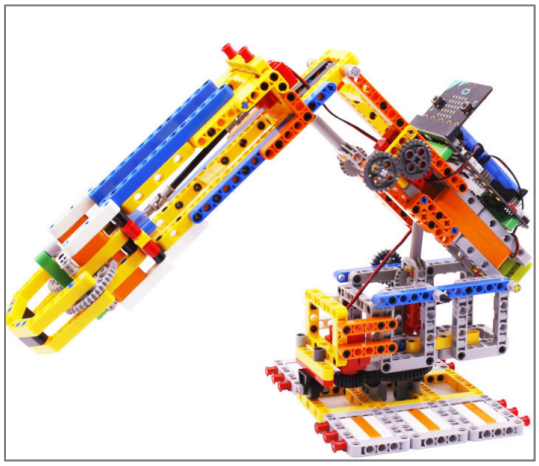
Arm:bit handle remote control



Learning goals:

In this lesson we will learn to use the Handle to remotely control the Arm:bit.

About wiring:

4.About wiring:

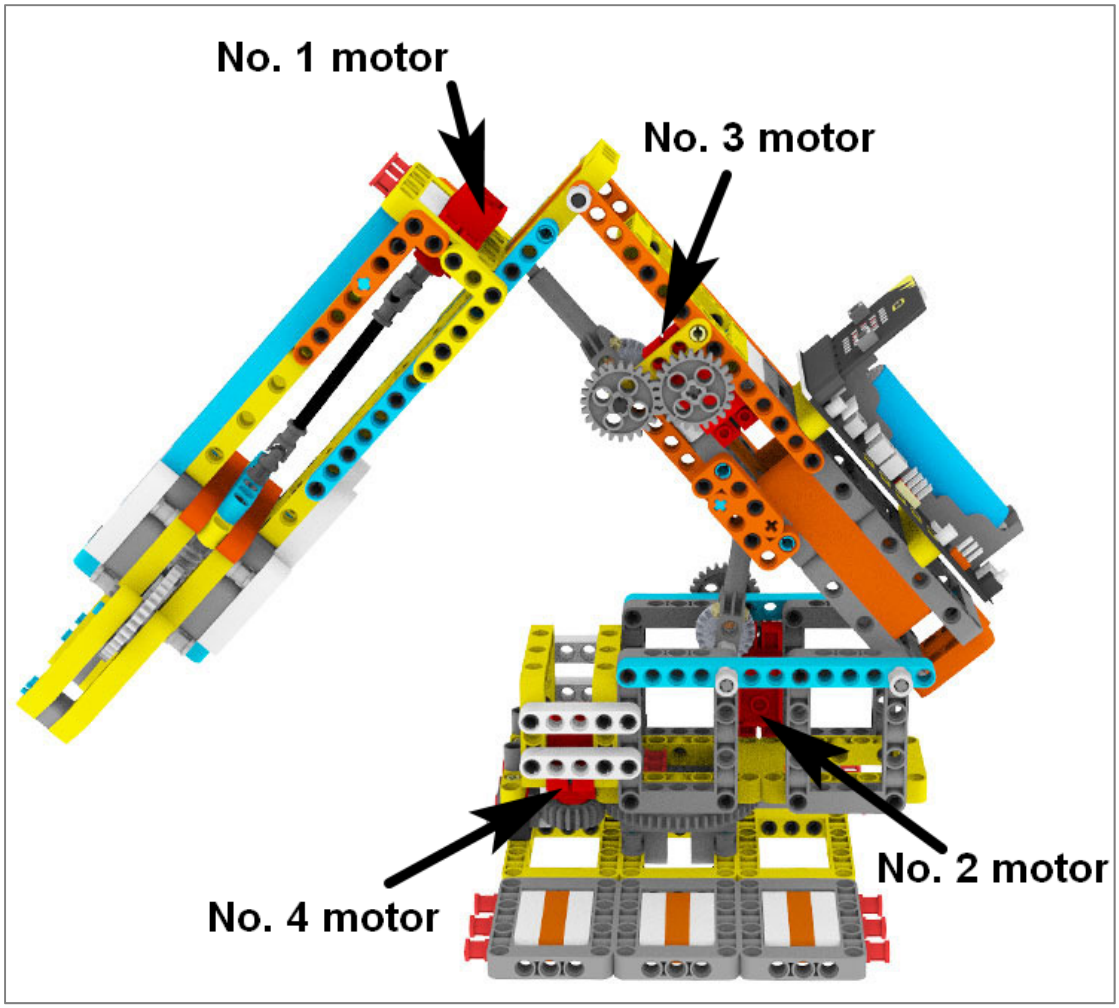
As shown below,

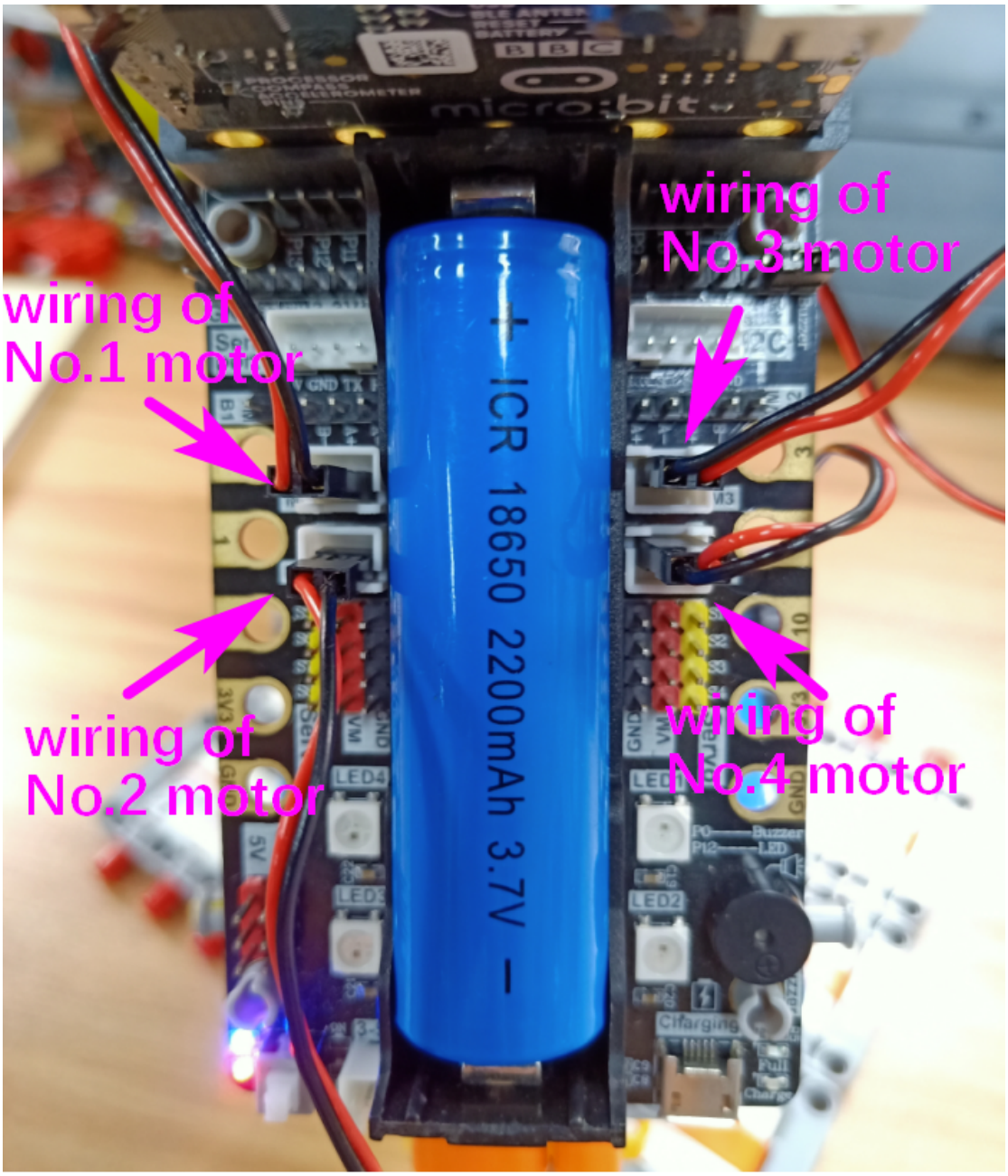
NO.1 motor connect to M1 interface of super:bit,

NO.2 motor connect to M2 interface of super:bit,

NO.3 motor connect to M3 interface of super:bit,

NO.4 motor connect to M4 interface of super:bit.


No. 1 motor
No. 3 motor
No. 2 motor
No. 4 motor



Wireless communication principles:

With the micro:bit radio module, different devices can work together through a simple wireless network. When the radio function is turned on for micro:bit, a simple wireless local area network is generated. The micro:bit board with radio function turned on can set parameters within the effective range.

Wireless communication is divided into sending and receiving two program blocks. Set the wireless group of radio to the same group, and the two micro:bit boards can communicate.

! Note:

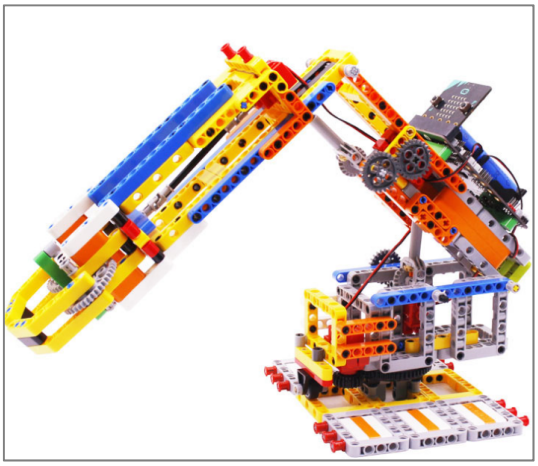
Due to the problem of the building block structure, if you want the spider

to move forward, the direction of the building block motor needs to be set backward.

About code:

Spider code:

Please use the MU software to open the **Arm_code.py** file we provided.



1) Import the libraries neopixel, super:bit and radio to be used.

display.show(Image.HEART): Show a “heart” on the micro:bit matrix;

radio.on (): Turn on the wireless function. Because the wireless function consumes more power and occupies memory, it is disabled by default. You can also use **radio.off ()** to turn off the wireless function.

radio.config (group = 1): configure wireless group = 1, so that other micro:bit devices with wireless group = 1 can communicate with each other, the default is 0, Range of group is 0 ~ 255.

np = neopixel.NeoPixel(pin12, 4): Define RGB lights on the expansion board, connect pin 12, there are 4 RGB lights in total.

!Note: the set group value It needs to be consistent with the handle setting, otherwise normal communication cannot be performed.

Code as shown below:

```
from microbit import *
import superbit
import radio
import neopixel

radio.on()
radio.config(group=1)


display.show(Image.HEART)

np = neopixel.NeoPixel(pin12, 4)
```

2) Control the car advance, back, spin left and spin right functions:
incoming = radio.receive (): Receives the wirelessly transmitted data and saves it to the “incoming” variable.
if incoming is 'up', the clip rise; 'down' makes the clip drop; 'left' makes the Arm:bit spin left; 'right' makes the Arm:bit spin right; And 'stop' makes the Arm:bit stop.
Code as shown below:

```
while True:
    incoming = radio.receive()
    if incoming == 'up':
        superbit.motor_control(superbit.M2, 255, 0)

    elif incoming == 'down':
        superbit.motor_control(superbit.M2, -255, 0)

    elif incoming == 'left':
        superbit.motor_control(superbit.M4, 180, 0)

    elif incoming == 'right':
        superbit.motor_control(superbit.M4, -180, 0)

    elif incoming == 'stop':
        superbit.motor_control(superbit.M1, 0, 0)
        superbit.motor_control(superbit.M2, 0, 0)
        superbit.motor_control(superbit.M3, 0, 0)
        superbit.motor_control(superbit.M4, 0, 0)
```

3) If incoming is ‘R’, the clip open, ‘G’ makes the arm rise, ‘B’ makes the arm

drop, and ‘Y’ makes the clip close.
Code as shown below:

```
if incoming == 'R':
    superbit.motor_control(superbit.M1, 255, 0)

elif incoming == 'G':
    superbit.motor_control(superbit.M3, -255, 0)

elif incoming == 'B':
    superbit.motor_control(superbit.M3, 255, 0)

elif incoming == 'Y':
    superbit.motor_control(superbit.M1, -255, 0)
```

!Note:
The value of incoming needs to correspond to the value sent by the handle. Only the same value can receive and execute the command.

Handle control code:
Please use the MU software to open the **Arm_handle_code.py** file we provided.

1) Import the libraries microbit, ghandle, and radio that you need to use.
display.show(Image("09090:09090:00900:09090:09090")): Show a icon on the micro:bit matrix;
radio.on (): Turn on the wireless function;
radio.config (group = 1): set wireless group = 1, which is consistent with the group of the car;
Code as shown below:

```
from microbit import *
import superbit
import radio
import neopixel

radio.on()
radio.config(group=1)


display.show(Image.HEART)

np = neopixel.NeoPixel(pin12, 4)
```

2) If it detects that **ghandle.rocker (ghandle.up)** is True, it means that the rocker of the handle is pushed up, and the 'up' command is sent wirelessly, and an upward icon is displayed on LED dot matrix.
If it detects that **ghandle.rocker(ghandle.down)** is True, it means that the rocker of the handle is pushed down, and the 'down' command is sent wirelessly, and an down icon is displayed on LED dot matrix.
If it detects that **ghandle.rocker(ghandle.left)** is True, it means that the rocker of the handle is pushed left, and the 'left' command is sent wirelessly, and an left icon is displayed on LED dot matrix.
If it detects that **ghandle.rocker(ghandle.right)** is True, it means that the rocker of the handle is pushed right, and the 'right' command is sent wirelessly, and an right icon is displayed on LED dot matrix.
If it detects that **ghandle.rocker(ghandle. pressed)** is True, it means that the rocker of the handle is pressed, and the 'pressed' command is sent wirelessly, and an "X" icon is displayed on LED dot matrix.
If it does not operate to send 'stop' and clear the display.
Determine whether the button is pressed. The commands 'R', 'G', 'B', 'Y' are sent for B1 (red), B2 (green), B3 (blue), and B4 (yellow).
Code as shown below:

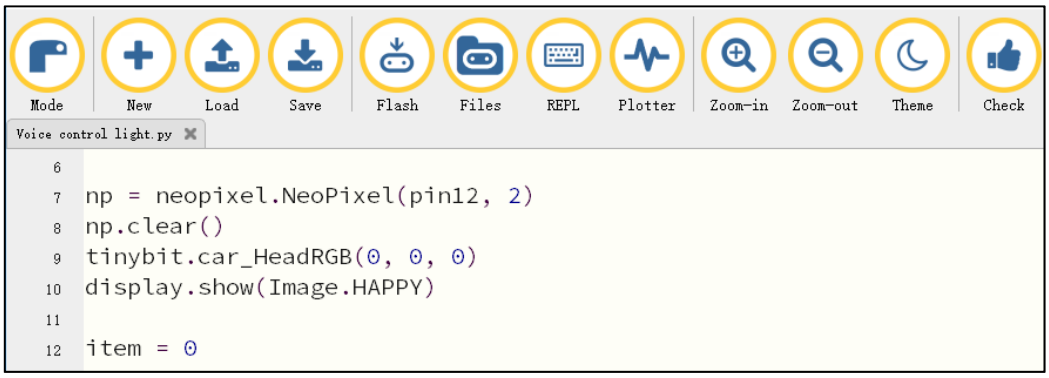
```
while True:

    if ghandle.rocker(ghandle.up):
        radio.send("up")
        display.show(Image.ARROW_N)
    elif ghandle.rocker(ghandle.down):
        radio.send("down")
        display.show(Image.ARROW_S)
    elif ghandle.rocker(ghandle.left):
        radio.send("left")
        display.show(Image.ARROW_W)
    elif ghandle.rocker(ghandle.right):
        radio.send("right")
        display.show(Image.ARROW_E)
    elif ghandle.rocker(ghandle.pressed):
        radio.send("turn_off")
        display.show(Image.NO)
    elif ghandle.B1_is_pressed():
        radio.send("R")
        display.show("R")
    elif ghandle.B2_is_pressed():
        radio.send("G")
        display.show("G")
    elif ghandle.B3_is_pressed():
        radio.send("B")
        display.show("B")
    elif ghandle.B4_is_pressed():
        radio.send("Y")
        display.show("Y")
```

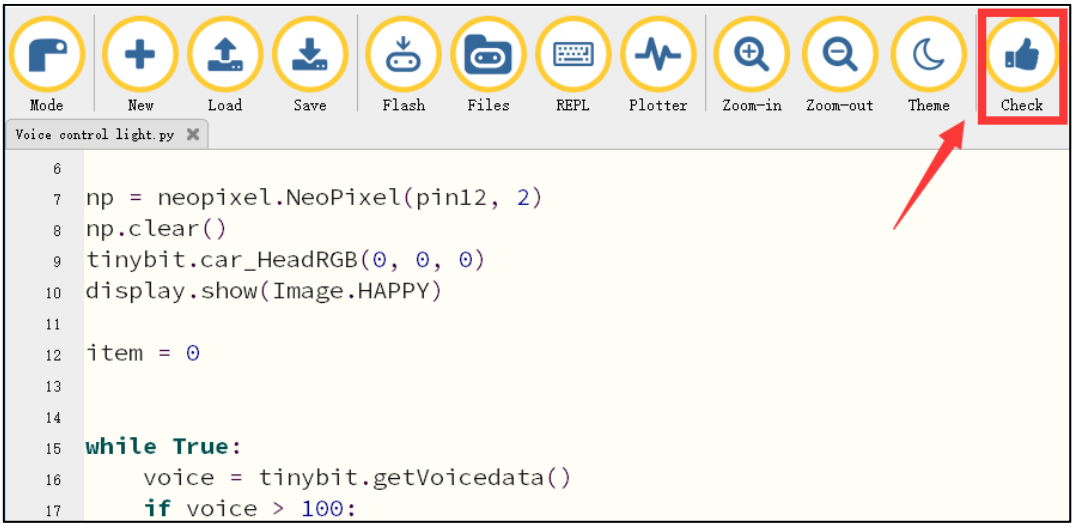
Programming and downloading：

1.You should open the Mu software, and enter the code in the edit window, , as shown below.

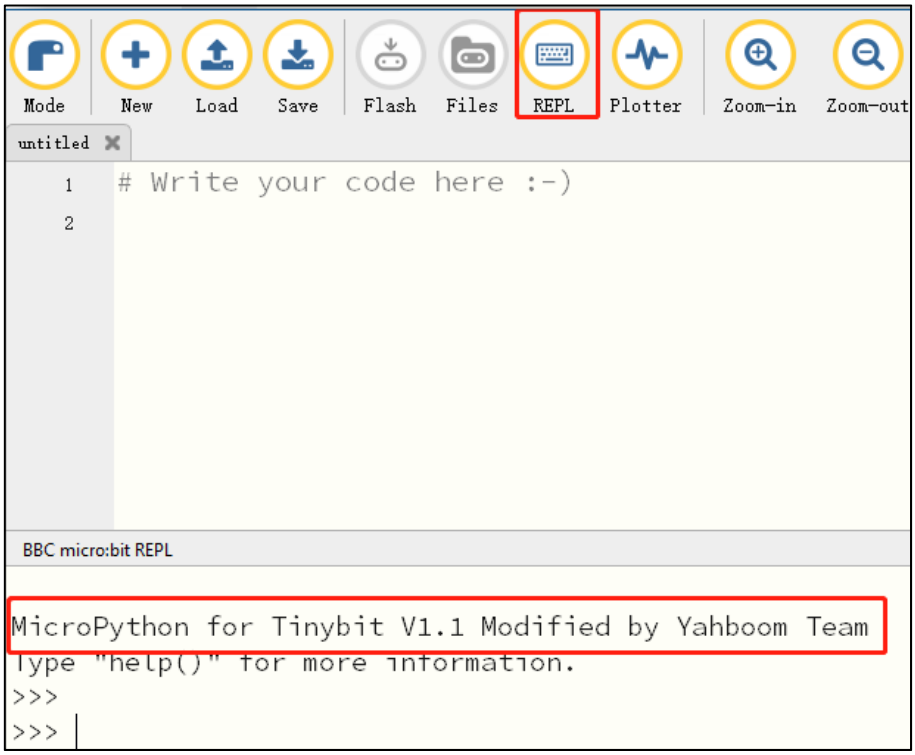
Note! All English and symbols should be entered in English, and the last line must be a space.



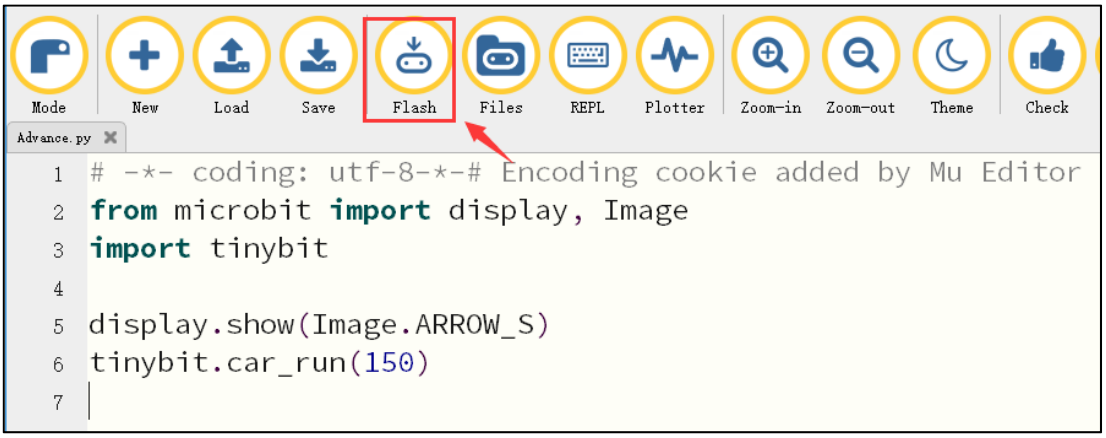
2.You can click the **"Check"** button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.



3.Click **"REPL"** button,check whether the tinybit library has been downloaded. If not, please refer to the [preparation before class]---> [Python programming]



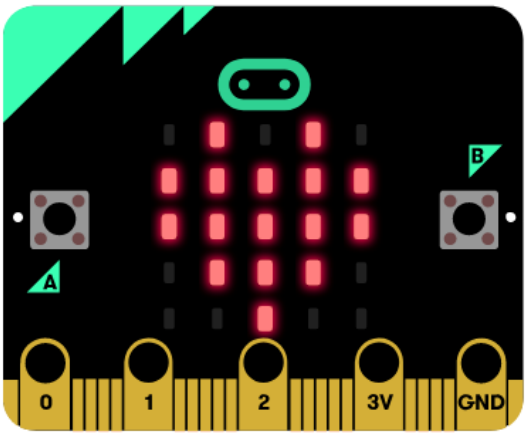
4.Click the **"Flash"** button to download the program to micro:bit board.

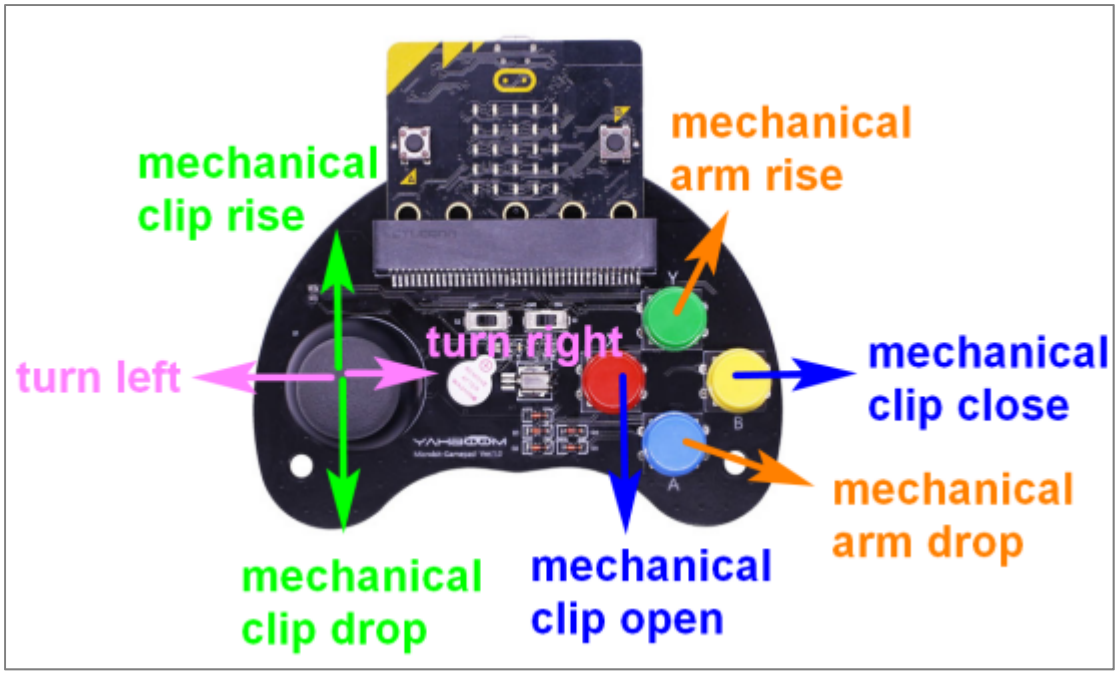


If the program is wrong or the experimental phenomenon is wrong after downloading, please confirm whether you have downloaded the Buildingbit libraryhex file we provided to the micro: bit board.
For the specific method of adding library files, please refer to 【1.Preparation before class】---【Python programming】

Experimental phenomena

After download is complete, open the power switch of car and handle. We can see micro:bit dot matrix of car will display a pattern as shown below.





!!!Tip:
Do not over-stretch at the joint of the mechanical arm to avoid unnecessary wear and affect the internal structure of the push rod.