Biped bit handle remote control

Learning goals

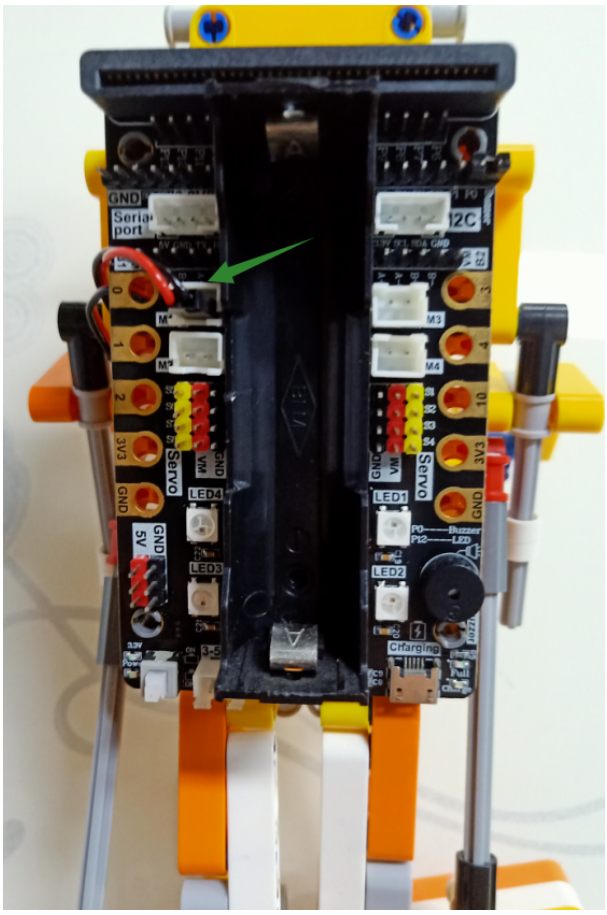
In this lesson we will learn to use the Handle to remotely control the building blocks Biped:bit .

About wiring

As shown below,

Building block motor connect to M1 interface of super:bit.

The black wiring of the motor is near the battery side.



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:

Biped:bit code

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

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

display.show(Image.HAPPY): Show a icon 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
import music

radio.on()
radio.config(group=1)
display.show(Image.HAPPY)
np = neopixel.NeoPixel(pin12, 4)
flag = 0
a = 0
```

2) Control the robot advance, back functions:

incoming = radio.receive (): Receives the wirelessly transmitted data and saves it to the “incoming” variable.

if incoming is 'up' and flag == 1 and a == 0, the robot advance ;

if incoming is 'down' and flag == 1 and a == 0, the robot back;

if incoming is 'up' and flag == 0 and a == 0, the robot advance;

if incoming is 'down' and flag == 0 and a == 0, the robot back;

.....

Code as shown below:

```
incoming = radio.receive()

if incoming == 'up' and flag == 1 and a == 0:
    superbit.motor_control(superbit.M1, -255, 0)

if incoming == 'down' and flag == 1 and a == 0:
    superbit.motor_control(superbit.M1, 255, 0)

if incoming == 'up' and flag == 0 and a == 0:
    superbit.motor_control(superbit.M1, -255, 0)

if incoming == 'down' and flag == 0 and a == 0:
    superbit.motor_control(superbit.M1, 255, 0)

if incoming == 'turn_off' and a == 0:
    superbit.motor_control(superbit.M1, 0, 0)
    music.play("C4:4")
    np.clear()
    if flag == 1:
        flag = 0
    else:
        flag += 1
if incoming == 'down':
    a = 1
if incoming == 'up':
    a = 1
if incoming == 'turn_off':
    a = 1
if incoming == 'T' and flag == 0:
    superbit.motor_control(superbit.M1, 0, 0)
    a = 0
elif incoming == 'T':
    a = 0
display.show(flag)
```

3) If incoming is ‘R’, the car headlights are red, ‘G’ is the car headlights are green, ‘B’ is the car headlights are blue, and ‘Y’ is the car headlights are yellow. Code as shown below:

```
if incoming == 'R':
    np.clear()
    np[0] = (255, 0, 0)
    np[1] = (255, 0, 0)
    np[2] = (255, 0, 0)
    np[3] = (255, 0, 0)
    np.show()

elif incoming == 'G':
    np[0] = (0, 255, 0)
    np[1] = (0, 255, 0)
    np[2] = (0, 255, 0)
    np[3] = (0, 255, 0)
    np.show()

elif incoming == 'B':
    np[0] = (0, 0, 255)
    np[1] = (0, 0, 255)
    np[2] = (0, 0, 255)
    np[3] = (0, 0, 255)
    np.show()

elif incoming == 'Y':
    np.clear()
    np[0] = (255, 255, 0)
    np[1] = (255, 255, 0)
    np[2] = (255, 255, 0)
    np[3] = (255, 255, 0)
    np.show()
```

!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 **Handle_code.py** file we provided.

1) Import the libraries microbit, ghandle, and radio that you need to use.

display.show(Image.HEART): 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 display, Image
import ghandle
import radio

display.show(Image.HEART)
radio.on()
radio.config(group=1)
```

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 arrow 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 arrow 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 arrow 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 arrow 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:

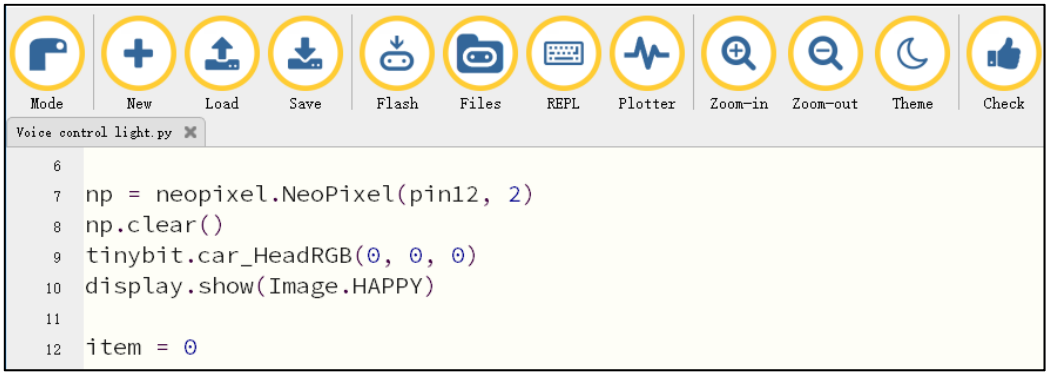
```
    if ghandle.rocker(ghandle.up):
        radio.send('up')
        display.show(Image.ARROW_N)
    elif ghandle.rocker(ghandle.down):
        radio.send('down')
        display.show(Image.ARROW_S)
    elif ghandle.rocker(ghandle.left):
        radio.send('left')
        display.show(Image.ARROW_W)
    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)
    else:
        radio.send('stop')
        display.clear()

    if ghandle.B1_is_pressed():
        radio.send('R')
        display.show("R")
    if ghandle.B2_is_pressed():
        radio.send('G')
        display.show("G")
    if ghandle.B3_is_pressed():
        radio.send('B')
        display.show("B")
    if ghandle.B4_is_pressed():
        radio.send('Y')
        display.show("Y")
```

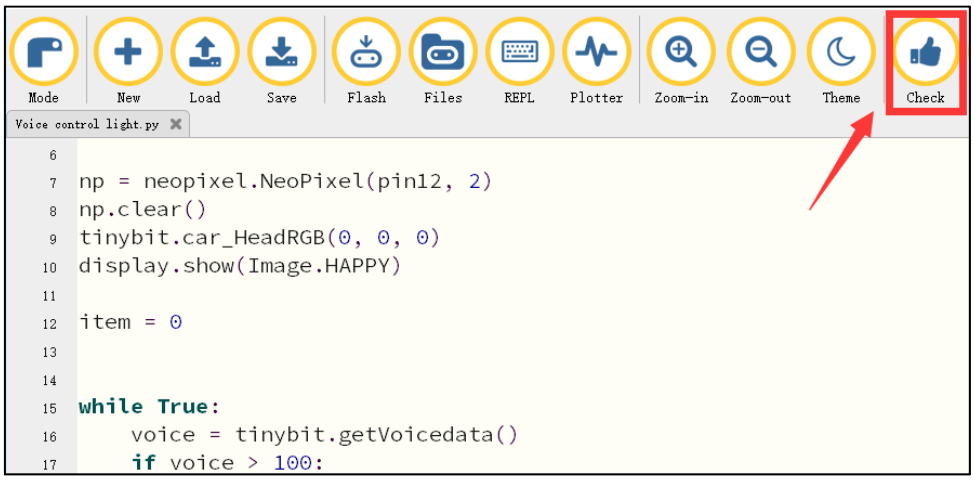
Programming and downloading

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

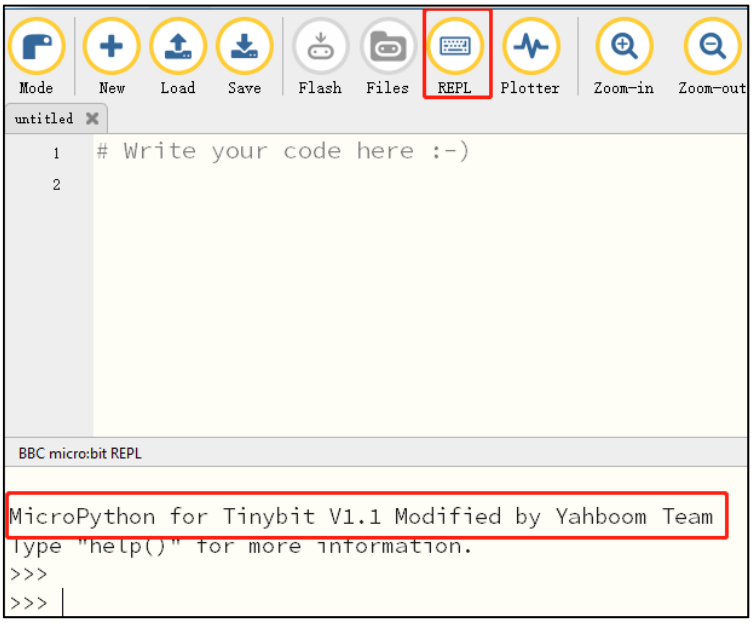
Note! All English and symbols should be entered in English, and the last line must be a space.



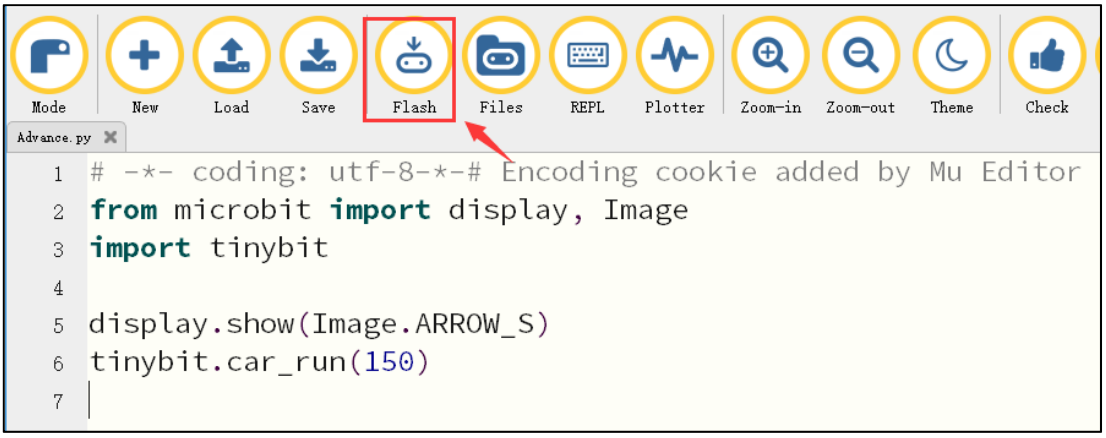
2.You can click the **“Check”** button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.



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, you can see that the micro:bit dot matrix of robot shows an “smile” pattern, as shown in Figure 1.1.
You can see that the micro:bit dot matrix of handle shows an “heart”, as shown in Figure 1.2.

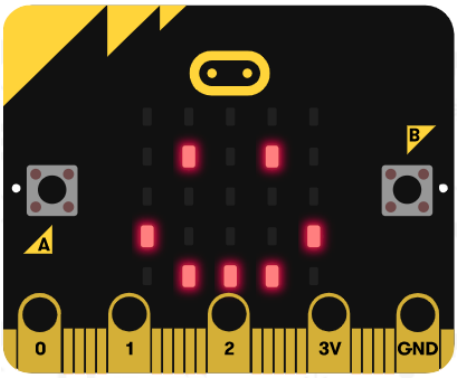


Figure 1.1

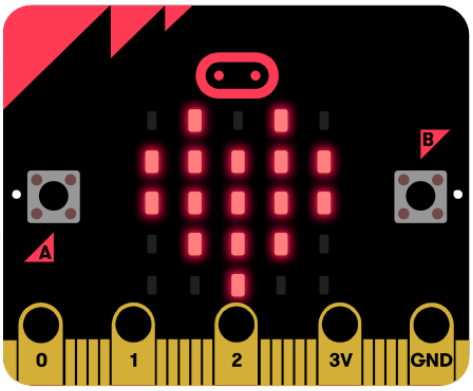
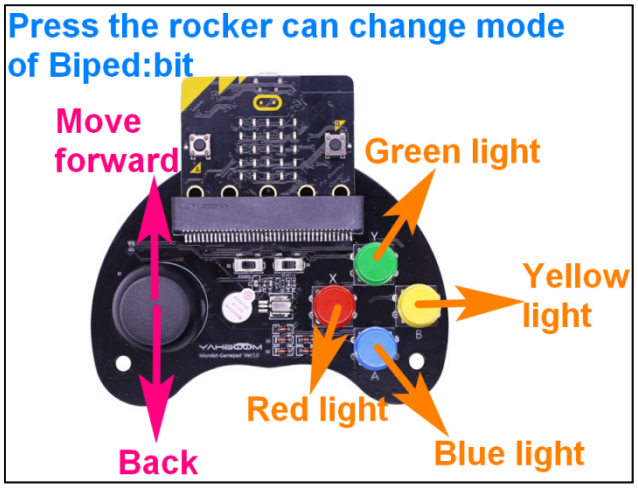


Figure 1.2



①Press the rocker to switch mode, if the mode switch successfully, the buzzer of biped:bit will sound, and micro:bit dot matrix will display the current mode(number 1 or number 2); RGB lights will off at the same time;
②The rocker pushes forward to control the biped:bit to advance;
mode 1: release and stop;
mode 2: release to continue forward, if you need to stop, press the rocker to switch mode;
③The rocker pushes back to control the biped:bit to back;
mode 1: release and stop;
mode 2: release to continue back, if you want to stop, press the rocker to

switch mode;
④Press the red button to control the RGB lights of the biped robot to become red;
⑤Press the green button to control the RGB lights of the biped robot to become green;
⑥Press the blue button to control the RGB lights of the biped robot to become blue;
⑦Press the yellow button to control the RGB lights of the biped robot to become yellow.