Proficient carrier Microbit handle control

1.Learning goals

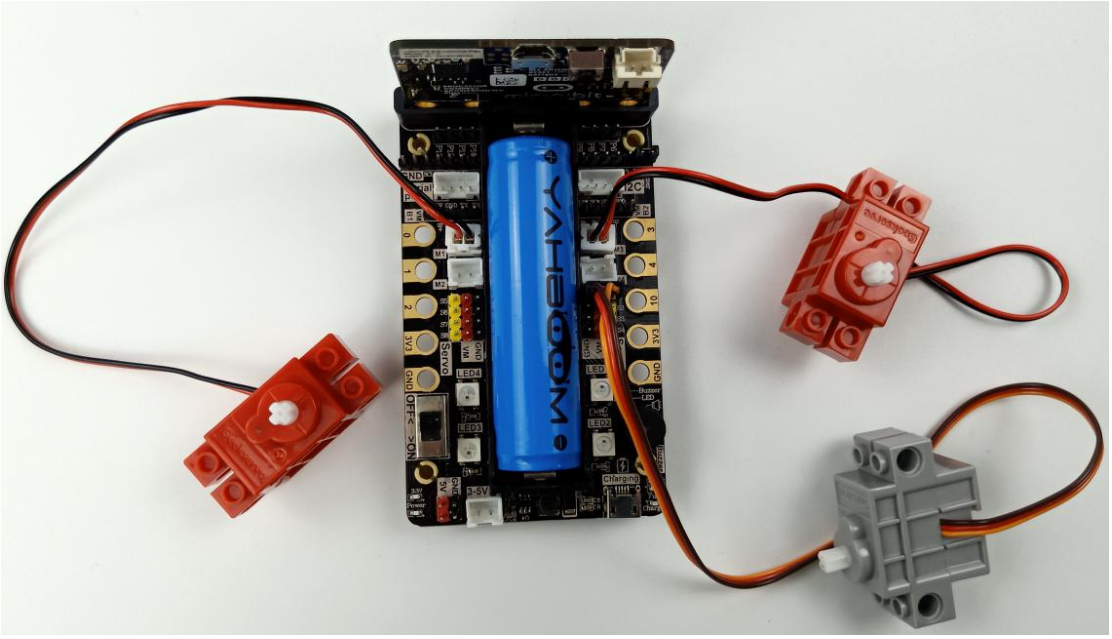
In this course, we mainly learn how to use handle control Proficient carrier.

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]-[Proficient carrier].

3.Wiring of motor and servo

The motor wiring on the left side of the car is inserted into the M1 interface of the Super:bit expansion board, and the black wire is close to the battery side;
The motor wiring on the right side of the car is inserted into the M3 interface of the Super:bit expansion board, and the black wire is close to the battery side;
Building block servo insert into the Super: bit expansion board S1 interface, and the orange wiring connect the yellow pin of S1.
As shown below:



Note:

For the first course related to building block servo, we need to remove the gear on the servo and upload the program of this course to micro: bit. Then, turn on the power switch of the Super:bit expansion board and wait for the building block servo turn to the initial position. Next, we can turn off the power, and adjust the shovel angle of the car parallel to the ground. Finally, install the servo. (If you have used programs related to clip robot before, you can skip this step)

4.Code and analysis

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

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

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.

```
Red = (255, 0, 0)
Orange = (255, 165, 0)
Yellow = (255, 255, 0)
Green = (0, 255, 0)
Blue = (0, 0, 255)
Violet = (148, 0, 211)
White = (255, 255, 255)
color_lib = {
    'Red': Red, 'Orange': Orange, 'Yellow': Yellow, 'Green': Green,
    'Blue': Blue, 'Violet': Violet, 'White': White}
def RGBLight_more_show(first, num, color):
    global np

    np.clear()
    for i in range(first, first + num):
        np[i] = color_lib[color]
    np.show()
```

This program is used to define RGB lights of different colors and define the function RGBLight_more_show to control the color of RGB lights.
This function will be called in the main loop.

```
np = neopixel.NeoPixel(pin12, 4)
display.show(Image.HEART)
superbit.servo270(superbit.S1, 120)
radio.on()
radio.config(group=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;

superbit.servo270 (superbit.S1, 120): Initialize to turn the servo to 120° (shovel lays flat).

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:
    incoming = radio.receive()
    if incoming == 'up':
        superbit.motor_control(superbit.M1, 255, 0)
        superbit.motor_control(superbit.M3, 255, 0)
```

....

In the main loop, it is judged that the car receives the command sent by the handle, and controls the motion state of the car 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 car move forward;
if incoming is 'down', the car move backward;
if incoming is 'left' , the car spin left;
if incoming is 'right' , the car spin right;
if incoming is 'stop' , the car stop.

If incoming is 'R', the RGB lights become red and the shovel unload;
If incoming is 'G', the RGB lights become green and the shovel lay flat;
If incoming is 'B', the RGB lights become blue;
If incoming is 'Y', the RGB lights become yellow and the shovel raise.
! 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.

```
# -*- coding: utf-8-*-# Encoding cook
from microbit import display, Image
import ghandle
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.

```
display.show(Image.HEART)
radio.on()
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):
        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.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;

```
    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;

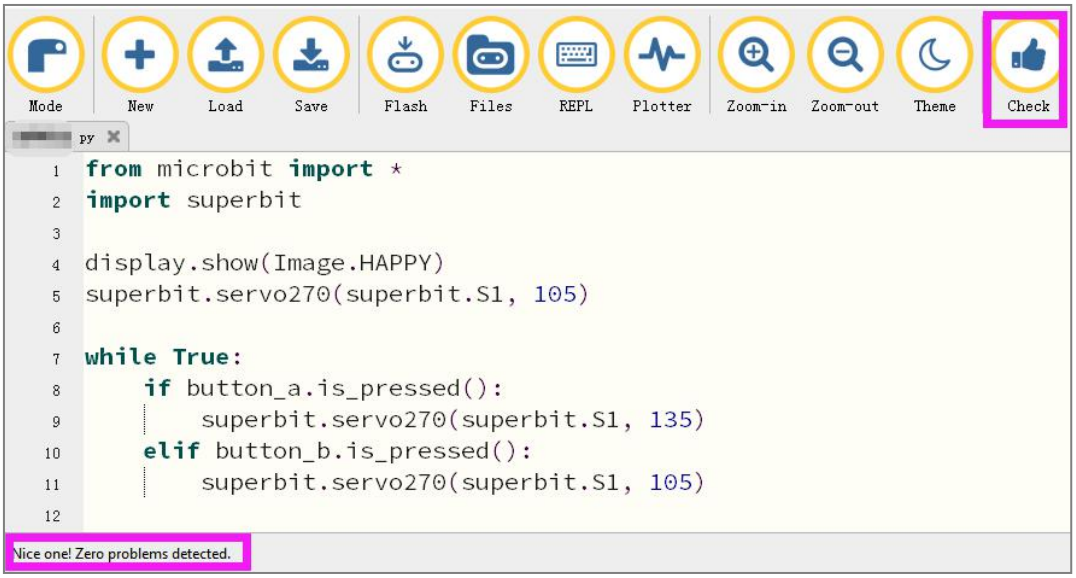
5.Writing and download code

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

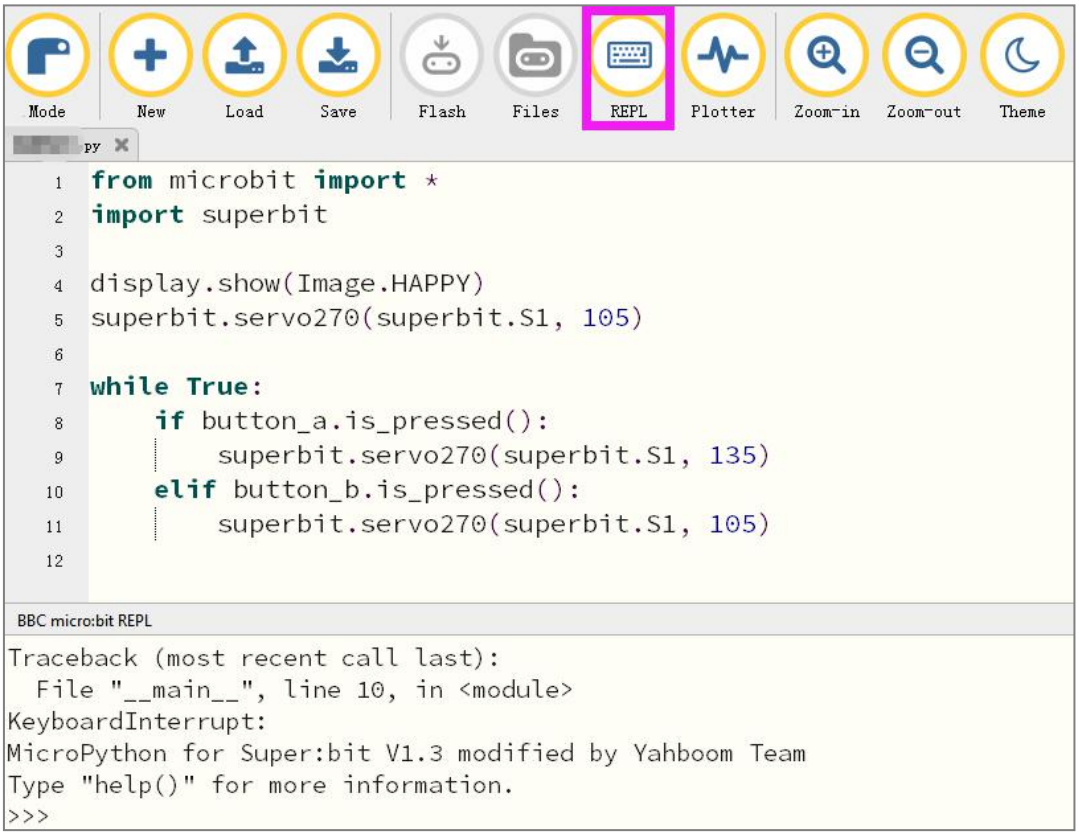
Note! All English and symbols should be entered in English, use the Tab key (tab key) to indent and the last line must be a space.

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

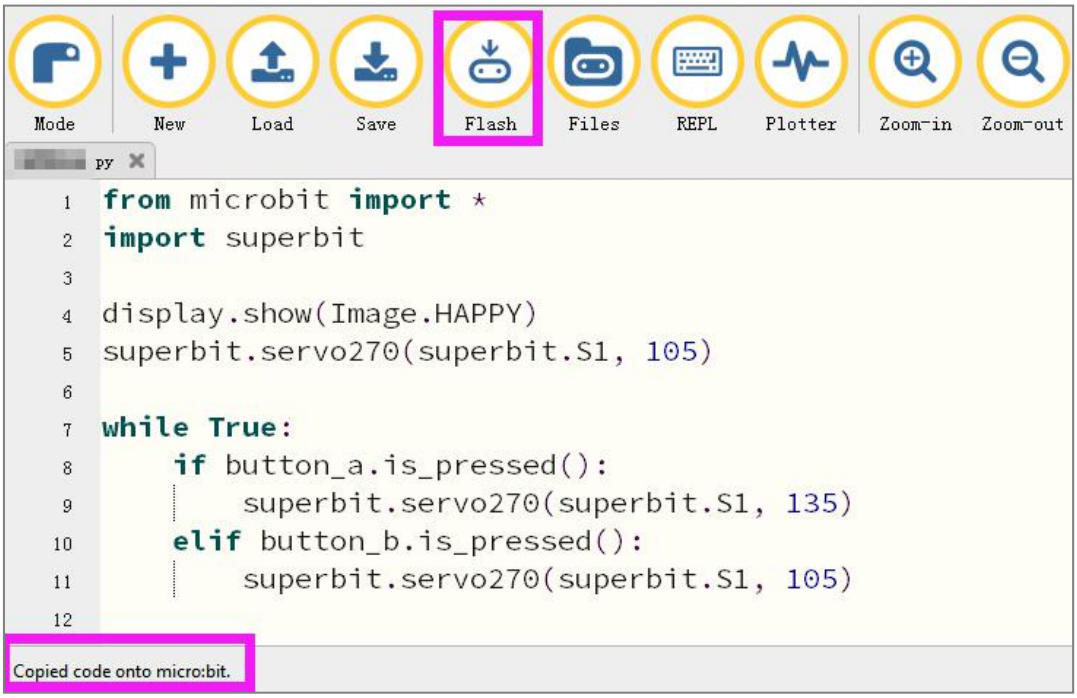
If a cursor or underline appears on a line, it indicates a syntax error, please check and modify. If there is no error in the program, the bottom left of the interface will prompt that there is no problem in detection.



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).

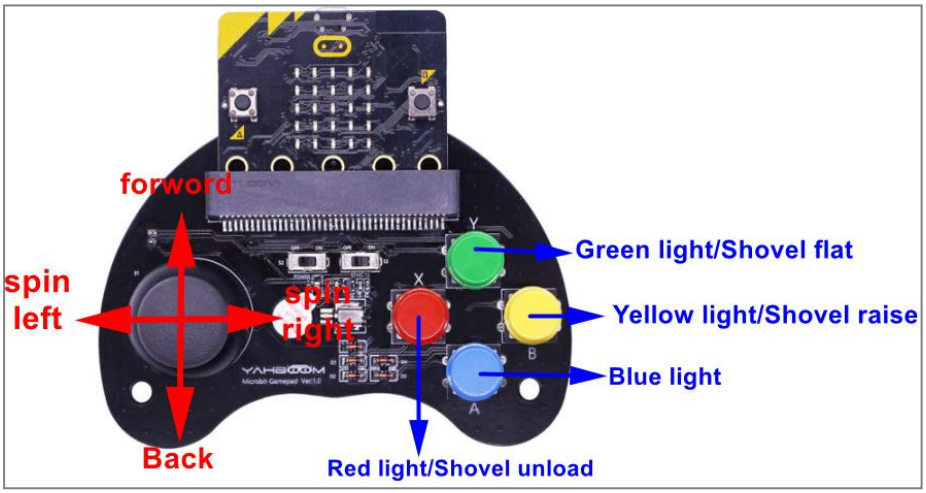


5.If the download failed, please confirm whether the micro:bit is connected to the computer through the micro USB data cable, and confirm whether the Super:bit Python library has been imported.

6.Experimental phenomena

We need to download the Proficient carrier code into the micro:bit board of the Proficient carrier. Open the power switch of the car, 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 Proficient carrier by handle.

The handle functions are shown below.