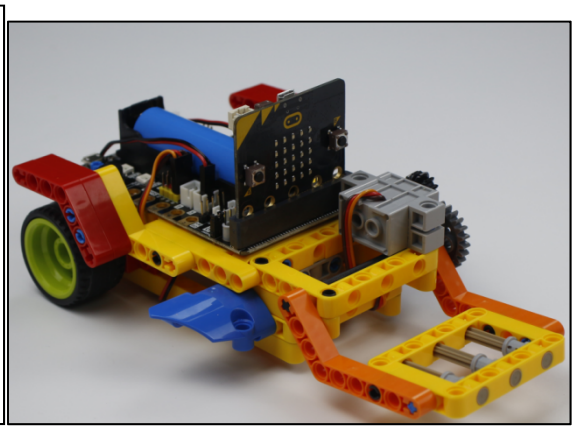
Proficient Carrier handle remote control



In this lesson we will learn to use the Handle to remotely control the building blocks Proficient Carrier.

About wiring

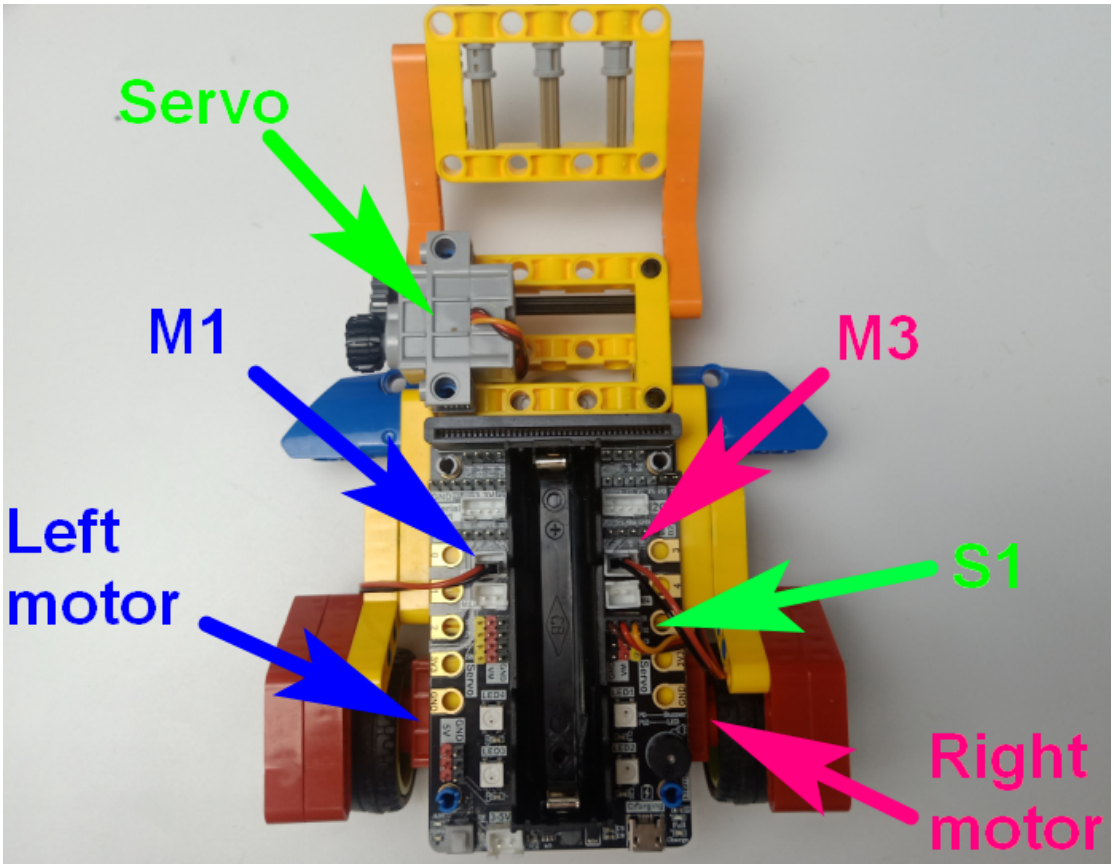
As shown below,

Left motor connect to M1 interface of super:bit.

Right motor connect to M3 interface of super:bit.

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

Servo connect to S1 interface of super:bit.



!!!Note:When using the program related to the building block servo for the first time.

First, we need to remove the gear on the building block servo.Then, download the program to micro:bit. Next, turn on the power of the Super:bit expansion board and wait for the building block servo to turn to the initial position and turn off the power. Finally, we need to open the trolley clip to the widest point, and then install the gear on the building block servo.

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.

About code:

Proficient Carrier code:

Please use the MU software to open the **Proficient Carrier code.py** file we provided.

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

display.show(Image.HEART): Show a smile pattern 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 microbit
import superbit
import radio
import neopixel
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()
np = neopixel.NeoPixel(pin12, 4)
display.show(Image.HEART)
radio.on()
radio.config(group=1)
angle = 240
flag = 0
```

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 car advance; 'down' the car back; 'left' the car turn left; 'right' makes the car turn right; And 'stop' makes the car stop.
Code as shown below:

```
    incoming = radio.receive()
    if incoming == 'up':
        superbit.motor_control(superbit.M1, 255, 0)
        superbit.motor_control(superbit.M3, 255, 0)
    elif incoming == 'down':
        superbit.motor_control(superbit.M1, -255, 0)
        superbit.motor_control(superbit.M3, -255, 0)
    elif incoming == 'left':
        superbit.motor_control(superbit.M1, 0, 0)
        superbit.motor_control(superbit.M3, 255, 0)
    elif incoming == 'right':
        superbit.motor_control(superbit.M1, 255, 0)
        superbit.motor_control(superbit.M3, 0, 0)
    elif incoming == 'stop':
        superbit.motor_control(superbit.M1, 0, 0)
        superbit.motor_control(superbit.M3, 0, 0)
```

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

```
elif incoming == 'R':
    RGBLight_more_show(0, 4, 'Red')
    superbit.servo270(superbit.S1, 60)
elif incoming == 'G':
    RGBLight_more_show(0, 4, 'Green')
    superbit.servo270(superbit.S1, 120)
elif incoming == 'B':
    RGBLight_more_show(0, 4, 'Blue')
elif incoming == 'Y':
    RGBLight_more_show(0, 4, 'Yellow')
    superbit.servo270(superbit.S1, 180)
```

!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(0): Show number 0 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:

```
# -*- coding: utf-8-*-# Encoding cook
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 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:

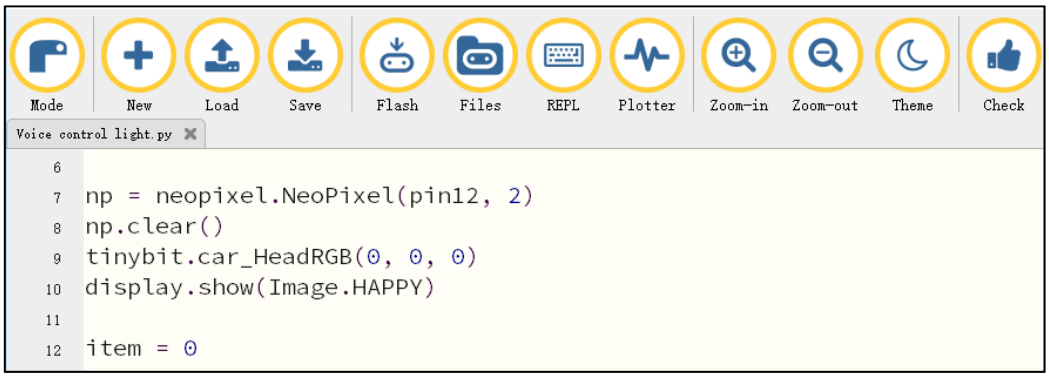
```
    if ghandle.rocker(ghandle.up):
        radio.send('up')
        display.show(Image.ARROW_N)
    elif ghandle.rocker(ghandle.down):
        radio.send('down')
        display.show(Image.ARROW_S)
    elif ghandle.rocker(ghandle.left):
        radio.send('left')
        display.show(Image.ARROW_W)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

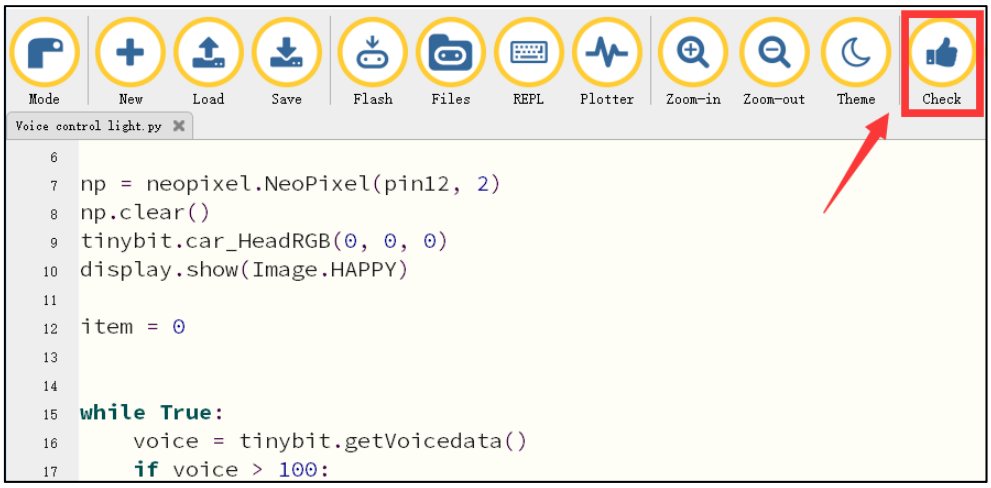
Programming and downloading

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

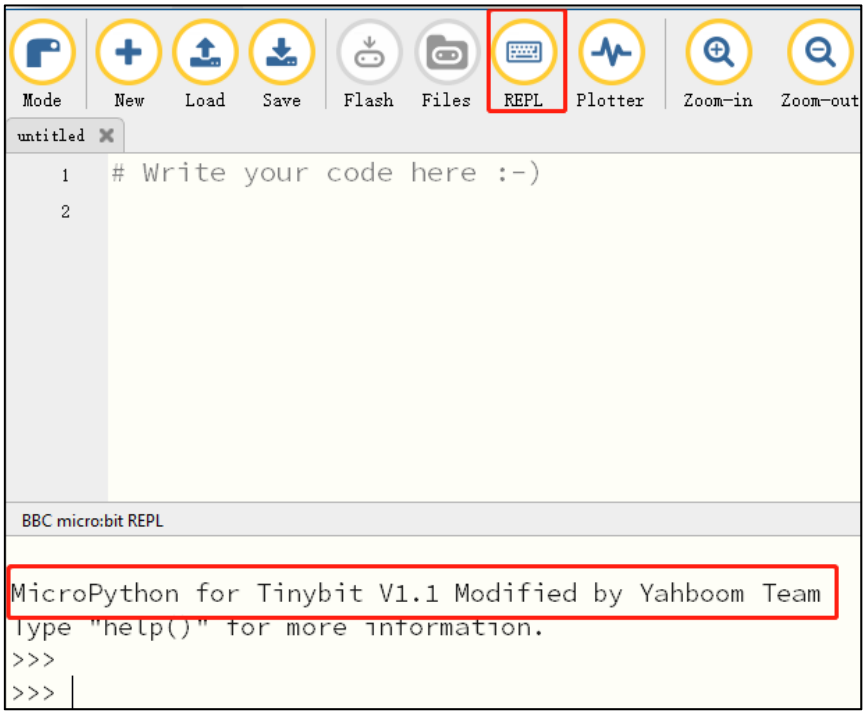
Note! All English and symbols should be entered in English, and the last line must be a space.



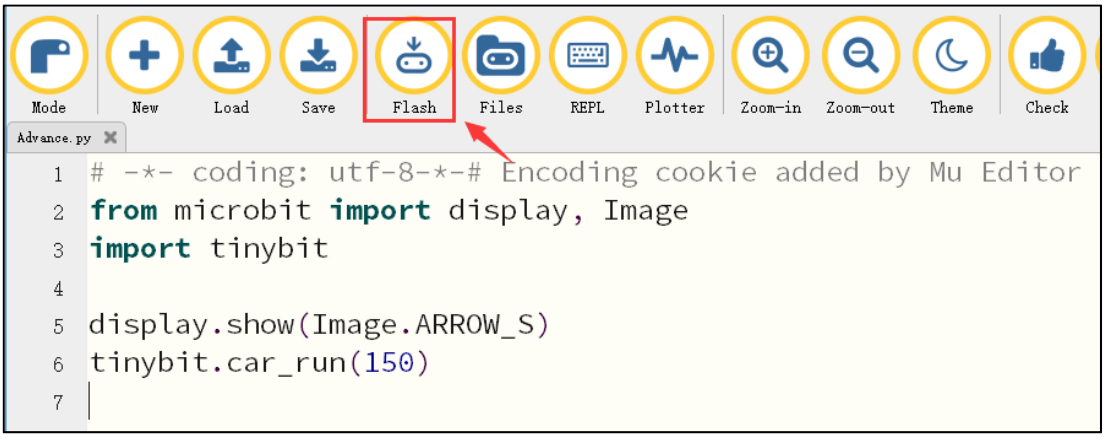
2.You can click the **"Check"** button to check if our code has an error. If a line appears with a cursor or an underscore, the program indicating this line is wrong.



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 Superbit library hex 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 handle will display heart pattern and clear.
We can see micro:bit dot matrix of car will display heart pattern all the time.

Then, they will pair automatically.
The button on the right side of the handle can control RGB lights. Press the rocker down to turn off the RGB lights. Push the rocker forward, backward, left and right to control the car movement.
Handle function as shown below.