**Course18-Robot advance**

****Learning goals:****

This lesson learns to use Python programming to make the robot advance.

**Code：**

from microbit import \*

import ustruct

import math

# Registers/etc:

PCA9685\_ADDRESS = 0x41

MODE1 = 0x00

MODE2 = 0x01

SUBADR1 = 0x02

SUBADR2 = 0x03

SUBADR3 = 0x04

PRESCALE = 0xFE

LED0\_ON\_L = 0x06

LED0\_ON\_H = 0x07

LED0\_OFF\_L = 0x08

LED0\_OFF\_H = 0x09

ALL\_LED\_ON\_L = 0xFA

ALL\_LED\_ON\_H = 0xFB

ALL\_LED\_OFF\_L = 0xFC

ALL\_LED\_OFF\_H = 0xFD

# Bits:

RESTART = 0x80

SLEEP = 0x10

ALLCALL = 0x01

INVRT = 0x10

OUTDRV = 0x04

RESET = 0x00

class PCA9685():

"""PCA9685 PWM LED/servo controller."""

def \_\_init\_\_(self, address=PCA9685\_ADDRESS):

"""Initialize the PCA9685."""

self.address = address

i2c.write(self.address, bytearray([MODE1, RESET]))

self.set\_all\_pwm(0, 0)

i2c.write(self.address, bytearray([MODE2, OUTDRV]))

i2c.write(self.address, bytearray([MODE1, ALLCALL]))

sleep(5) # wait for oscillator

i2c.write(self.address, bytearray([MODE1]))

mode1 = i2c.read(self.address, 1)

mode1 = ustruct.unpack('<H', mode1)[0]

mode1 = mode1 & ~SLEEP # wake up (reset sleep)

i2c.write(self.address, bytearray([MODE1, mode1]))

sleep(5) # wait for oscillator

def set\_pwm\_freq(self, freq\_hz):

"""Set the PWM frequency to the provided value in hertz."""

prescaleval = 25000000.0 # 25MHz

prescaleval /= 4096.0 # 12-bit

prescaleval /= float(freq\_hz)

prescaleval -= 1.0

# print('Setting PWM frequency to {0} Hz'.format(freq\_hz))

# print('Estimated pre-scale: {0}'.format(prescaleval))

prescale = int(math.floor(prescaleval + 0.5))

# print('Final pre-scale: {0}'.format(prescale))

i2c.write(self.address, bytearray([MODE1]))

oldmode = i2c.read(self.address, 1)

oldmode = ustruct.unpack('<H', oldmode)[0]

newmode = (oldmode & 0x7F) | 0x10 # sleep

i2c.write(self.address, bytearray([MODE1, newmode])) # go to sleep

i2c.write(self.address, bytearray([PRESCALE, prescale]))

i2c.write(self.address, bytearray([MODE1, oldmode]))

sleep(5)

i2c.write(self.address, bytearray([MODE1, oldmode | 0x80]))

def set\_pwm(self, channel, on, off):

"""Sets a single PWM channel."""

if on is None or off is None:

i2c.write(self.address, bytearray([LED0\_ON\_L+4\*channel]))

data = i2c.read(self.address, 4)

return ustruct.unpack('<HH', data)

i2c.write(self.address, bytearray([LED0\_ON\_L+4\*channel, on & 0xFF]))

i2c.write(self.address, bytearray([LED0\_ON\_H+4\*channel, on >> 8]))

i2c.write(self.address, bytearray([LED0\_OFF\_L+4\*channel, off & 0xFF]))

i2c.write(self.address, bytearray([LED0\_OFF\_H+4\*channel, off >> 8]))

def set\_all\_pwm(self, on, off):

"""Sets all PWM channels."""

i2c.write(self.address, bytearray([ALL\_LED\_ON\_L, on & 0xFF]))

i2c.write(self.address, bytearray([ALL\_LED\_ON\_H, on >> 8]))

i2c.write(self.address, bytearray([ALL\_LED\_OFF\_L, off & 0xFF]))

i2c.write(self.address, bytearray([ALL\_LED\_OFF\_H, off >> 8]))

def duty(self, index, value=None, invert=False):

if value is None:

pwm = self.set\_pwm(index)

if pwm == (0, 4096):

value = 0

elif pwm == (4096, 0):

value = 4095

value = pwm[1]

if invert:

value = 4095 - value

return value

if not 0 <= value <= 4095:

raise ValueError("Out of range")

if invert:

value = 4095 - value

if value == 0:

self.set\_pwm(index, 0, 4096)

elif value == 4095:

self.set\_pwm(index, 4096, 0)

else:

self.set\_pwm(index, 0, value)

# Initialise the PCA9685 using the default address (0x41).

pwm = PCA9685()

# Configure min and max servo pulse lengths

servo\_min = 150 # Min pulse length out of 4096 0?

servo\_max = 600 # Max pulse length out of 4096: 180?

# Set frequency to 60hz, good for servos.

pwm.set\_pwm\_freq(60)

display.show(Image.HAPPY)

pwm.set\_pwm(12, 0, 4095)

pwm.set\_pwm(13, 0, 0)

pwm.set\_pwm(15, 0, 4095)

pwm.set\_pwm(14, 0, 0)

pwm.set\_pwm(3, 0, servo\_min)

sleep(1000)

pwm.set\_pwm(3, 0, servo\_max)

sleep(1000)

This section of the experiment uses I2C communication, through the PCA9685PW chip can output 16 PWM, so we can control the output of 4-way PWM control car forward.

**Programming and downloading：**

1. You should open the Mu software, and enter the code in the edit window, , as shown in Figure 14-1.

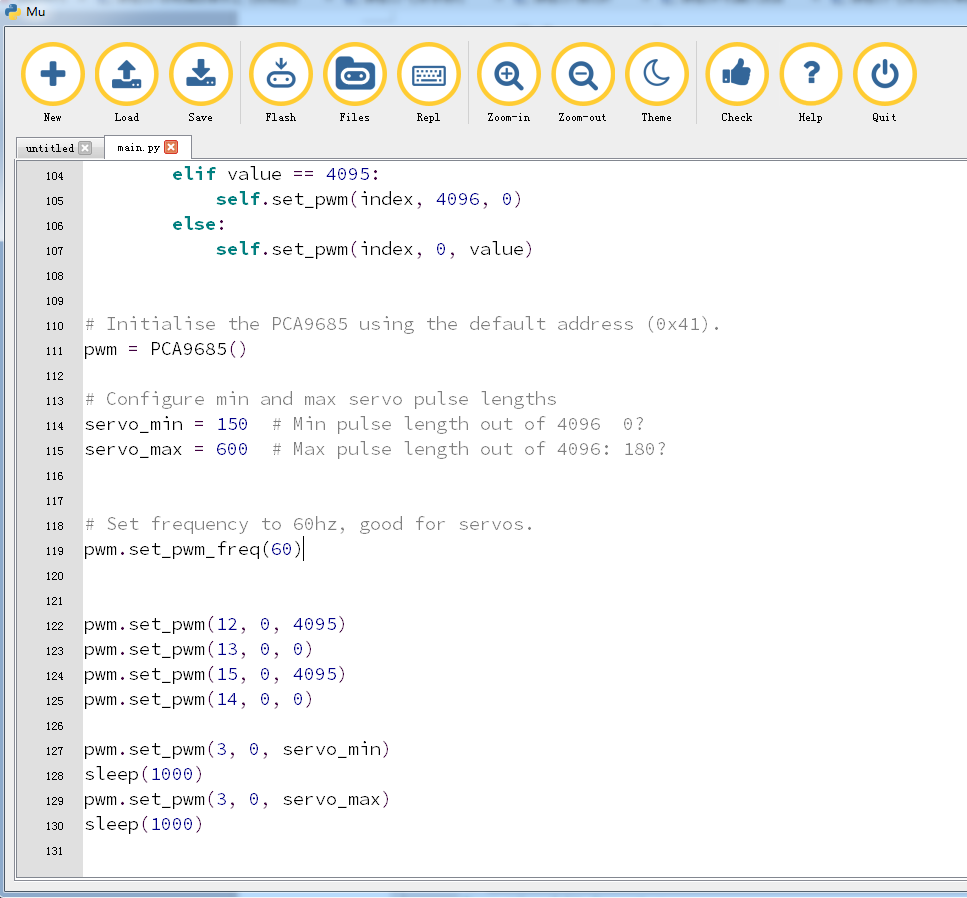


Figure 14-1

2.As shown in Figure 14-2, 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.

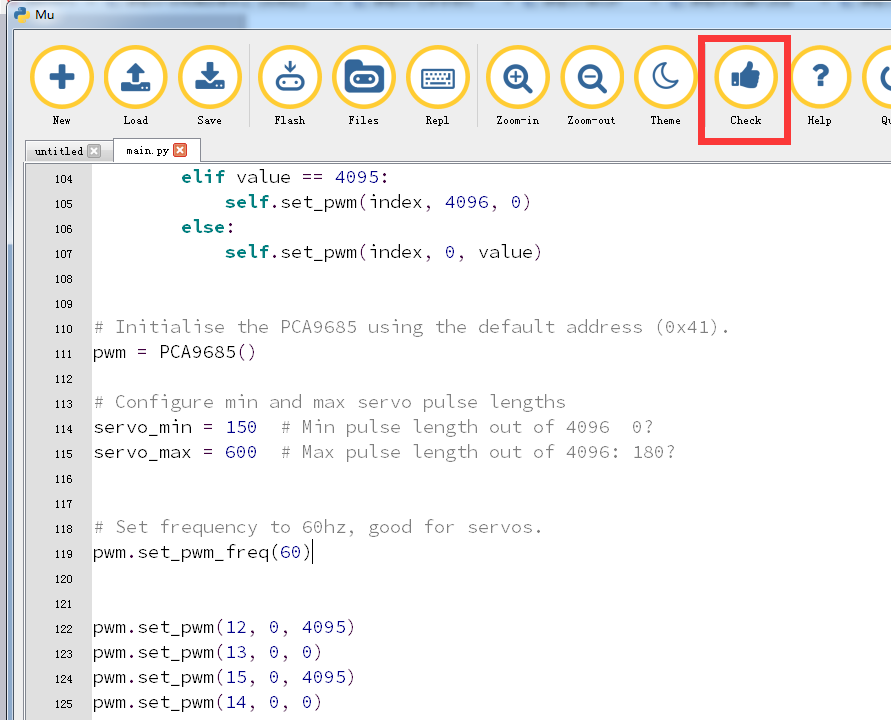


Figure 14-2

3.You need to connect the micro data cable to micro:bit and the computer, then click the Flash button to download the program to micro:bit as shown in Figure 14-3.

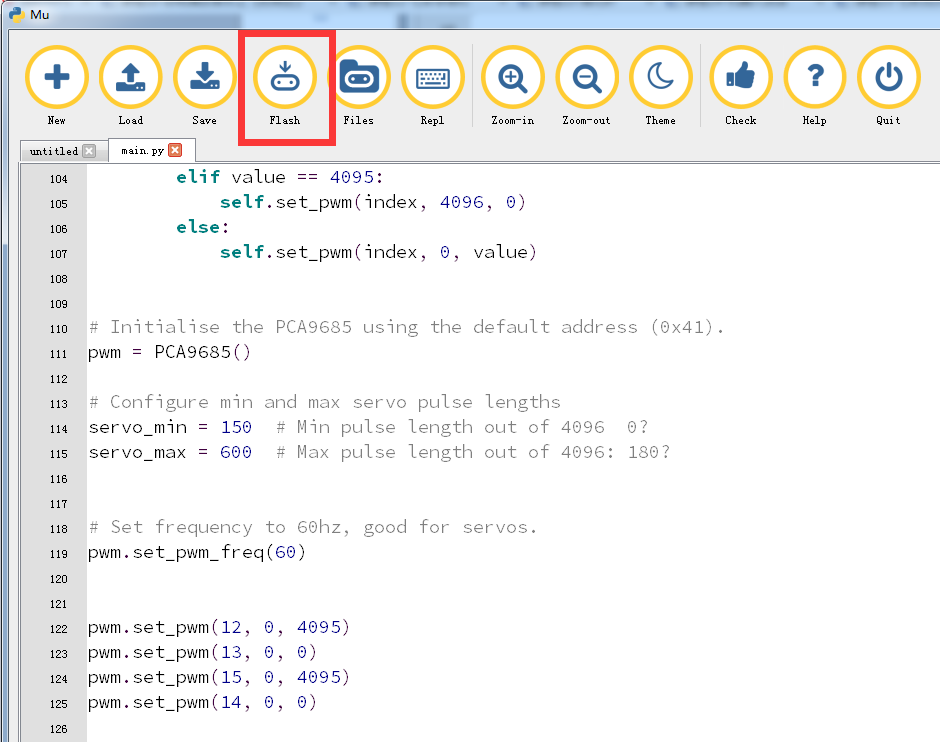


Figure 14-3

4. The schematic diagram of the robot's PCA9685PW chip and motor is shown in Figure 14-4 and Figure 14-5. As shown in the figure, the robot's motor is connected to the LINA, LINB, RINA, and RINB pins of the PCA9685PW chip, while the PCA9685PW The SCL and SDA are connected to the P19 and P20 pins of the micro:bit chip.

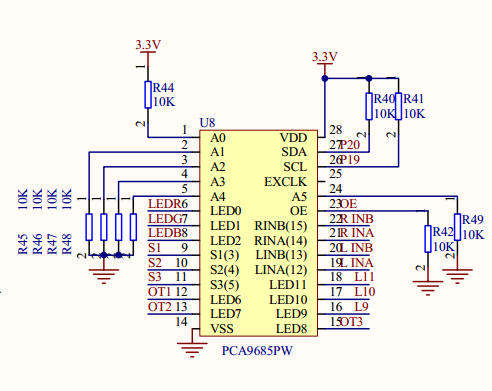


Figure 14-4

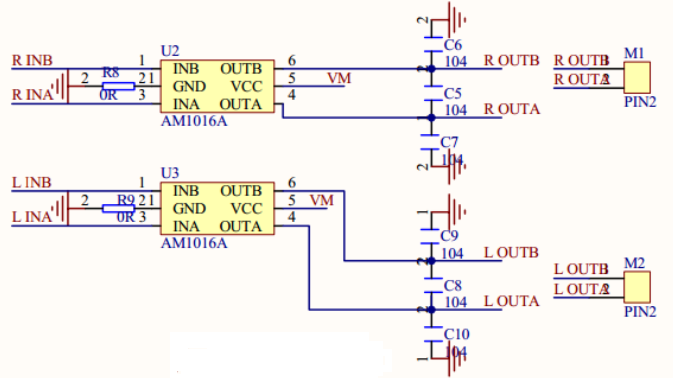


Figure 14-5

5. After downloading the program to micro:bit, you can see the robot will advance. As shown in Figures 14-6.



Figures 14-6