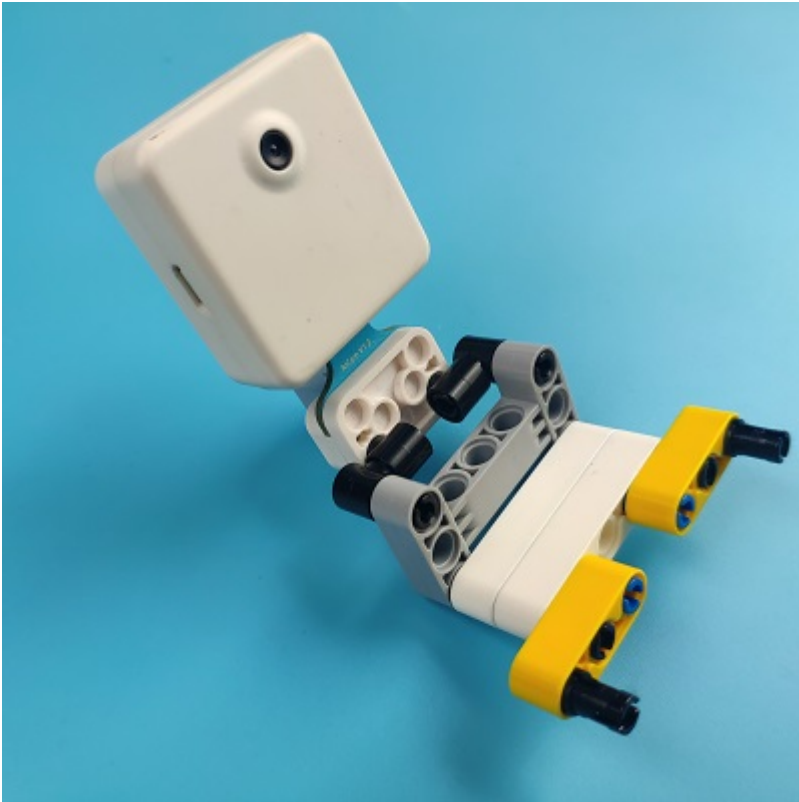
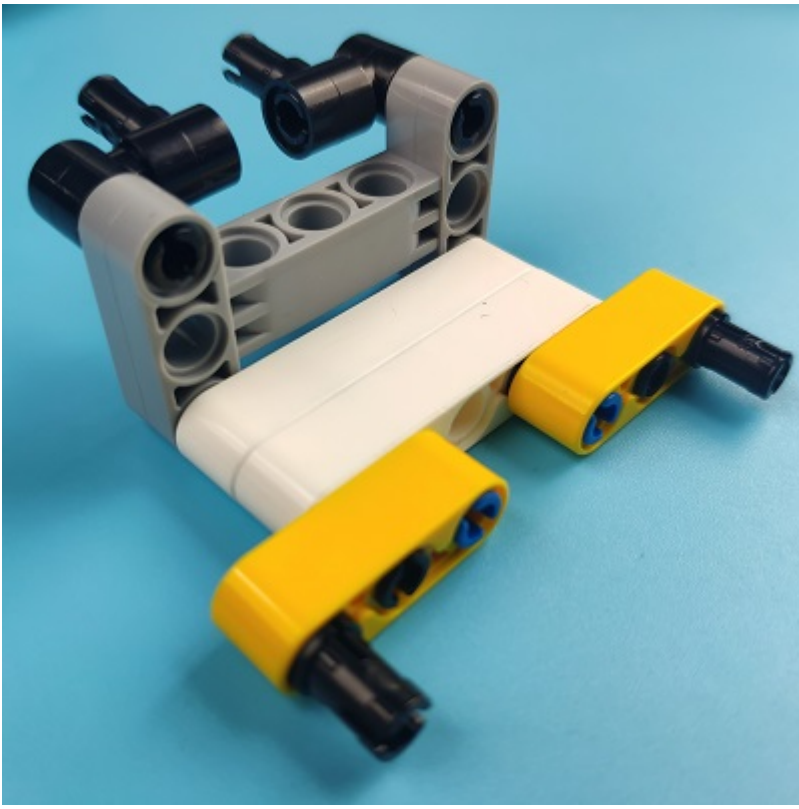
Parts list:



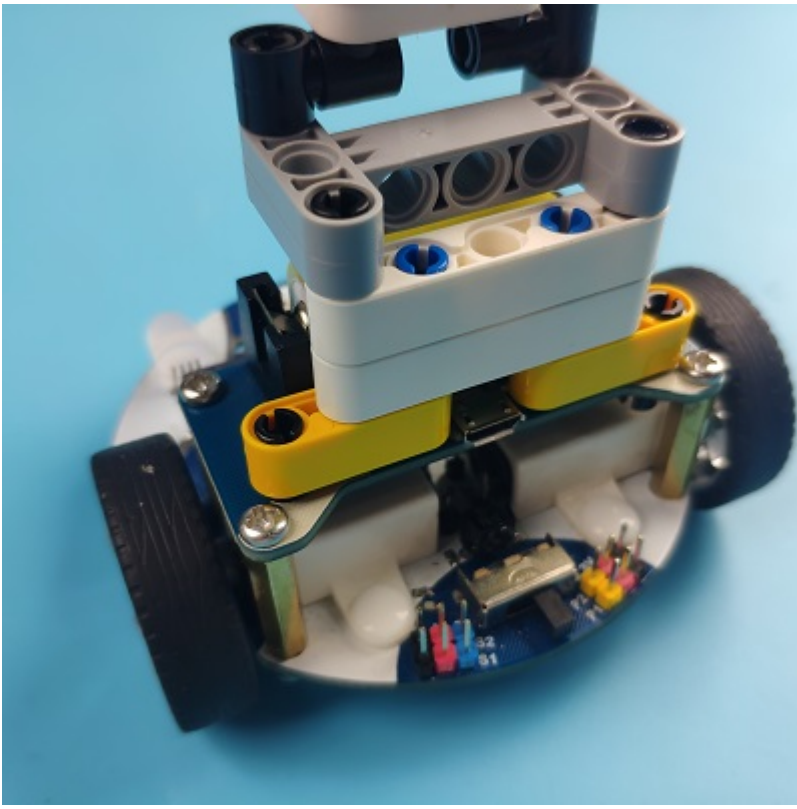
Steps of build-up:





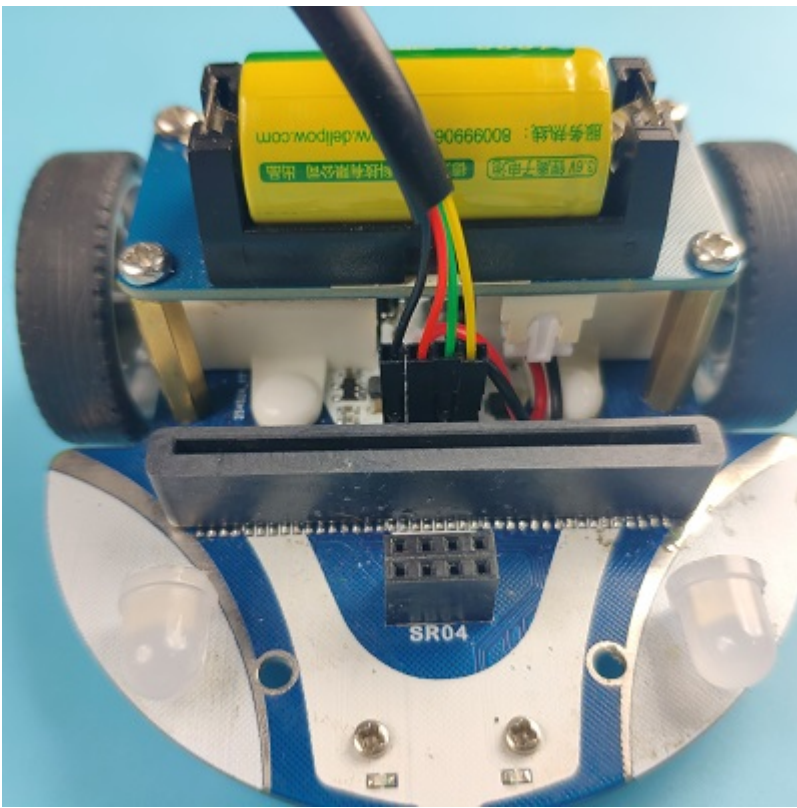






### **Connections of the AI Lens:**

Connect the RJ11 cable with the AI Lens and the other end in Dupont connection to the circled place in the below picture (make sure you connect to the right connections).



*Tips: the bricks holder here is flexible to be adjusted, we may manually adjust the angles of the AI lens to meet the requirements of the functions that you want to achieve.*

## **Software Platform:**

---

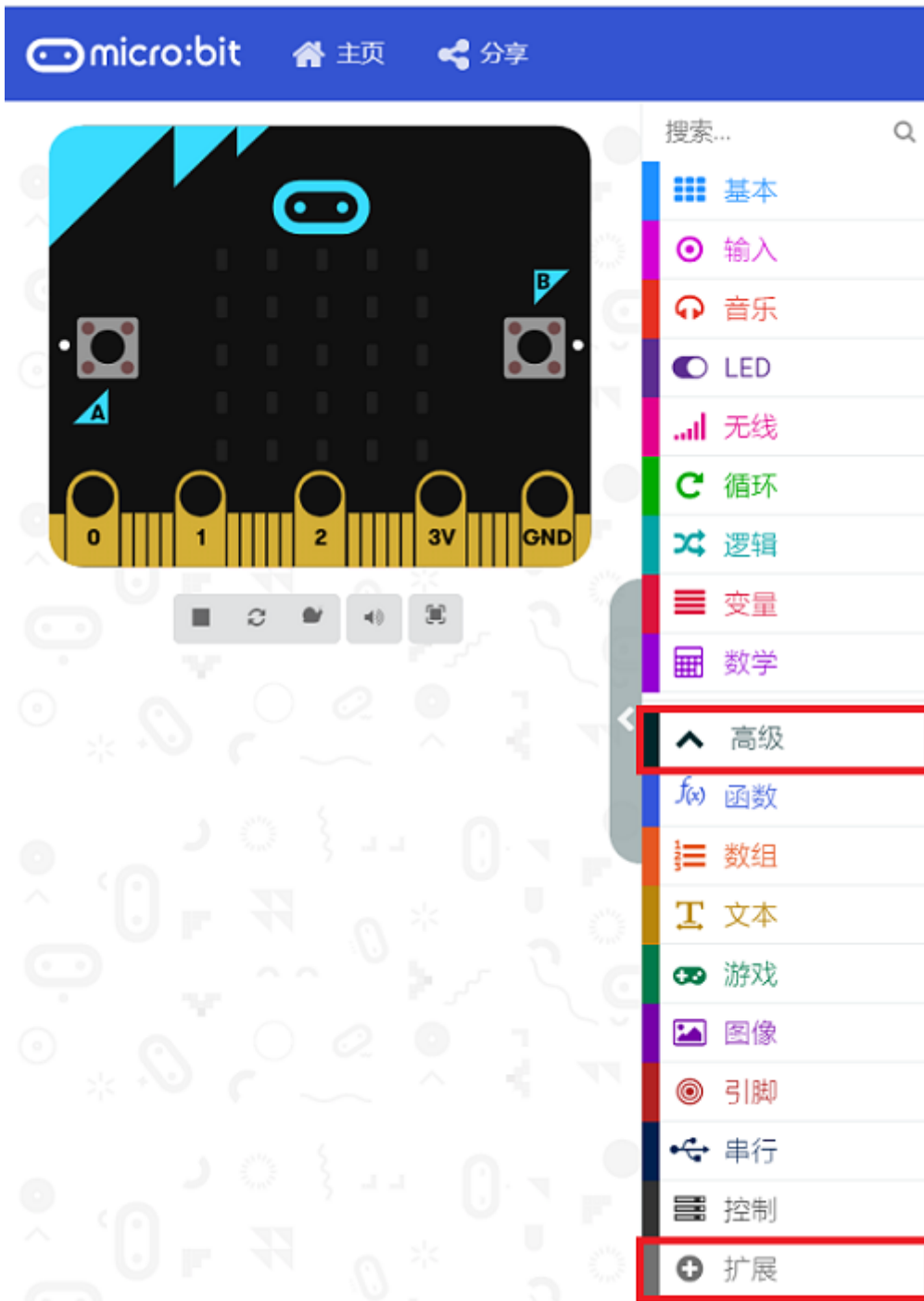
MicroSoft MakeCode

## **Programming**

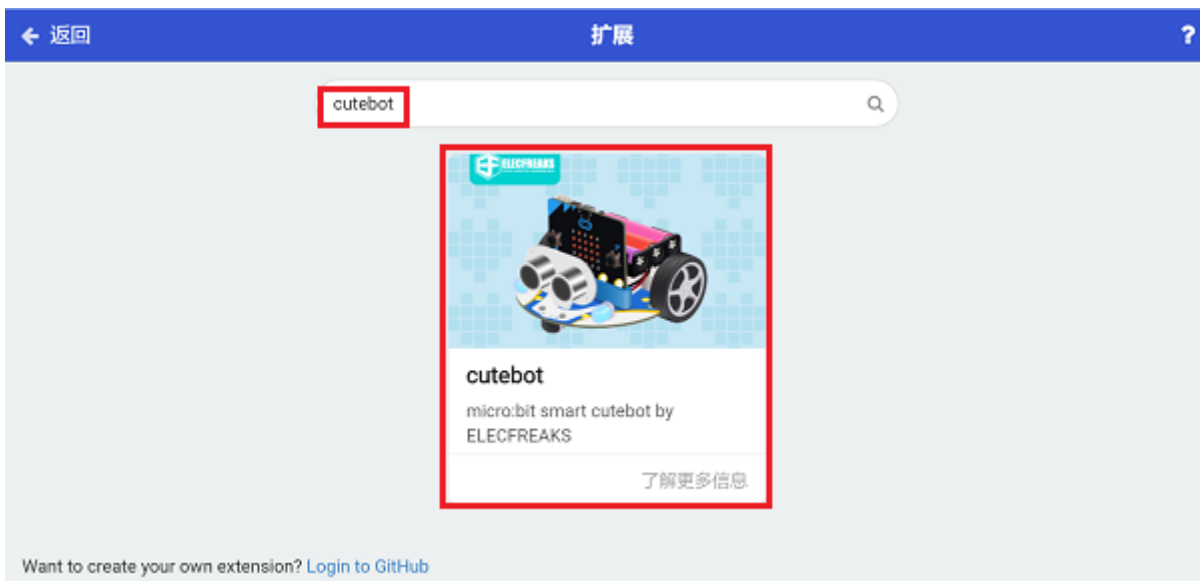
---

### **Step 1**

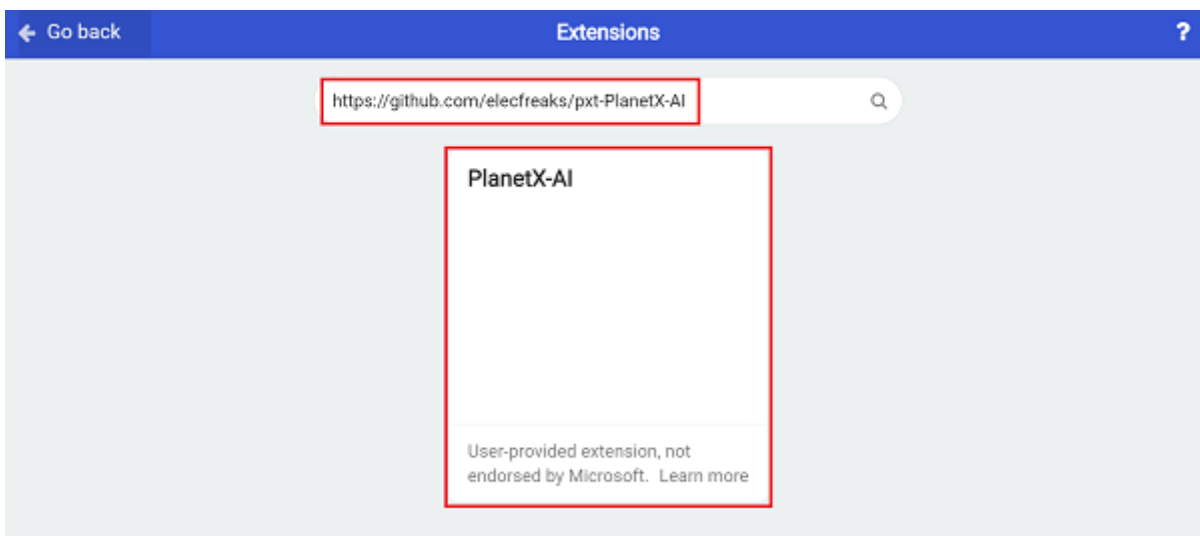
Click “Advanced” in the drawer to see more choices.



- We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “cutebot” in the dialogue box to download it.



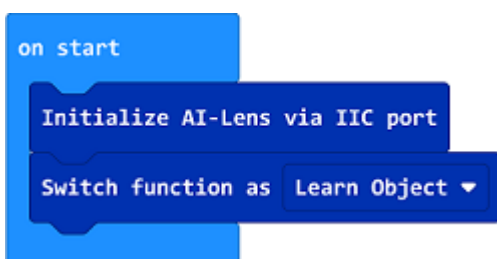
We need to add a package for programming the AI lens kit. Click “Extensions” in the bottom of the drawer and search with “<https://github.com/electfreaks/pxt-PlanetX-AI>” in the dialogue box to download it.



*Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.*

## Step 2

- In the on start brick, initialize the AI Lens and switch to characters acquisition function.



- While press the button A, set the learnt target as ID1.



```
on button A pressed
  Learn an object with: ID1
```

- In the “forever” brick, set to get one image from the AI lens. If the ID1 is in the image, we set the color of the headlights in blue and set a ✓ to display on the screen. Or set the color of the headlights in red and set a ✗ to display on the screen.

```
forever
  Get one image from AI-Lens
  if Image contains learned objects: ID1 then
    Set LED headlights ALL color blue
    show icon ✓
  else
    Set LED headlights ALL color red
    show icon ✗
```

## Code

```
on start
  Initialize AI-Lens via IIC port
  Switch function as Learn Object ▼

on button A ▼ pressed
  Learn an object with: ID1 ▼

forever
  Get one image from AI-Lens
  if Image contains learned objects: ID1 ▼ then
    Set LED headlights ALL ▼ color ●
    show icon [ ] ▼
  else
    Set LED headlights ALL ▼ color ●
    show icon [ ] ▼
```

Link: [https://makecode.microbit.org/\\_8Xe7ERhxEgza](https://makecode.microbit.org/_8Xe7ERhxEgza)

You may also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼    ↗ Edit

---

## Result

---

- While pressing button A, set the learnt target as ID1, after learning it, the LED headlights light on in blue and the icon √ is displaying on the screen if ID1 is recognized; or set the LED lights on in red and display icon × .

## Exploration

---

## FAQ

---

## Relevant Files

---

## 18.6. Cutebot & AI Lens- Face Tracking

### Purpose

---

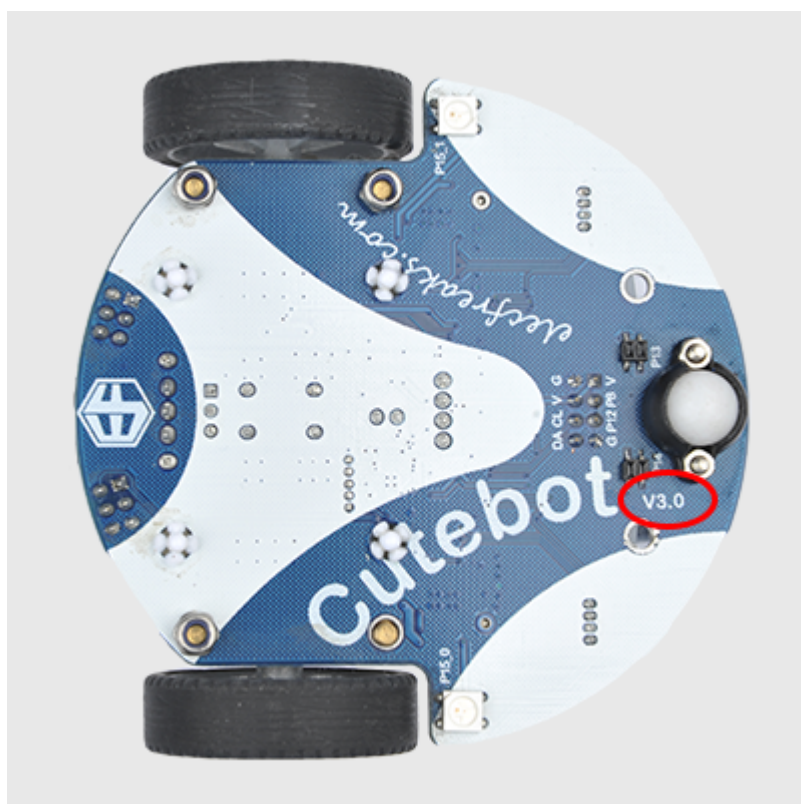
- Use the Cutebot and AI Lens to achieve the face tracking function.

### Materials required

---

- 1 × Cutebot V3.0
- 1 × Cutebot lithium battery pack
- 1 × AI Lens Kit

*Note: The AI Lens kit works with Cutebot V3.0 only(You can see the version number printed on the baseboard).*



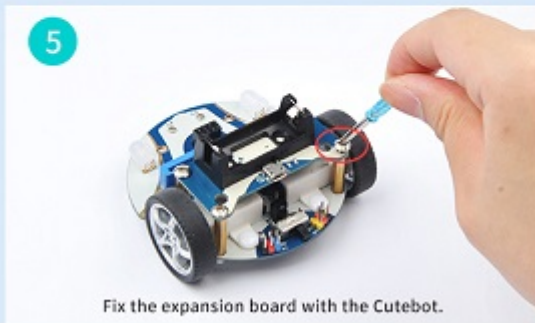
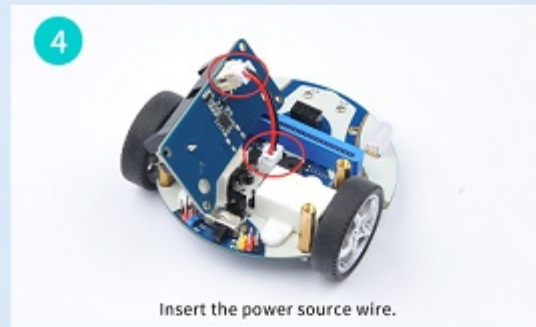
### Connections:

---

### Steps to install the lithium battery pack:

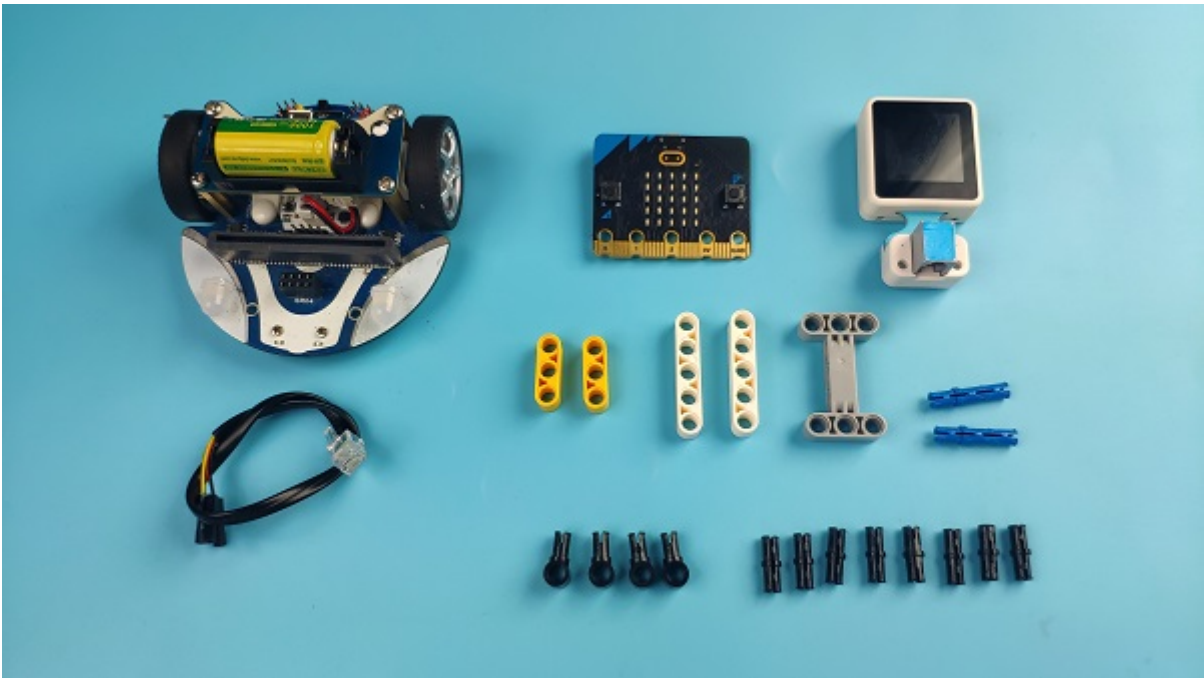
## Assembly steps for the battery pack

Follow the below assembly figures



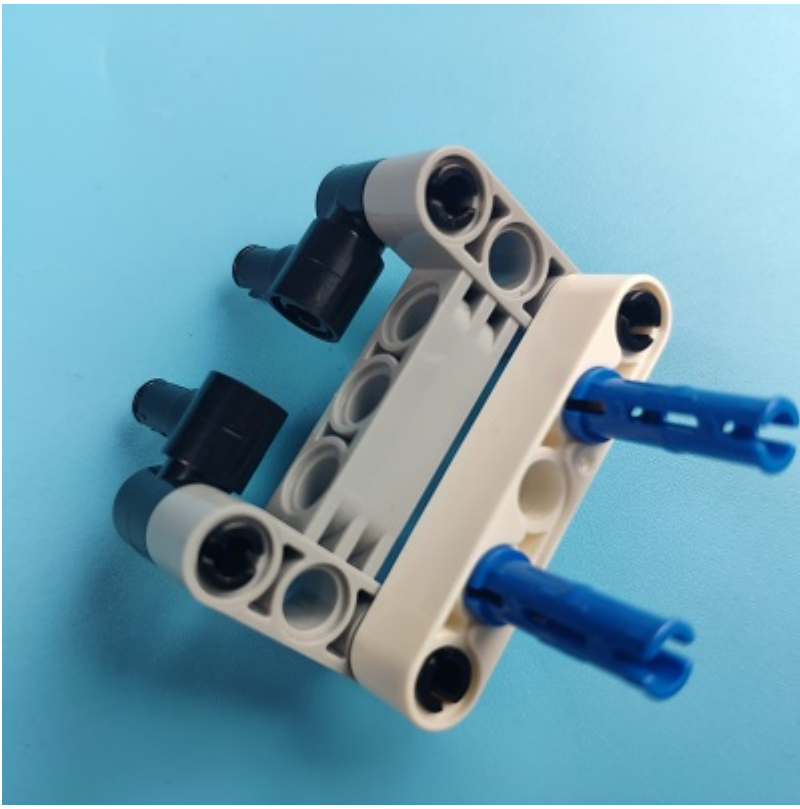
Assembly steps for bricks:

Parts list:

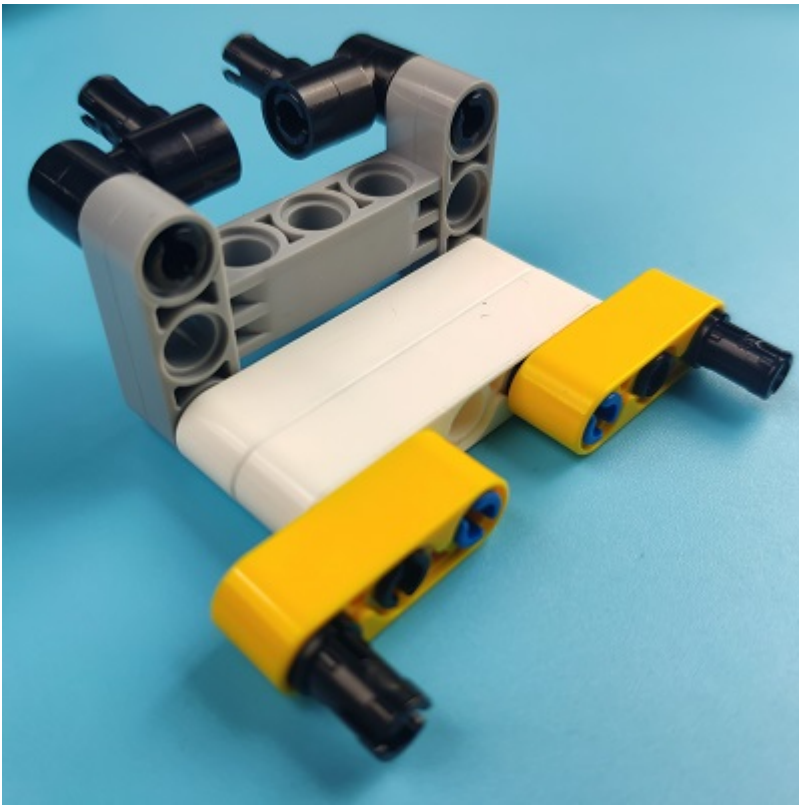


Steps of build-up:

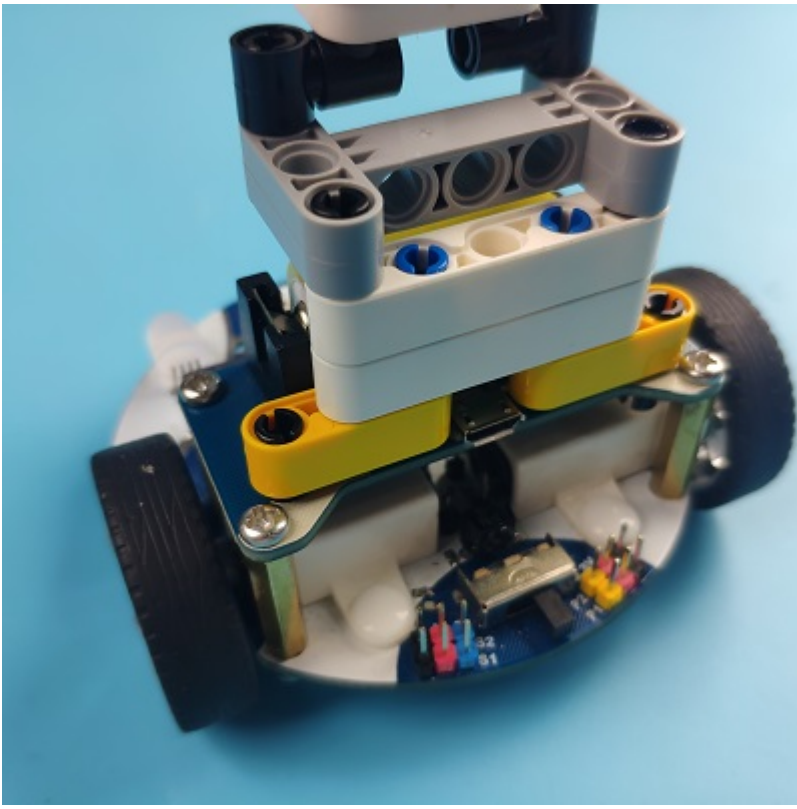






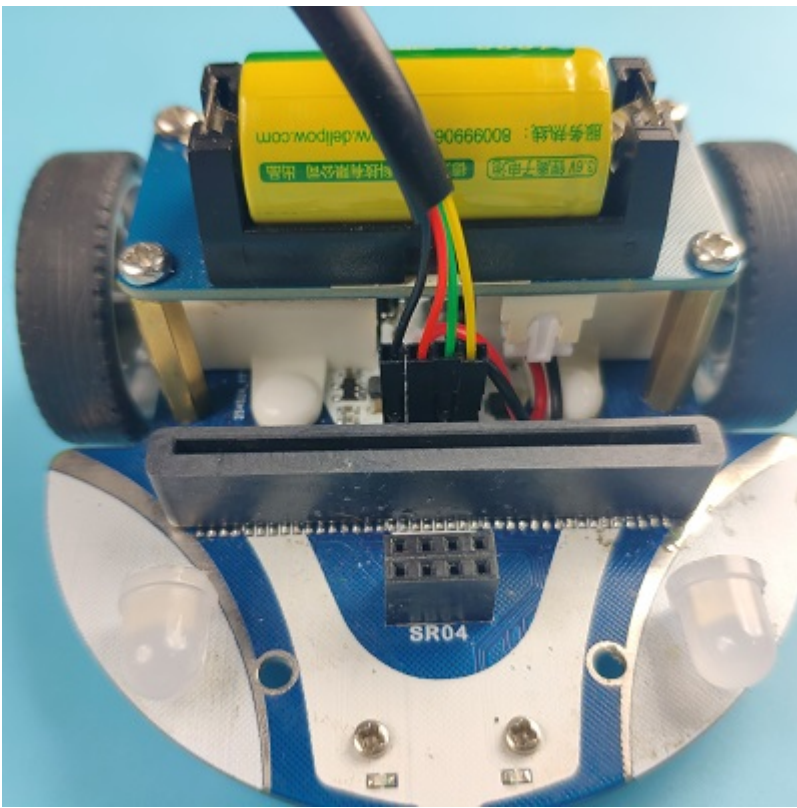






### **Connections of the AI Lens:**

Connect the RJ11 cable with the AI Lens and the other end in Dupont connection to the circled place in the below picture (make sure you connect to the right connections).



*Tips: the bricks holder here is flexible to be adjusted, we may manually adjust the angles of the AI lens to meet the requirements of the functions that you want to achieve.*

## **Software Platform:**

---

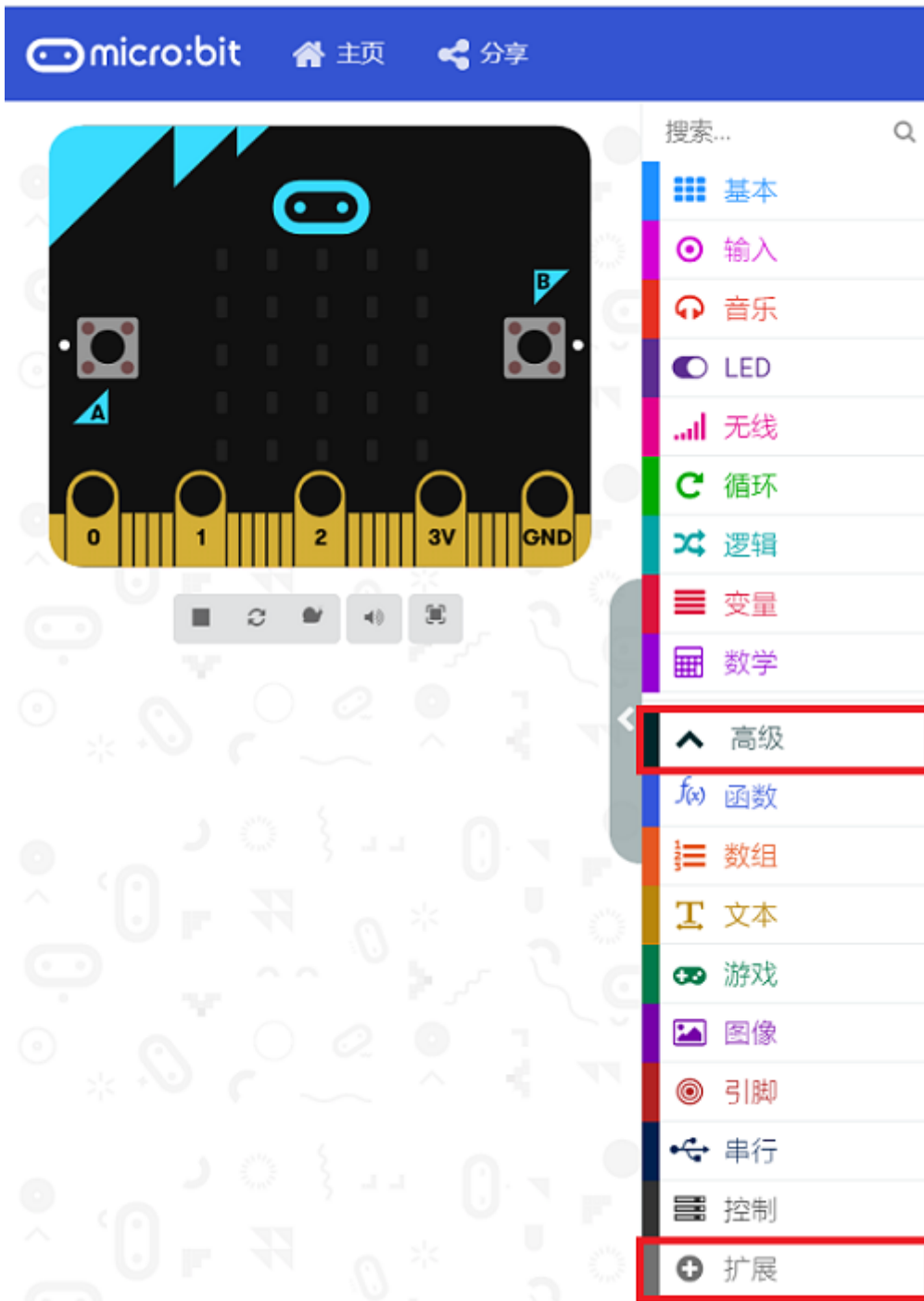
MicroSoft MakeCode

## **Programming**

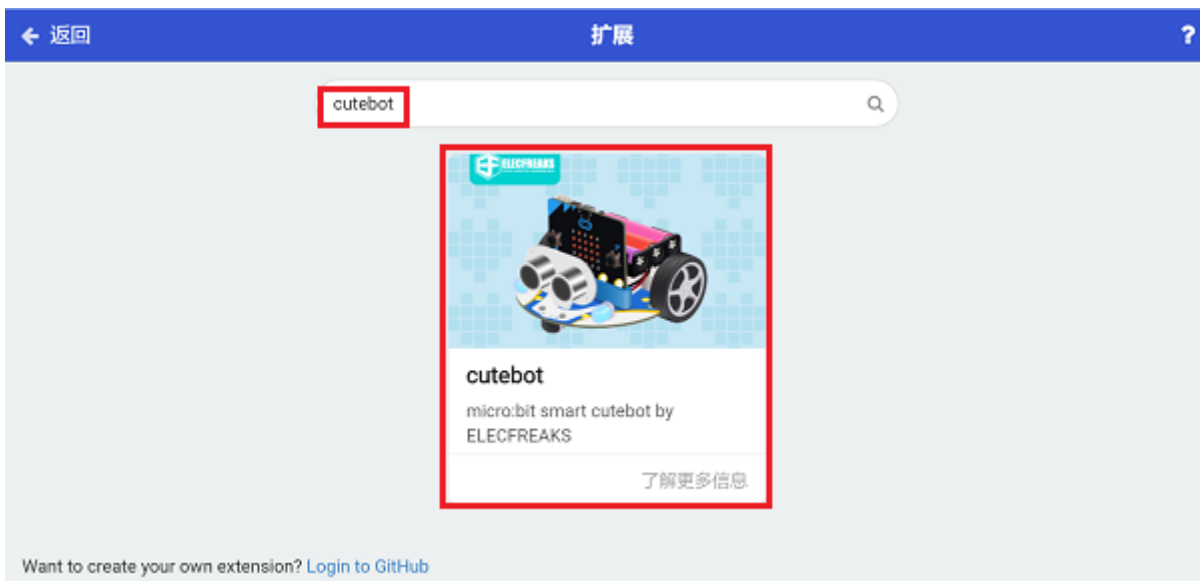
---

### **Step 1**

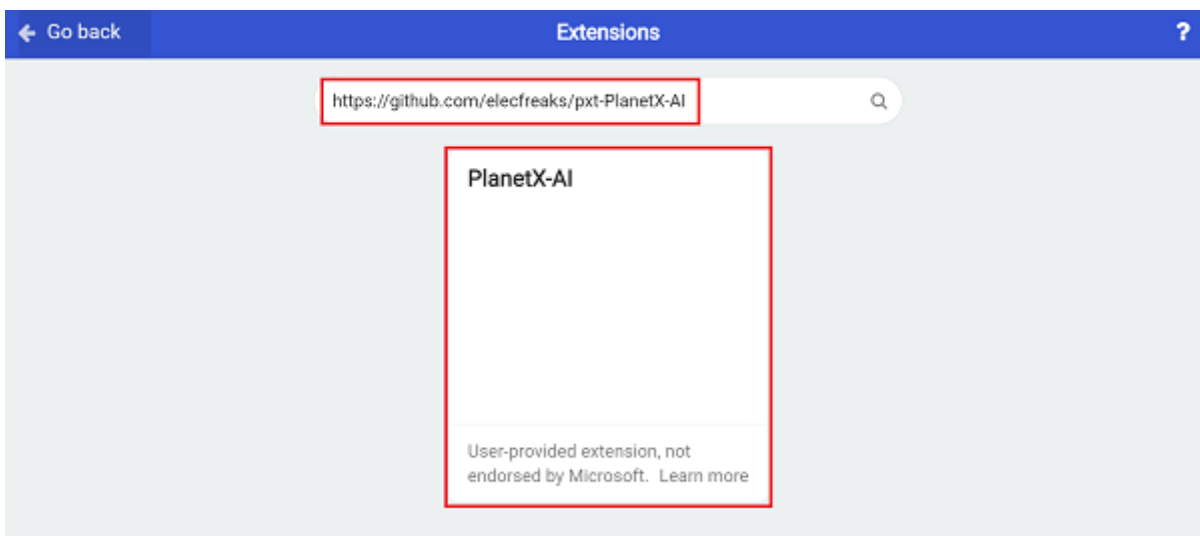
Click “Advanced” in the drawer to see more choices.



- We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “cutebot” in the dialogue box to download it.



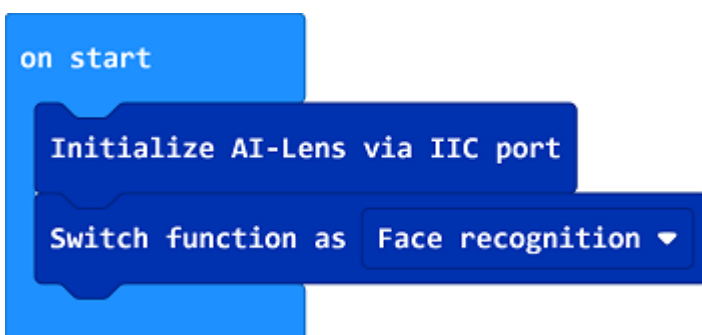
We need to add a package for programming the AI lens kit. Click “Extensions” in the bottom of the drawer and search with “https://github.com/electfreaks/pxt-PlanetX-AI” in the dialogue box to download it.



*Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.*

## Step 2

- In the on start brick, initialize the AI Lens and switch to face recognition function.



- In the “forever” brick, set to get one image from the AI lens. If a face is on the image, set the LED screen to display √; or set to display ×.

```
forever
  Get one image from AI-Lens
  if Image contains a face then
    show icon [5x5 grid icon]
  else
    show icon [5x5 grid icon with 'x']
```

## Link

```
on start
  Initialize AI-Lens via IIC port
  Switch function as Face recognition


forever
  Get one image from AI-Lens
  if Image contains a face then
    show icon [5x5 grid icon]
  else
    show icon [5x5 grid icon with 'x']
```

Link: [https://makecode.microbit.org/\\_TFYEFhLym9xk](https://makecode.microbit.org/_TFYEFhLym9xk)

You may also download it directly below:

▶ Simulator    🧩 Blocks    JS JavaScript    ▼

 Edit

[Microsoft MakeCode](#) | [Terms of Use](#) | [Privacy](#) |  [Download](#)

---

## Result

---

- If the AI Lens recognizes the face(s), the LED screen displays √; or it displays ×.

## Exploration

---

## FAQ

---

## Relevant Files

---