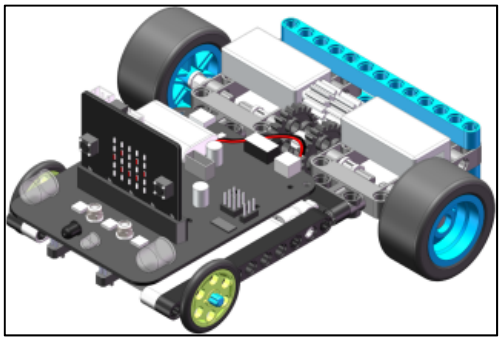
Python course Following car --- “micro:bit handle remote control”



1. Learning goals

After downloading the program, open the power switch of the Following car and the power switch of the micro:bit handle, they will be automatically paired. When you press the micro:bit handle remote control button, the Following car will have the corresponding action. The button on the right side of the remote control is used to control the color of the lights, and pressing the joystick down to turn off the light. Pushing the rocker forward, backward, left, right is to control the advance, back, turn left and turn right of the Following car .

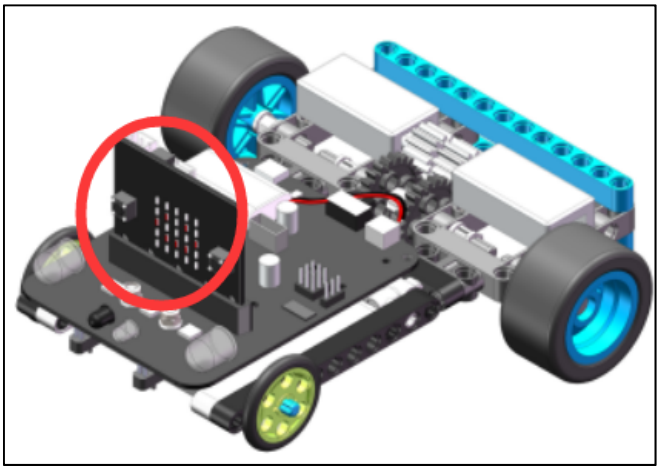
2. Preparation before class

We needs to be ready:
Building Block Following car *1
micro:bit handle*1
USB data cable*1

3. Programming

Following car code:

We started to write the program for the building block Following car. After the writing, we need to download the program to the micro:bit board of the building block Following car.



1) Import the libraries display, Image, buildingbit and radio to be used.

display.show (Image.HEART): Show a love icon on the microbit 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.

Note: the set group value It needs to be consistent with the handle setting, otherwise normal communication cannot be performed.

Code as shown below:

```
# -*- coding: utf-8-*-# Encoding cookie added by Mu Editor
from microbit import display, Image
import buildingbit
import radio

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

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 hook of the Tower crane rises; 'down' the Tower crane rises down; ‘left’ the car spin left; ‘right’ the car spin right; and 'stop' makes the motor stop.

Code as shown below:

```
while True:
    incoming = radio.receive()
    if incoming == 'up':
        buildingbit.car_run(100, 100, 0)
    elif incoming == 'down':
        buildingbit.car_back(100, 100, 0)
    elif incoming == 'left':
        buildingbit.car_spinleft(100, 100, 0)
    elif incoming == 'right':
        buildingbit.car_spinright(100, 100, 0)
    elif incoming == 'stop':
        buildingbit.car_stop()
```

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.

```
    if incoming == 'R':
        buildingbit.car_HeadRGB(255, 0, 0)
    elif incoming == 'G':
        buildingbit.car_HeadRGB(0, 255, 0)
    elif incoming == 'B':
        buildingbit.car_HeadRGB(0, 0, 255)
    elif incoming == 'Y':
        buildingbit.car_HeadRGB(255, 255, 0)
    elif incoming == 'turn_off':
        buildingbit.car_HeadRGB(0, 0, 0)
```

Handle control program:

We start to write the handle program. After writing, we need to download the program to the micro:bit board of the handle.

1) Import the libraries display, sleep, Image, ghandle, and radio that you need to use.

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 cookie added by Mu Editor
from microbit import display, sleep, Image
import ghandle
import radio

display.show(0)
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.

Code as shown below:

```
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()
```

3) 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.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():
        radio.send('R')
        display.show("R")
    if ghandle.B2_is_pressed():
        radio.send('G')
        display.show("G")
```

The above two programs, which we have provided. They are in the course directory.

If you want to use the program we provide directly, Please read the following carefully:

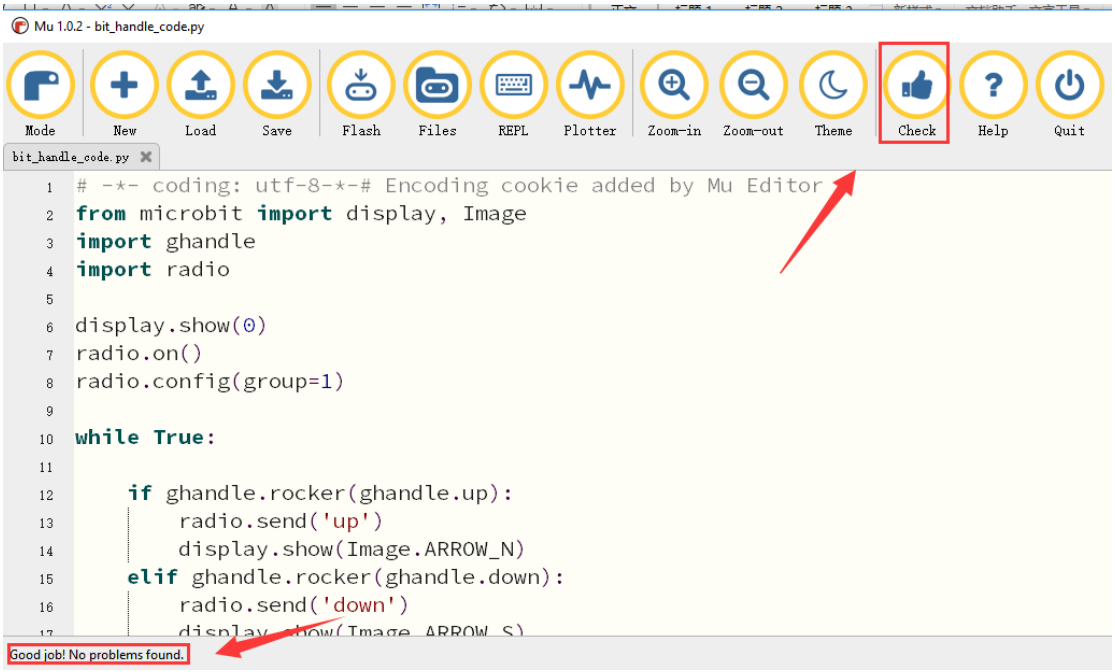
①You should download the Following_car_code.py to the micro:bit board of the Tower crane.

②You should download the bit_handle_code.py to the micro:bit board of the handle.

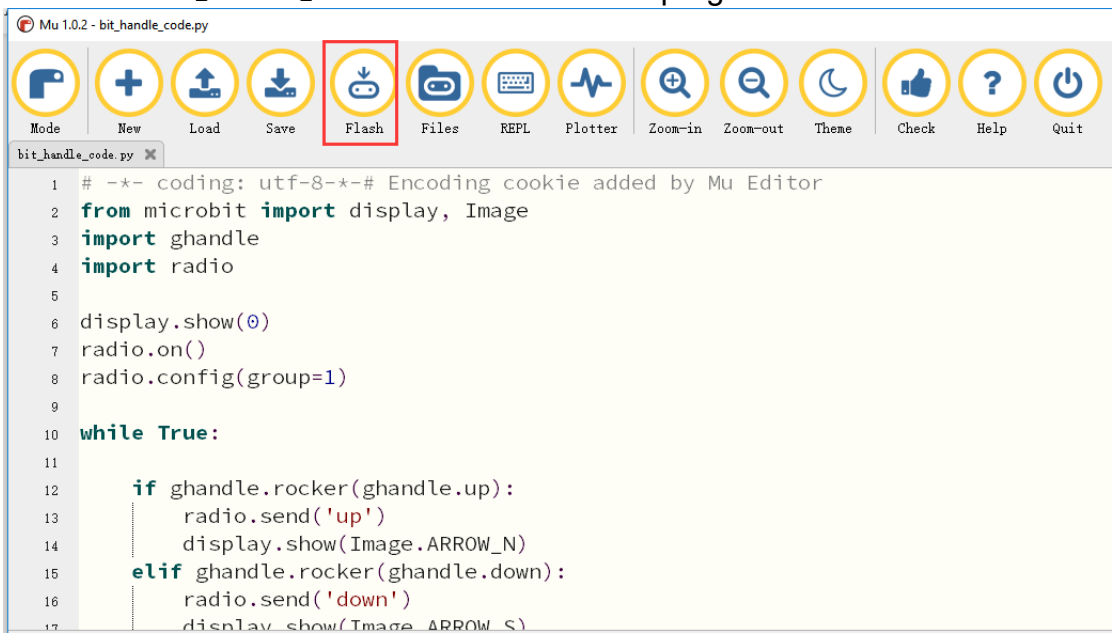
4. Download program

4.1 After programming is complete, please connect the computer and the micro:bit board with a Micro USB data cable.

4.2 You need to 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. If there is no cursor or underline, it means that the code is correct, and the bottom left will prompt that the check is OK.



4.3 Click the【**Flash**】button to download the program to the micro: bit board.



If the program is wrong or the experimental phenomenon is wrong after downloading, please confirm whether you have downloaded the Buildingbit 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】

After the program is downloaded, the handle and the car are powered up normally, they will be automatically paired, you can start to control Tower crane by handle within valid range.