Biped robot code Micro:bit handle control

1.Learning goals

In this course, we mainly learn how to use handle control Biped robot.

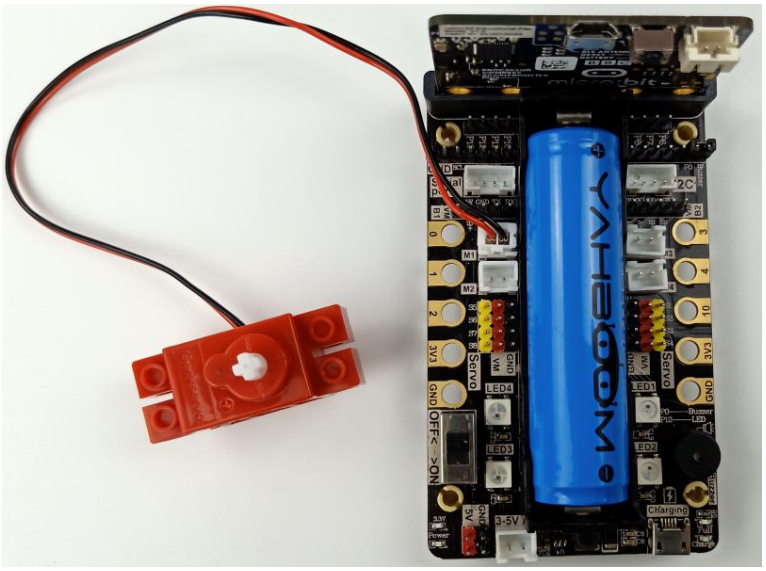
2.Building block assembly steps

For the building block construction steps, please refer to the installation manual or building block installation picture of [Assembly course]-[Biped robot].

3.Wiring of motor

The motor wiring is inserted into the M1 interface of the Super:bit expansion board, and the black wire is close to the battery side;

As shown below.



4.Code and analysis

The program for this course, please view .py file.

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

First, we need to import the library needed for this lesson from micro:bit, superbit library is dedicated to super:bit expansion board; neopixel is used to control RGB lights; radio for micro:bit wireless communication function.

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

display.show(Image.HEART)
microbit.sleep(1000)
radio.on()
radio.config(group=1)
mode = 1
```

np = neopixel.NeoPixel (pin12, 4): RGB lamp initialization settings, a total of 4 RGB lamps, connected to the P12 pin of the micro:bit board (you can check the hardware interface manual);
display.show(Image.HEART): Display the heart pattern on the micro:bit matrix;
radio.on(): Turn on the wireless function, because the wireless function consumes more power and occupies memory, so it be closed by default. We can also use radio.off() to turn off the wireless function;
microbit.sleep(1000)： Delay 1s;
radio.config(group=1): configure wireless group=1, other micro:bit devices with wireless group=1 can communicate with each other, the default is 0. The selectable group is 0~255. The set group value needs to be consistent with the setting of the handle, otherwise it cannot communicate normally;
mode = 1: Initialize the variable mode to 1.

```
while True:
    display.show(mode)
    incoming = radio.receive()
    if incoming == 'up':
        superbit.motor_control(superbit.M1, -255, 0)
    elif incoming == 'down':
        superbit.motor_control(superbit.M1, 255, 0)
```

....
In the main loop, it is judged that the robot receives the command sent by the handle, and controls the motion state of the robot and the color of the RGB lights.

incoming = radio.receive(): Receive the wirelessly transmitted data and save it in the incoming variable;
if incoming is 'up', the robot advance;
if incoming is 'down' , the robot back;

If incoming is 'R', the RGB lights become red;
If incoming is 'G', the RGB lights become green;
If incoming is 'B', the RGB lights become blue ;
If incoming is 'Y', the RGB lights become yellow.

```
if incoming == 'turn_off':
    np.clear()
    np.show()
    if mode == 1:
        mode = 2
    else:
        mode = 1
```

This code is used to switch the current mode.

! Note:

The incoming value needs to correspond to the value sent by the handle. Only the same value can receive and execute commands.

The handle program for this course, please view .py file.

```
1 # -*- coding: utf-8-*-# Encoding cook
2 from microbit import display, Image
3 import ghandle
4 import radio
```

First, we need to import the library needed for this lesson from microbit, ghanle library is dedicated to micro:bit handle; radio is used for wireless communication function of micro:bit.

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

display.show(Image.HEART)：Display heart pattern on micro:bit matrix.

radio.on()：Turn on the wireless function, because the wireless function consumes more power and occupies memory, so it be closed by default. We can also use radio.off() to turn off the wireless function;

radio.config(group=1): configure wireless group=1, other micro:bit devices with wireless group=1 can communicate with each other, the default is 0. The selectable group is 0~255. The set group value needs to be consistent with the setting of the handle, otherwise it cannot communicate normally;

```
while True:
    if ghandle.rocker(ghandle.up):
        display.show(Image.ARROW_N)
        radio.send('up')
    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.noState):
        radio.send('stop')
        display.clear()
```

if ghandle.rocker (ghandle.up) is True, it means that the rocker of the handle is pushed up, the wireless send the ‘up’ command and display an up icon on micro:bit matrix;
if the ghandle.rocker (ghandle.down) is True, it means that the rocker of the handle is pushed up, the wireless send the ‘down’ command and display a down icon on micro:bit matrix;
if the detection of ghandle.rocker(ghandle.left) is True, it means that the rocker of the handle is pushed left, the wireless send the ‘left’ command and display an left icon on micro:bit matrix;
if ghandle.rocker(ghandle.right) is detected as True, it means that the rocker of the handle is pushed right, the wireless send the ‘right’ command and display an right icon on micro:bit matrix;
if the detection of ghandle.rocker(ghandle.pressed) is True, it means that the rokcer of the handle is pressed, the wireless send ‘pressed’ command, and display ‘X’ icon on micro:bit matrix;
If there is no operation on the handle, send ‘stop’ and clear the display;

Special treatment:

```
    if ghandle.rocker(ghandle.pressed):
        if rocker_key == 0:
            rocker_key = 1
            radio.send('turn_off')
        display.show(Image.NO)
    else:
        rocker_key = 0
```

When the rocker is pressed, we use a variable rocker_key to ensure that the turn_off command is sent only once. Corresponding to the robot program, change the mode value once.

```
    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")
```

if ghandle.B1_is_pressed(): is True, it means that the B1(red button) is pressed, the wireless send the 'R' command and display "R" on micro:bit matrix;
if ghandle.B2_is_pressed(): is True, it means that the B2(green button) is pressed, the wireless send the 'G' command and display "G" on micro:bit matrix;
if ghandle.B3_is_pressed(): is True, it means that the B3(blue button) is pressed, the wireless send the 'B' command and display "B" on micro:bit matrix;
if ghandle.B4_is_pressed(): is True, it means that the B4(yellow button) is pressed, the wireless send the 'Y' command and display "Y" on micro:bit matrix;

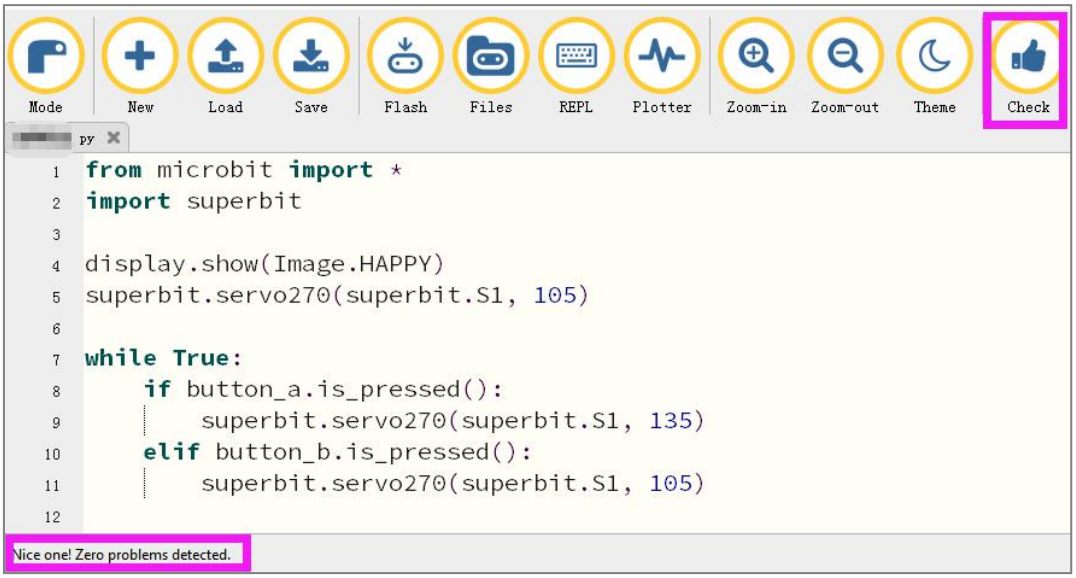
5.Writing and download code

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, use the Tab key (tab key) to indent and the last line must be a space.

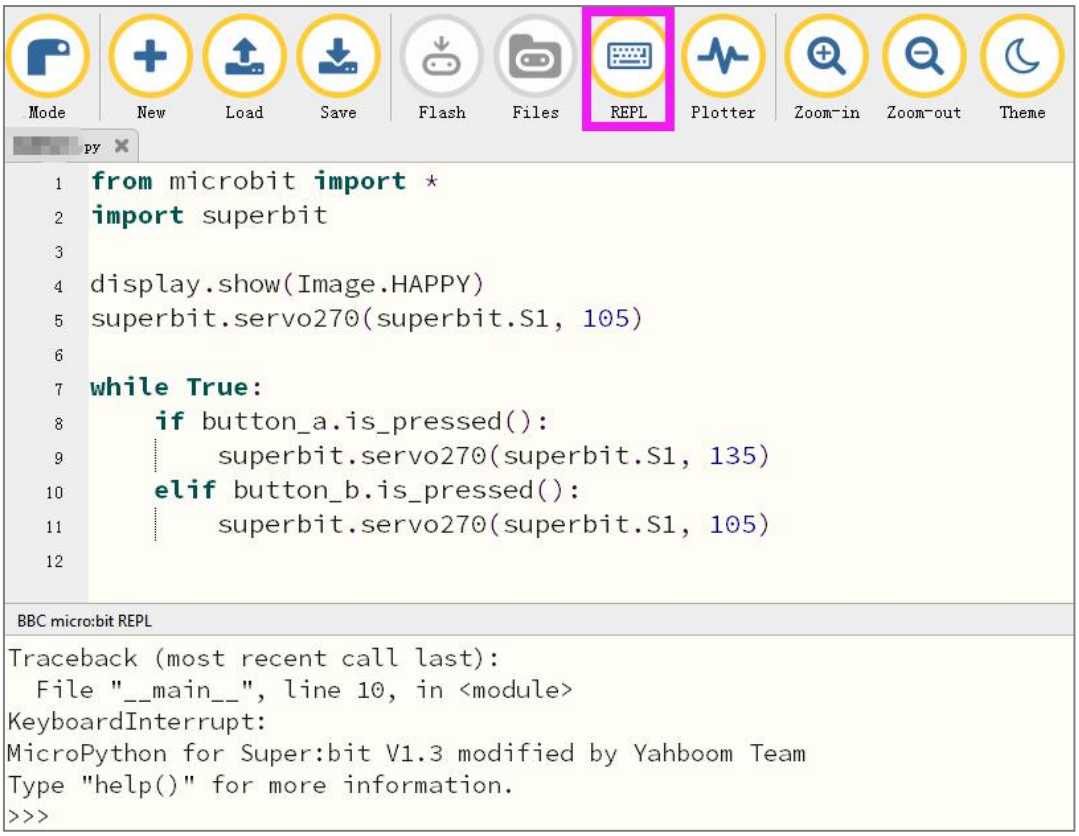
2.You can click the "Check" button to check if our code has an error.

If a cursor or underline appears on a line, it indicates a syntax error, please check and modify. If there is no error in the program, the bottom left of the interface will prompt that there is no problem in detection.

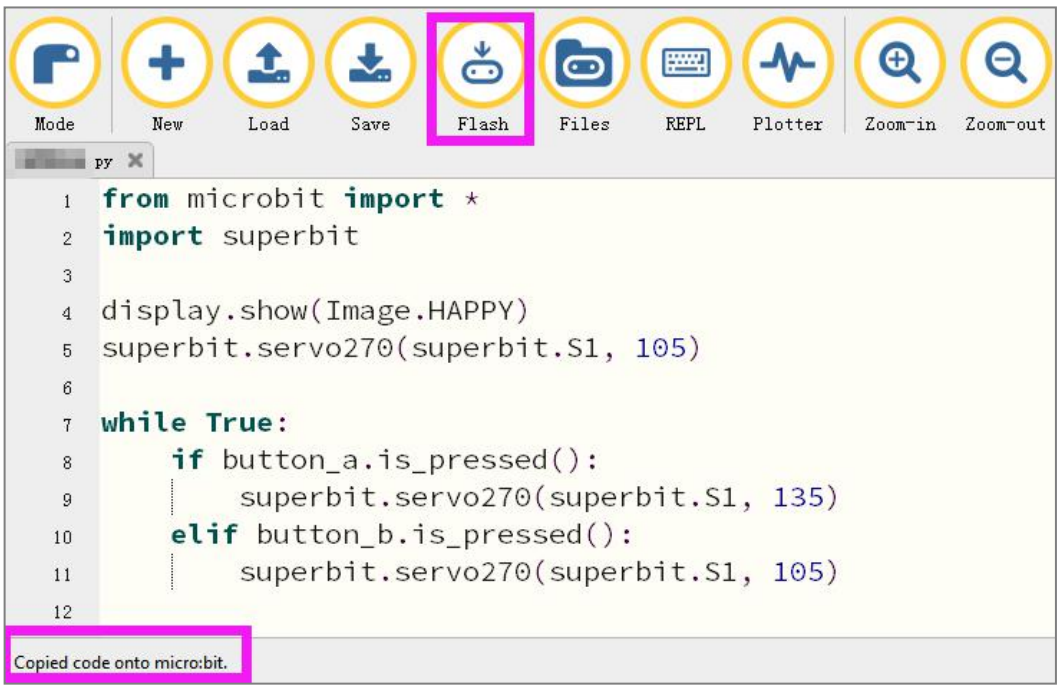


3. Click the ‘REPL’ button to check whether the Superbit library has been downloaded.

If not, please refer to [Preparation before class] --> [2.4 Python Programming Guide] .



4. After the program is written, use a micro USB cable to connect the computer and the micro:bit board. Please click the ‘Flash’ button to download the program to the micro:bit motherboard (You need to click the ‘REPL’ button again to close the function of importing library files before you download the program).



5.If the download failed, please confirm whether the micro:bit is connected to the computer through the micro USB data cable, and confirm whether the Super:bit Python library has been imported.

6.Experimental phenomena

We need to download the Biped robot code into the micro:bit board of the Biped robot. Open the power switch of the Biped robot, we can see a heart pattern displayed on the micro:bit dot matrix;

We need to download the Handle code into the micro:bit board of the handle. Open the power switch of the handle, we can see that the micro: bit dot matrix will initially display a heart pattern, and then display an "X" pattern, indicating that the handle is in the default(no data is sent).

They will automatically pairing, then, we can start remote control the Biped robot by handle.

The handle functions are shown below.

Rocker control

After the handle and the Freestyle are paired successfully, we can see that the

Freestyle displays the number 1 on the micro:bit dot matrix, which indicating it in mode 1 at this time.
In the case of mode 1:

- The rocker is pushed forward to control the Freestyle movement, and it stops when you release your hand;
- Press the red button to light up the red RGB light;
- Press the green button to light the green RGB light;
- Press the yellow button to light the yellow RGB light;
- Press the blue button to light up the blue RGB light.

We can press the rocker to switch to mode 2. At this time, we can see the number 2 on the micro: bit dot matrix of the Freestyle, indicating that it in mode 2.
In the case of mode 2:

- Push the rocker forward to control the Freestyle,movement, and keep advance when you release your hand;
- Press the red button to turn on the red RGB light to stop the Freestyle;
- Press the green button to light the green RGB light;
- Press the yellow button to light the yellow RGB light;
- Press the blue button to light up the blue RGB light.

When we press the rocker each time, it will switch Mode 1 and Mode 2, and the RGB light will turn off.