

Microbit handle control

Microbit handle control

1. Learning objectives
2. Building blocks
3. Servo wiring
4. Code analysis
 - 4.1 Security automatic door
 - 4.2 Handle
5. Write and download the program
6. Experimental phenomenon

1. Learning objectives

In this course, we mainly learn how to use Python programming to control the security automatic door with the micro:bit handle.

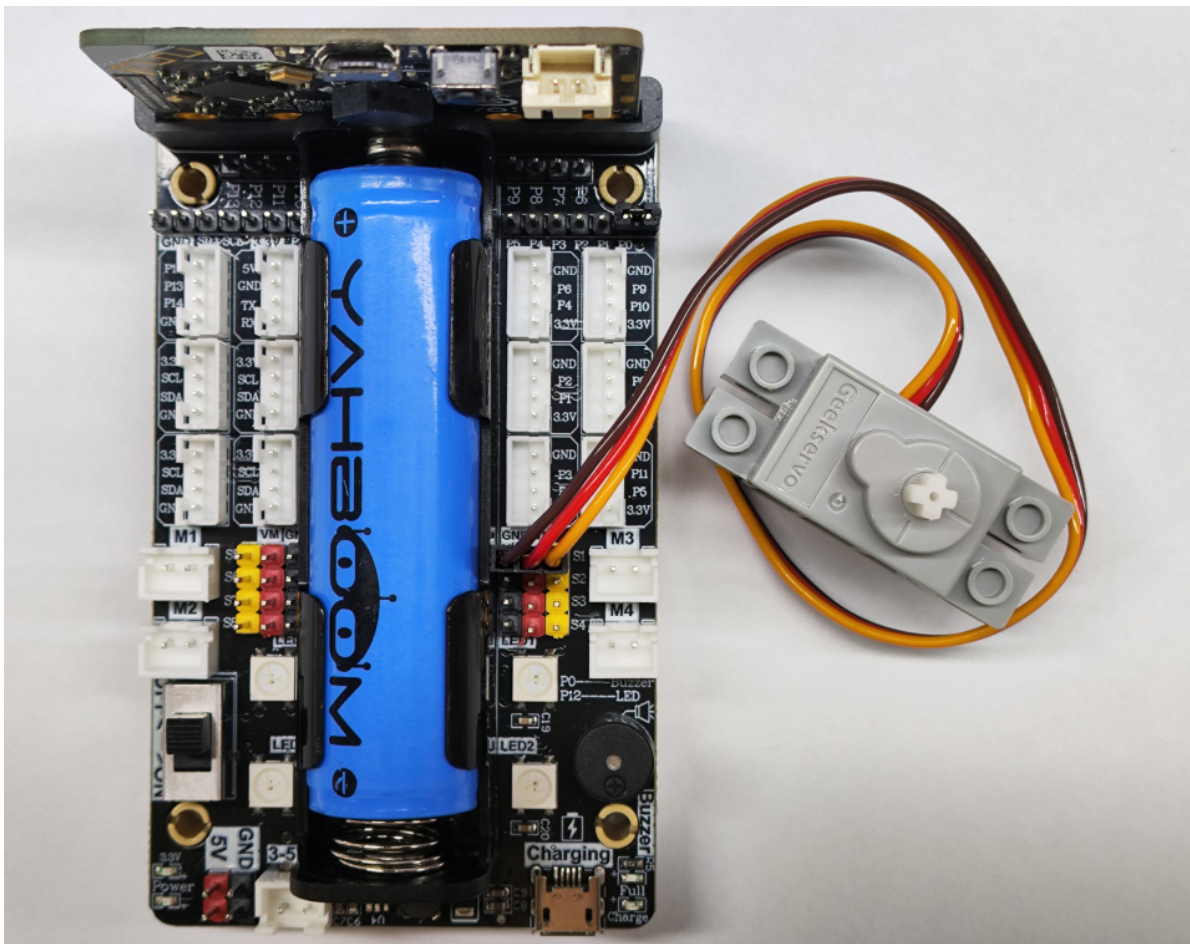
2. Building blocks

For the building blocks steps, please refer to the installation drawings of [Assembly course]-[Lifting platform] or the building blocks installation album in the materials.

3. Servo wiring

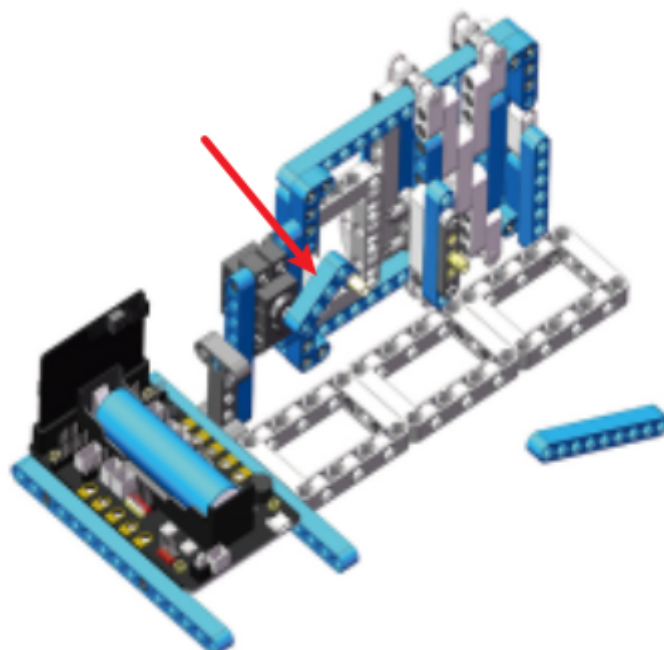
Insert the building blocks servo wiring into the Super:bit expansion board S1 interface, and the servo orange wiring into the yellow pin of S1.

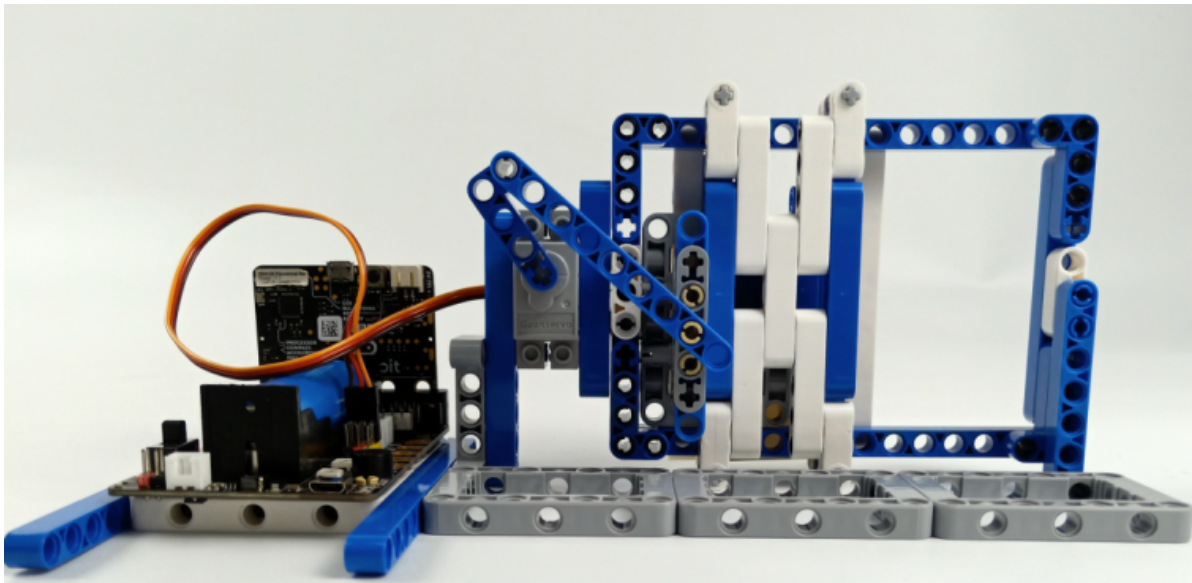
As shown in the figure below:



! Notes:

When taking the course related to the building block servo for the first time, we need to remove the blue building block installed on the servo and upload the program of this course to the micro:bit; then turn on the power switch of the Super:bit expansion board and wait for the building block servo to turn to the initial position; then, we can turn off the power, adjust the security automatic door to the open state, as shown in the figure below, and then install the blue building block. (If you have used the security automatic door and servo related program before, you can skip this step)





4. Code analysis

4.1 Security automatic door

For the program, please refer to the **Auto-door code.py** file.

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

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

```
radio.on()  
radio.config(group=1)  
display.show(Image.HEART)  
np = neopixel.NeoPixel(pin12, 4)  
superbit.servo270(superbit.S1, 90)
```

display.show(Image.HEART): Display the heart pattern on the microbit dot matrix;
np = neopixel.NeoPixel(pin12, 4): RGB light initialization settings, there are 4 RGB lights, connected to the P12 pin of the microbit motherboard (you can check the hardware interface manual);
superbit.servo270(superbit.S1, 90): Initialize the servo to 90 degrees;

radio.on(): Turn on the wireless function. Because the wireless function consumes more power and occupies more memory, it is turned off 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 microbit devices with wireless group=1 can communicate with each other. The default is 0, and the selectable group is 0~255. The set group value needs to be consistent with the handle setting, otherwise it will not communicate normally;

superbit.servo270(superbit.S1, 90): Initialize the servo to rotate to 90°, that is, open the door.

```

while True:
    incoming = radio.receive()
    if incoming == 'up':
        superbit.servo270(superbit.S1, 0)
    elif incoming == 'down':
        superbit.servo270(superbit.S1, 90)
    ...

```

In the main loop, determine whether the Face Changing King receives the command sent by the handle, and control the movement state of the Face Changing King and the color of the RGB light.

`incoming = radio.receive()`: Receive the data transmitted wirelessly and save it to the `incoming` variable; if `incoming` is 'up', open the Face Changing King mask, and 'down' close the Face Changing King mask; if `incoming` is 'left', display the duck pattern on the dot matrix, and 'right' display the cow pattern on the dot matrix;

If `incoming` is 'R', the body RGB lights up red, 'G' makes the body RGB light up green, 'B' makes the body RGB light up blue, and 'Y' makes the body RGB light up yellow.

! Note:

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

4.2 Handle

Please refer to the **Handle code.py** file for the program.

```

from microbit import display, Image
import ghandle
import radio

```

First, import the libraries needed for this lesson from `microbit`: the `ghandle` library is dedicated to the `micro:bit` hand; `radio` is used for the wireless communication function of `micro:bit`.

```

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

```

`display.show(Image.HEART)`: Display a heart pattern on the `microbit` dot matrix;

`radio.on()`: Turn on the wireless function. Because the wireless function consumes more power and occupies more memory, it is turned off 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 `microbit` devices with wireless `group=1` can communicate with each other. The default is 0, and the selectable group is 0~255. The set group value needs to be consistent with the handle setting, otherwise it will not 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 joystick of the handle is pushed up, so that the wireless sends the 'up' command and displays an upward icon;

If `ghandle.rocker(ghandle.down)` is True, it means that the joystick of the handle is pushed down, so that the wireless sends the 'down' command and displays a downward icon;

If `ghandle.rocker(ghandle.left)` is True, it means that the joystick of the handle is pushed to the left, so that the wireless sends the 'left' command and displays a left icon;

If `ghandle.rocker(ghandle.right)` is True, it means that the joystick of the handle is pushed to the right, so that the wireless sends the 'right' command and displays a right icon;

If `ghandle.rocker(ghandle.pressed)` is True, it means that the joystick of the handle is pressed, so the wireless sends the 'pressed' command and displays the 'X' icon;

If the remote control has no operation, it sends 'stop' and clears the display;

```

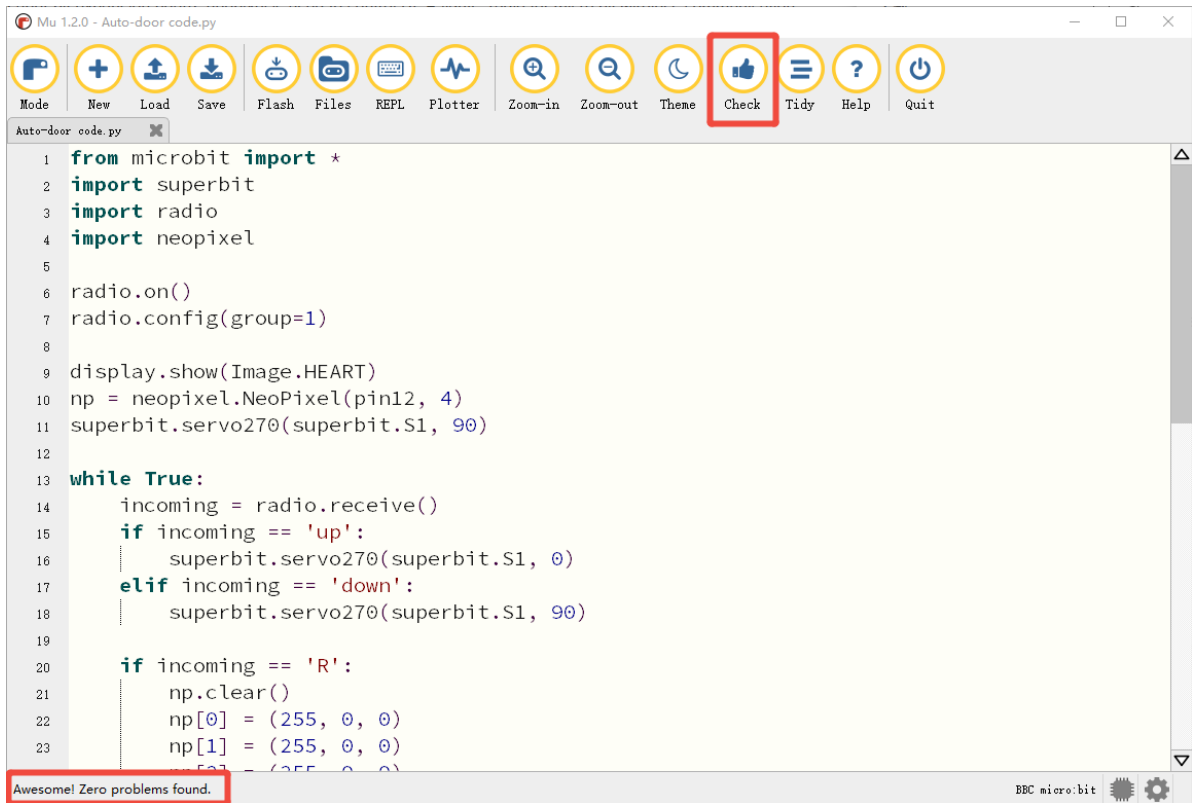
if ghandle.B1_is_pressed():
radio.send('R')
display.show("R")
if ghandle.B2_is_pressed():
radio.send('G')
display.show("G")
if ghandle.B3_is_pressed():
radio.send('B')
display.show("B")
if ghandle.B4_is_pressed():
radio.send('Y')
display.show("Y")

```

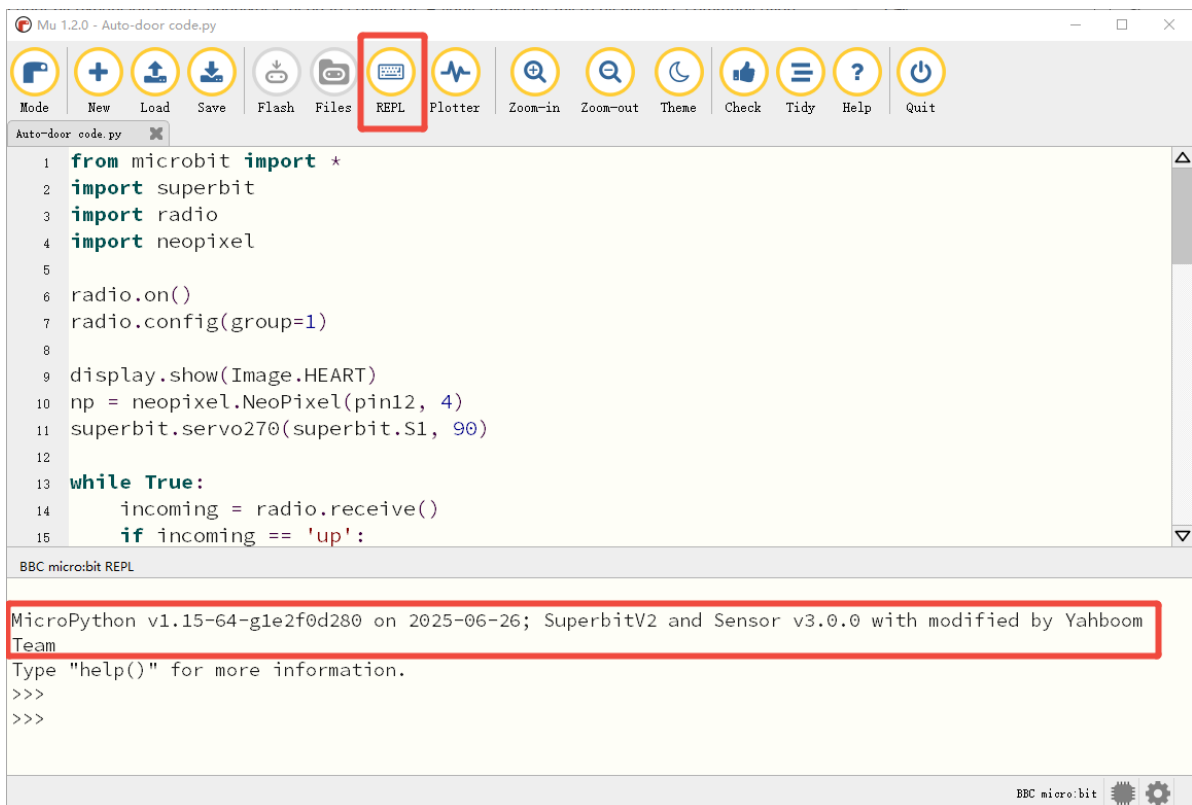
Next, detect the button and send R, 'G', 'B', 'Y' commands corresponding to B1 (red), B2 (green), B3 (blue), and B4 (yellow).

5. Write and download the program

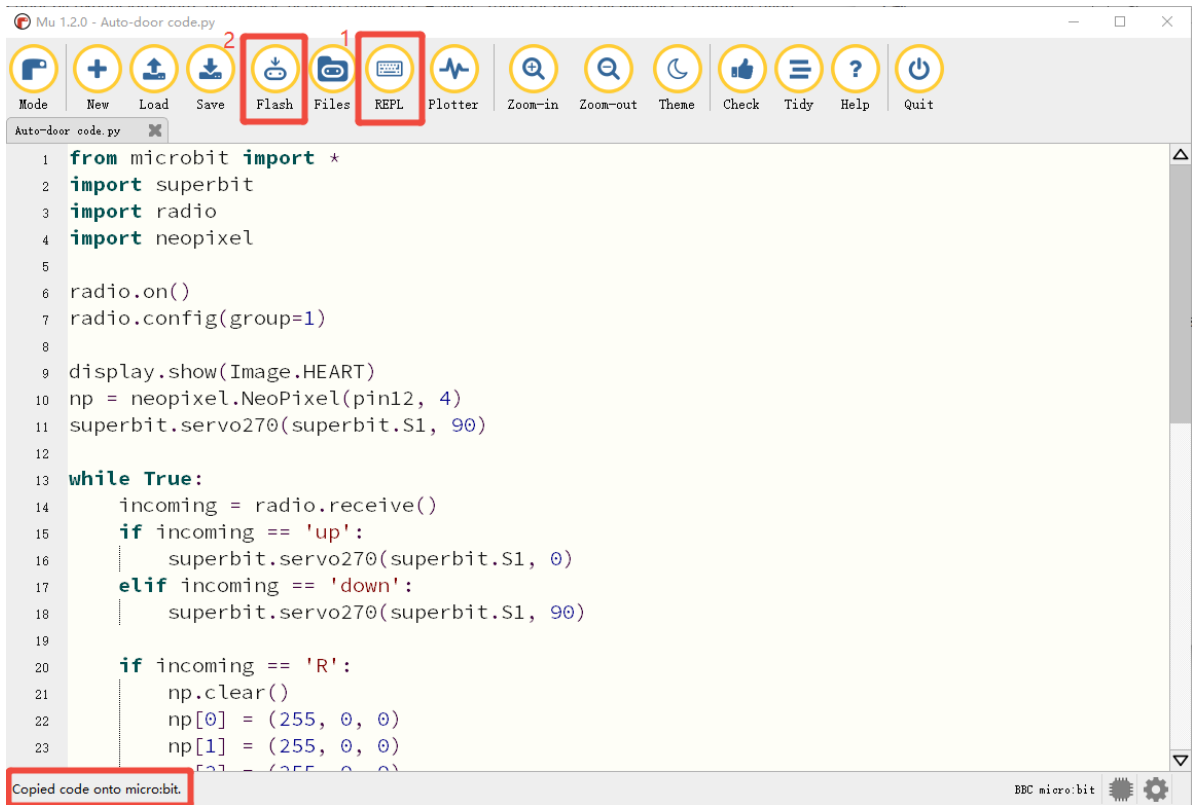
1. Open the Mu software and enter the code in the editing window. **Note! All English and symbols should be entered in English, use the Tab key for indentation, and the last line ends with a blank program.**
2. Click the thumb 'Check' button to check if there are any errors in our code. If a cursor or underline appears in a line, it means a syntax error. Please check and modify it. If there is no error, the lower left corner will prompt that there is no problem with the 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, connect the computer and microbit mainboard with a microUSB data cable, please click the 'Flash' button to download the program to the micro:bit mainboard. **(You need to click the 'REPL' button again to turn off the import library file function before you can download the program normally).**



5. If the download fails, please confirm whether the microbit is connected to the computer normally via the microUSB data cable and the Superbit Python library has been imported.

6. Experimental phenomenon

We need to download the **Auto-door code.py** file to the micro:bit mainboard of the security automatic door, turn on the power switch of the security automatic door, and we can see a smiley face pattern displayed on the micro:bit dot matrix;

Download the **Handle code.py** file to the micro:bit mainboard of the handle, turn on the power switch of the handle, and we can see that the micro:bit dot matrix will be initialized to display a heart pattern, and then an "X" pattern will be displayed, indicating that the handle is in the default state and no data is sent.

The two will automatically complete the pairing, and then we can start remote control of the security automatic door.

The handle functions are as follows.

