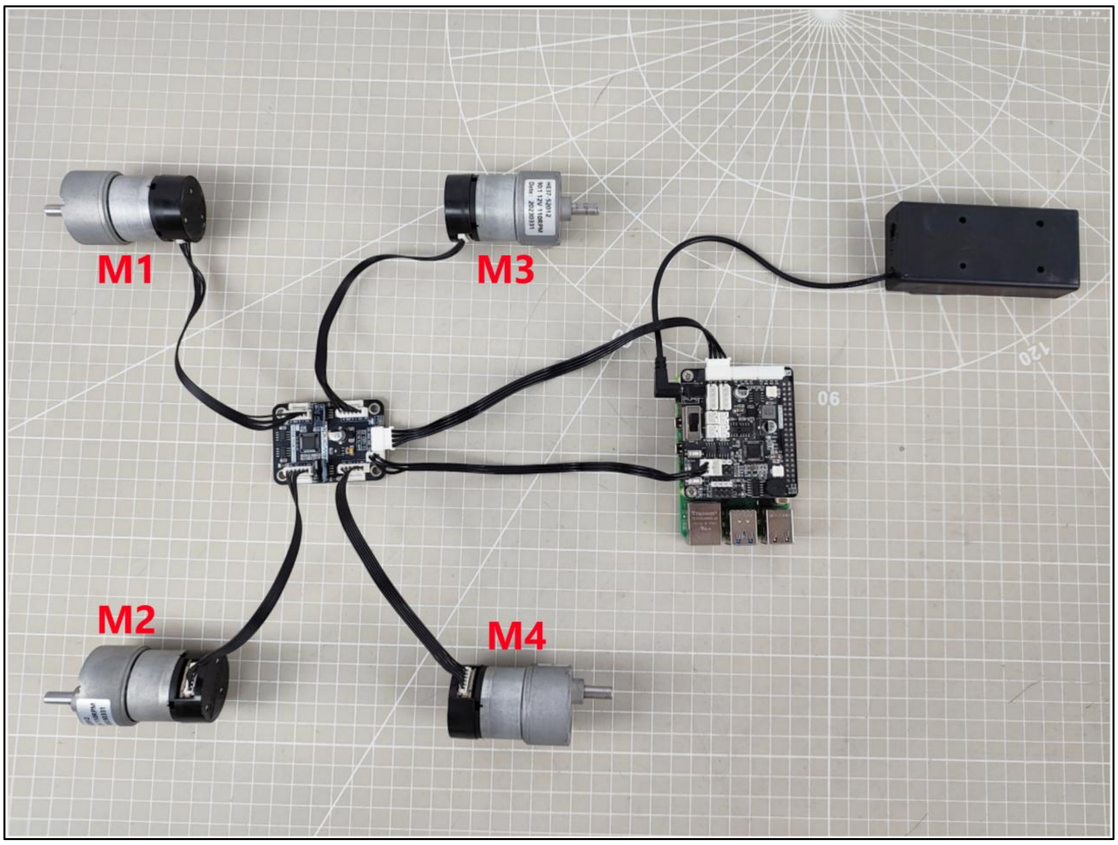
Raspberry Pi Development

This lesson employs the DC geared motor for control demonstration. It is applicable to Raspberry Pi 4B and 5 controllers.

1. Hardware Connection

Before calling the program, connect the 4-ch encoder motor driver to the Raspberry Pi. The specific connection method is shown below:



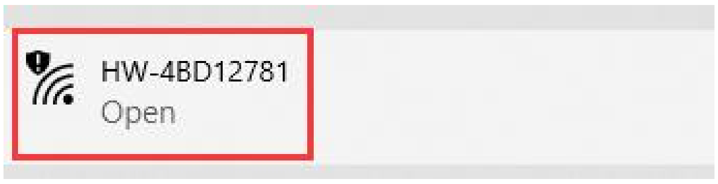
2. Program Download

The entered command should be case sensitive. The "Tab" key on the keyboard can be used to complete the keywords.

This program takes the Raspberry Pi 4B controller as an example. Please prepare a Raspberry Pi 4B development board and expansion board.

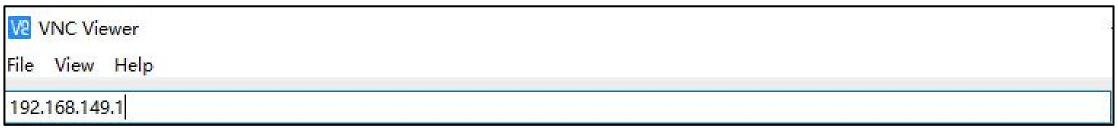
Operation Steps:

1) Turn on the Raspberry Pi. Connect the computer to corresponding WiFi.

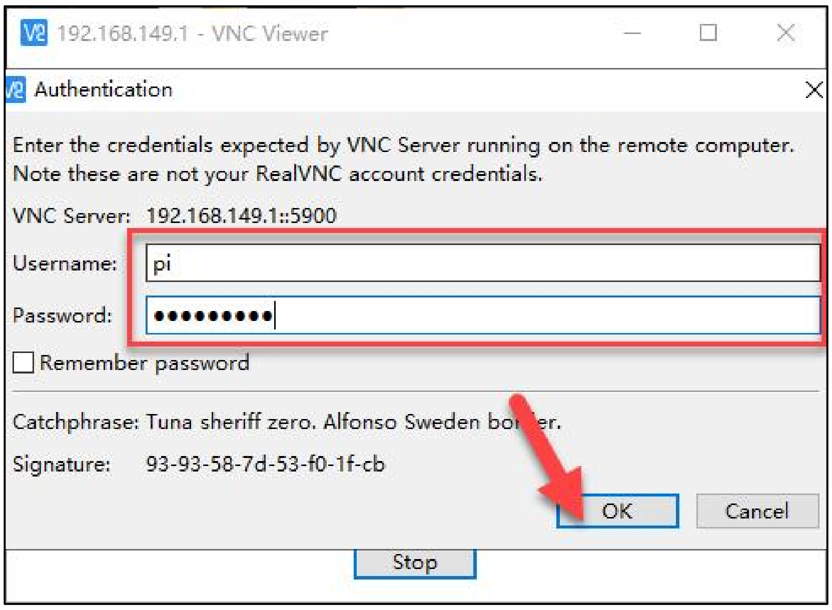


2) Open VNC Viewer.

3) Enter the default IP address of the Raspberry Pi "192.168.149.1" in the VNC Viewer. Then, press "Enter".



4) Enter the password "raspberrypi" in the pop-up prompt box. If it requires you to enter an account name, use "pi". Check the "Remember password" box. Then click "OK". The Raspberry Pi's desktop will display on your computer's monitor.



Enter the credentials expected by VNC Server running on the remote computer.
Note these are not your RealVNC account credentials.

VNC Server: 192.168.149.1::5900

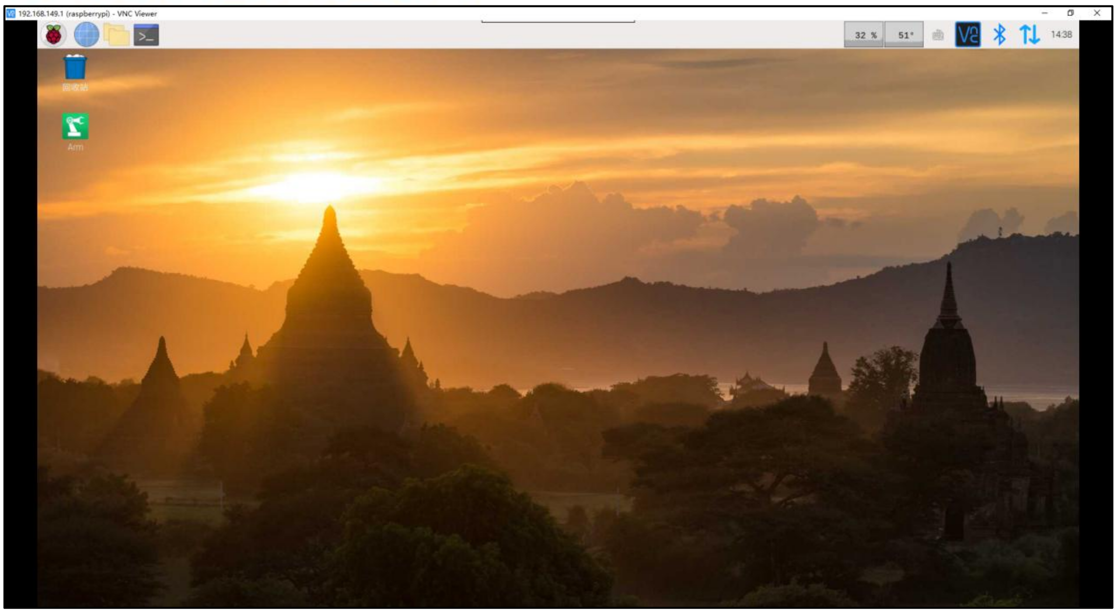
Username: pi

Password: ●●●●●●●●●●

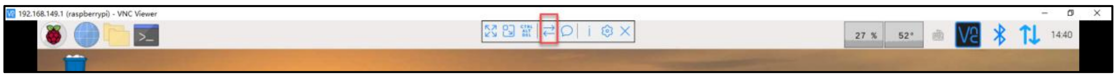
☐ Remember password

Catchphrase: Tuna sheriff zero. Alfonso Sweden bo[illegible]er.

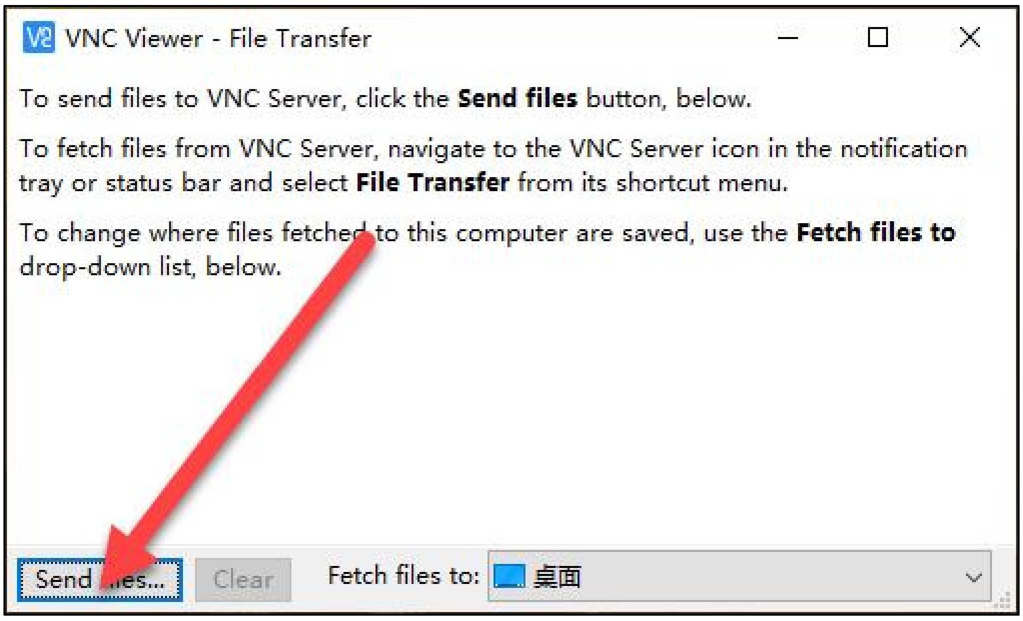
Signature: 93-93-58-7d-53-f0-1f-cb



5) Click to transfer files.



6) Click "Send files" to select the file.



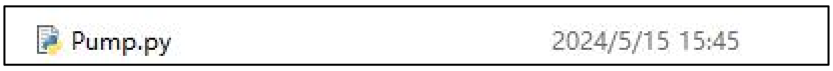
VNC Viewer - File Transfer

To send files to VNC Server, click the **Send files** button, below.

To fetch files from VNC Server, navigate to the VNC Server icon in the notification tray or status bar and select **File Transfer** from its shortcut menu.

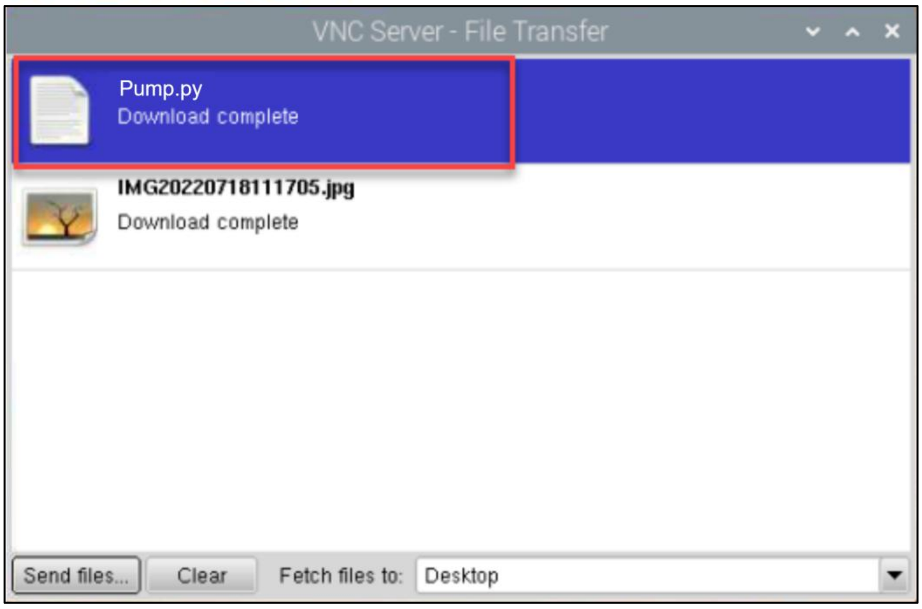
To change where files fetched to this computer are saved, use the **Fetch files to** drop-down list, below.

7) Select the file "Pump.py".



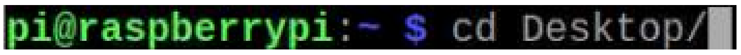
Pump.py 2024/5/15 15:45

8) When the following image appears, it means that the transfer is complete:



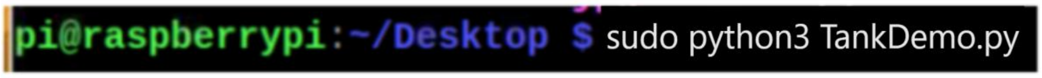
3. Program Outcome

1) Enter the command "cd Desktop" to enter the desktop.



```
pi@raspberrypi:~ $ cd Desktop/
```

2) Enter the command "sudo python3 TankDemo.py" to run the program.



```
pi@raspberrypi:~/Desktop $ sudo python3 TankDemo.py
```

3) After running the program, the four motors will continuously rotate clockwise for 3 seconds, and counterclockwise for 3 seconds.

4. Program Analysis

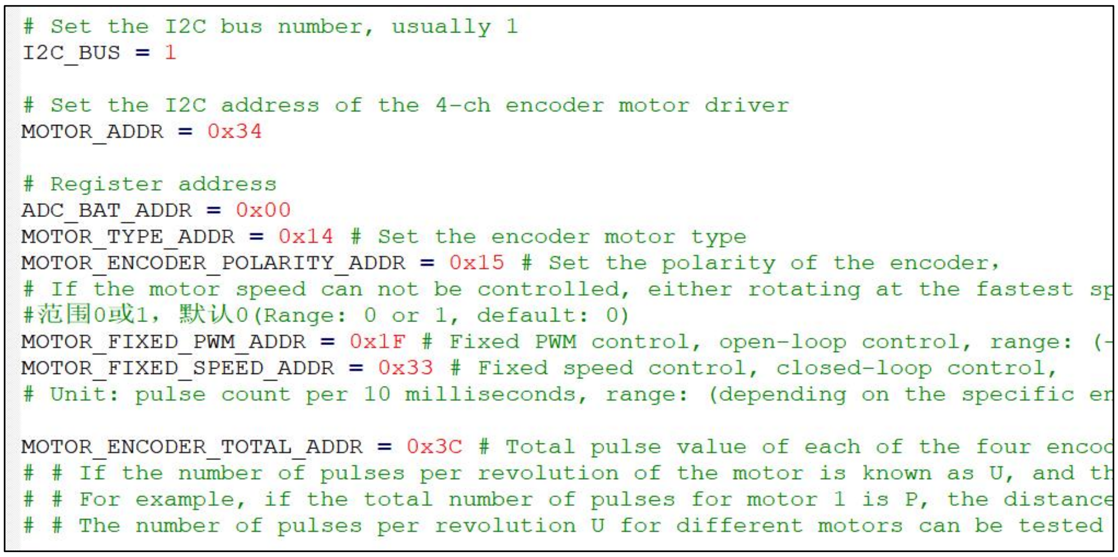
The overall process of the program is to prepare the necessary libraries and the relevant parameters for motor initialization. Then, control the motor through logical methods.

● **Import necessary libraries**

```python
import smbus
import time
import struct
```

The smbus module is used for I2C communication. The 4-ch encoder motor driver is controlled and read through I2C. The time module is used for system delay. The struct module is used for data parsing.

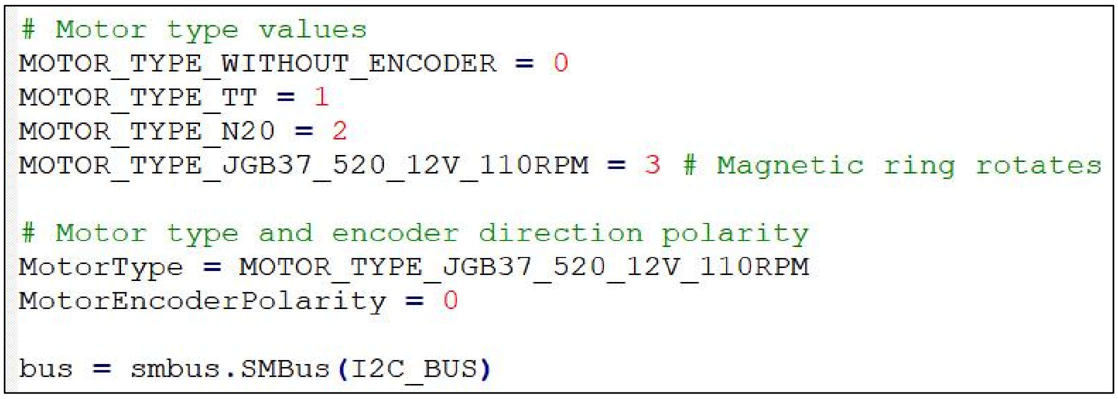
- **Initialize communication address**



```
# Set the I2C bus number, usually 1
I2C_BUS = 1

# Set the I2C address of the 4-ch encoder motor driver
MOTOR_ADDR = 0x34

# Register address
ADC_BAT_ADDR = 0x00
MOTOR_TYPE_ADDR = 0x14 # Set the encoder motor type
MOTOR_ENCODER_POLARITY_ADDR = 0x15 # Set the polarity of the encoder,
# If the motor speed can not be controlled, either rotating at the fastest sp
#范围0或1，默认0(Range: 0 or 1, default: 0)
MOTOR_FIXED_PWM_ADDR = 0x1F # Fixed PWM control, open-loop control, range: (-
MOTOR_FIXED_SPEED_ADDR = 0x33 # Fixed speed control, closed-loop control,
# Unit: pulse count per 10 milliseconds, range: (depending on the specific en

MOTOR_ENCODER_TOTAL_ADDR = 0x3C # Total pulse value of each of the four encod
# # If the number of pulses per revolution of the motor is known as U, and th
# # For example, if the total number of pulses for motor 1 is P, the distance
# # The number of pulses per revolution U for different motors can be tested
```

Define the I2C communication address and the address of each type of encoder motor for easy calling later. The I2C communication address connected to the driver board is 0x34, which depends on your own hardware device. The type of the encoder motor is set to 0x14. Its direction polarity is set to 0x15. The above two numbers represent the address position of the written parameters, not the actual parameter values. Their values depend on your hardware device. Keep them default.

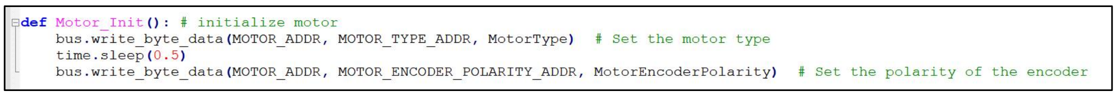
- **Initialize motor type**



```
# Motor type values
MOTOR_TYPE_WITHOUT_ENCODER = 0
MOTOR_TYPE_TT = 1
MOTOR_TYPE_N20 = 2
MOTOR_TYPE_JGB37_520_12V_110RPM = 3 # Magnetic ring rotates

# Motor type and encoder direction polarity
MotorType = MOTOR_TYPE_JGB37_520_12V_110RPM
MotorEncoderPolarity = 0

bus = smbus.SMBus(I2C_BUS)
```

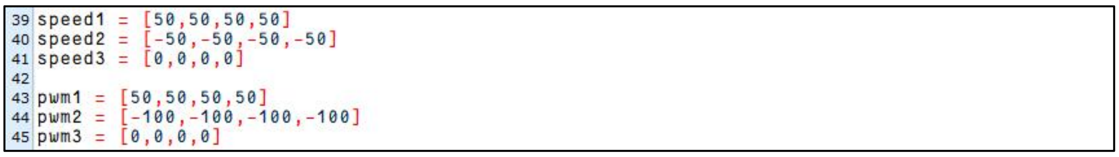
The 4-ch encoder motor driver supports driving multiple types of motors, including TTL, N20, and JGB motor types. They are all defined as specific types. In this development, JGB motors are used. Therefore, the motor model definition is set to 3. The I2C communication object is initialized.

- **Module initialization**



```
def Motor_Init(): # initialize motor
    bus.write_byte_data(MOTOR_ADDR, MOTOR_TYPE_ADDR, MotorType)  # Set the motor type
    time.sleep(0.5)
    bus.write_byte_data(MOTOR_ADDR, MOTOR_ENCODER_POLARITY_ADDR, MotorEncoderPolarity)  # Set the polarity of the encoder
```

Use the “bus.write_byte_data()” function to write messages to the motor. The motor type and encoding polarity are set.

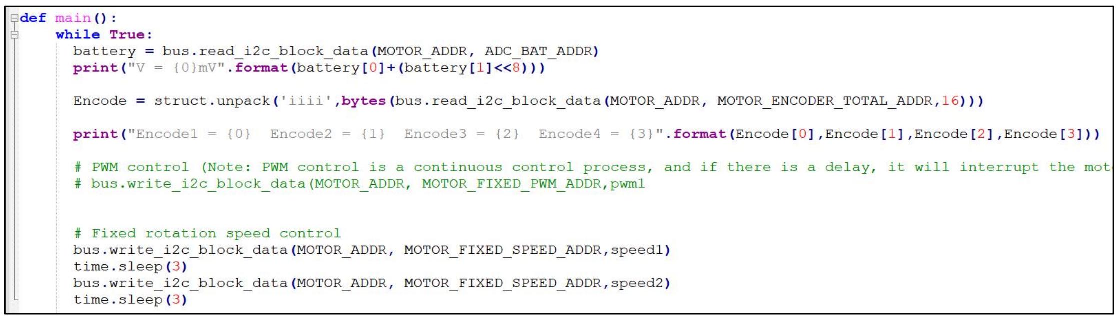


```
speed1 = [50,50,50,50]
speed2 = [-50,-50,-50,-50]
speed3 = [0,0,0,0]

pwm1 = [50,50,50,50]
pwm2 = [-100,-100,-100,-100]
pwm3 = [0,0,0,0]
```

In this program, mainly call the I2C communication sending data set. The motor driver module can control the rotation of 4 motors simultaneously. Encapsulate the rotation speed of the 4 motors in an array. Send it to the motor driver module. After the data is received, the driver module will control the motor rotation based on the set speed.

Note: If it is PWM control, continuous sending is required. If there is a delay in the program, it will affect the motor rotation speed!

- **Main function**



```
def main():
    while True:
        battery = bus.read_i2c_block_data(MOTOR_ADDR, ADC_BAT_ADDR)
        print("V = {0}mV".format(battery[0]+(battery[1]<<8)))

        Encode = struct.unpack('iiii',bytes(bus.read_i2c_block_data(MOTOR_ADDR, MOTOR_ENCODER_TOTAL_ADDR,16)))

        print("Encode1 = {0}  Encode2 = {1}  Encode3 = {2}  Encode4 = {3}".format(Encode[0],Encode[1],Encode[2],Encode[3]))

        # PWM control (Note: PWM control is a continuous control process, and if there is a delay, it will interrupt the mot
        # bus.write_i2c_block_data(MOTOR_ADDR, MOTOR_FIXED_PWM_ADDR,pwm1


        # Fixed rotation speed control
        bus.write_i2c_block_data(MOTOR_ADDR, MOTOR_FIXED_SPEED_ADDR,speed1)
        time.sleep(3)
        bus.write_i2c_block_data(MOTOR_ADDR, MOTOR_FIXED_SPEED_ADDR,speed2)
        time.sleep(3)
```

In the main function, use the “bus.read_i2c_block_data()” function to read the battery level value of the 4-ch encoder motor driver. The “MOTOR_ ENCODER_TOTAL_ADDR” register is read to obtain the total pulse value of each of the 4 encoder motors.

Use the “bus.write_i2c_block_data()” to write the speed of motor control. The “MOTOR_FIXED_PWM_ADDR” represents the PWM speed control address.

The “MOTOR_FIXED_SPEED_ADDR” represents the fixed speed control address. The pwm1, speed1, and speed2 arrays are used to control the motor speed. Adjust the corresponding motor speed value in the array to control the motor with different motion effects .