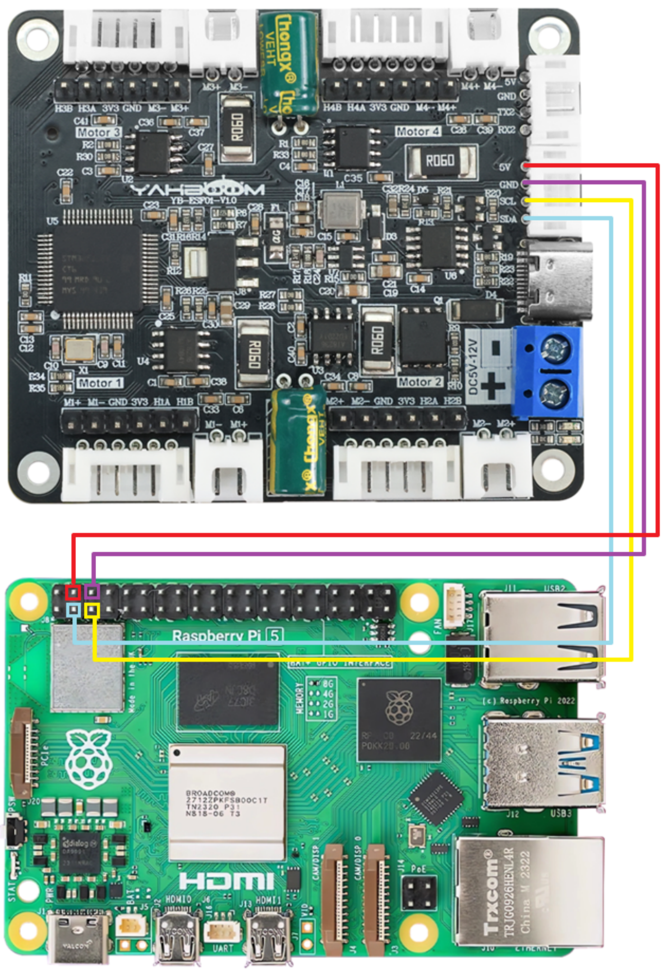
Drive motor and read encoder-IIC

1.1 Explanation

Please read 《0. Motor introduction and usage》first to understand the motor parameters, wiring method, and power supply voltage you are currently using. To avoid improper operation and damage to the driver board or motor.

I2C and serial communication cannot be shared, only one can be selected.

Hardware wiring:



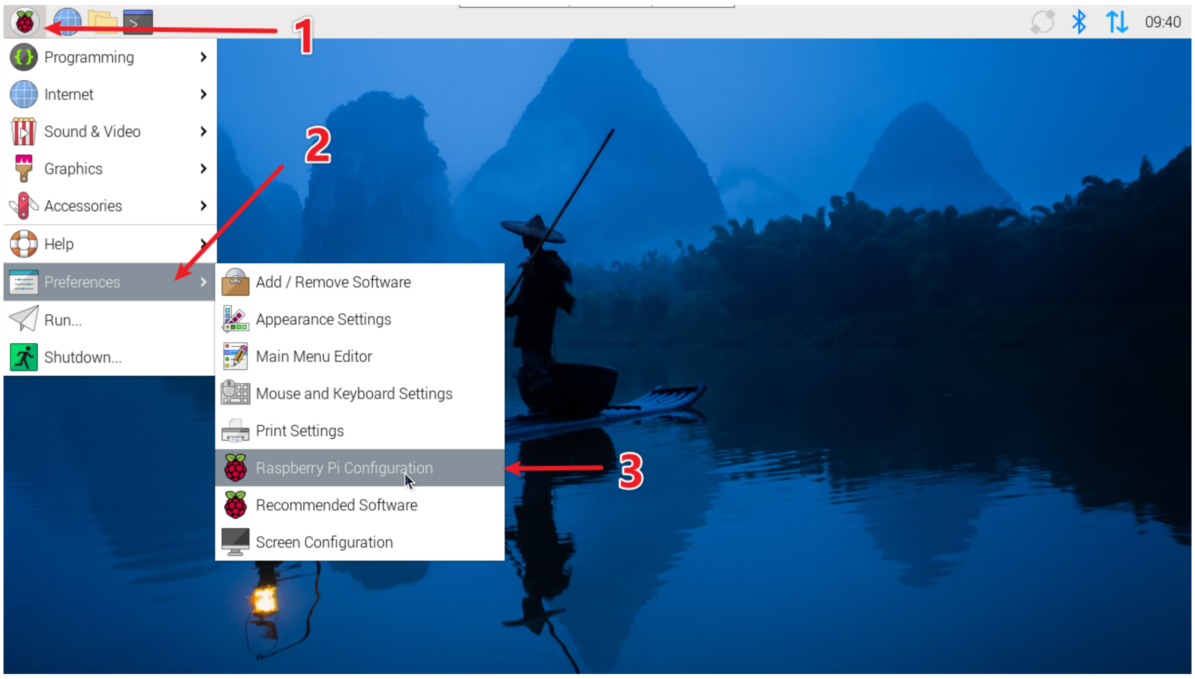
Motor	4-channel motor drive board(Motor)
M2	M-
V	3V3
A	H1A
B	H1B
G	GND
M1	M+

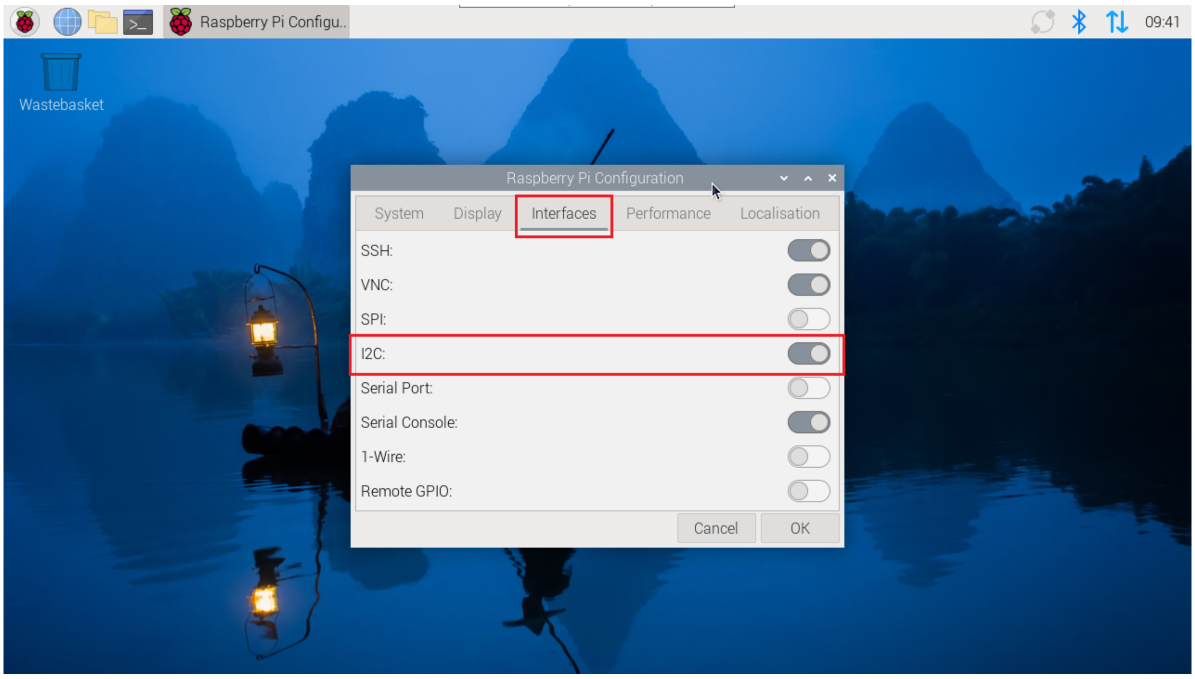
4-channel motor drive board	Raspberry Pi 5(Physical Pins)
SDA	3
SCL	5
GND	6
5V	4

1.2 Instructions

1.2.1 Enable I2C support on the Raspberry Pi board

The setting method of the graphical interface is as follows:





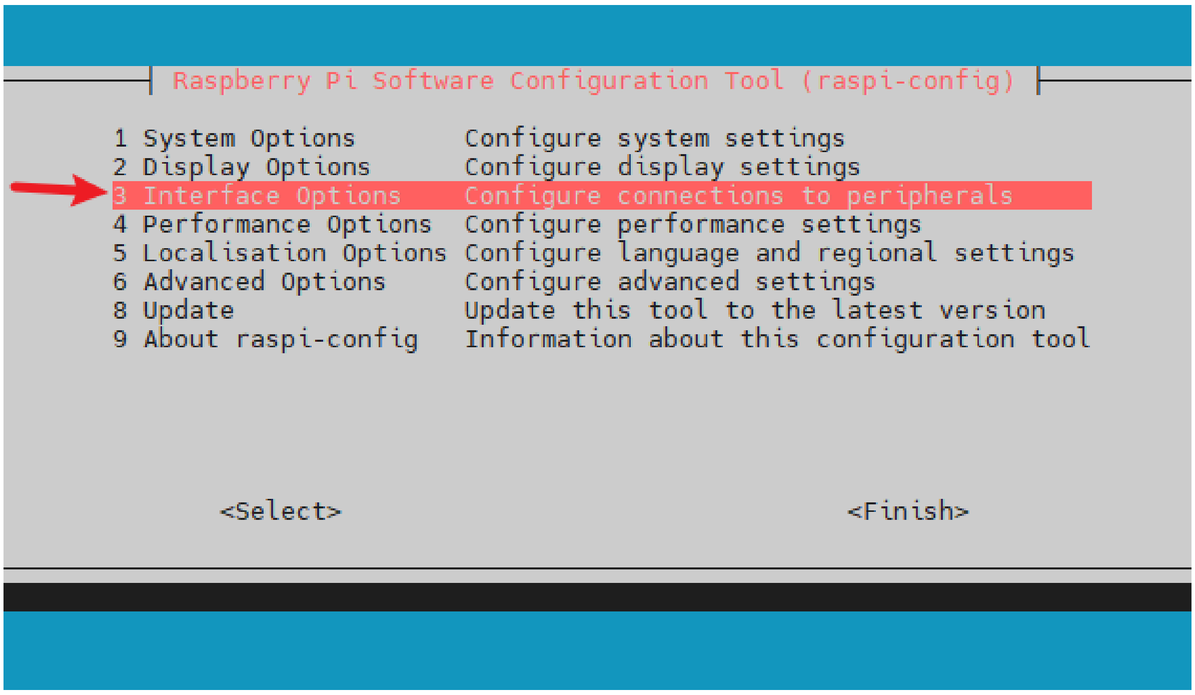
SSH: on
VNC: on
SPI: off
I2C: on
Serial Port: off
Serial Console: on
1-Wire: off
Remote GPIO: off

Turn on the I2C switch in the window again and click OK.

The command line settings are as follows:

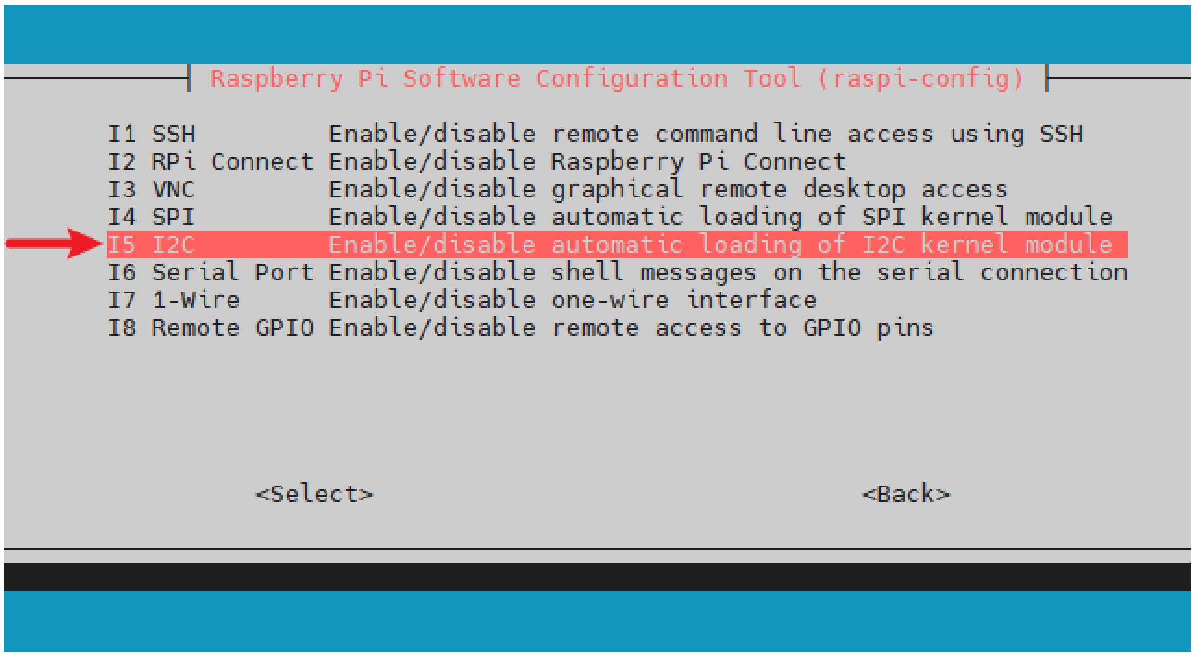
After logging in using SSH, use the following command to enter the raspi-config command interface.

```
sudo raspi-config
```



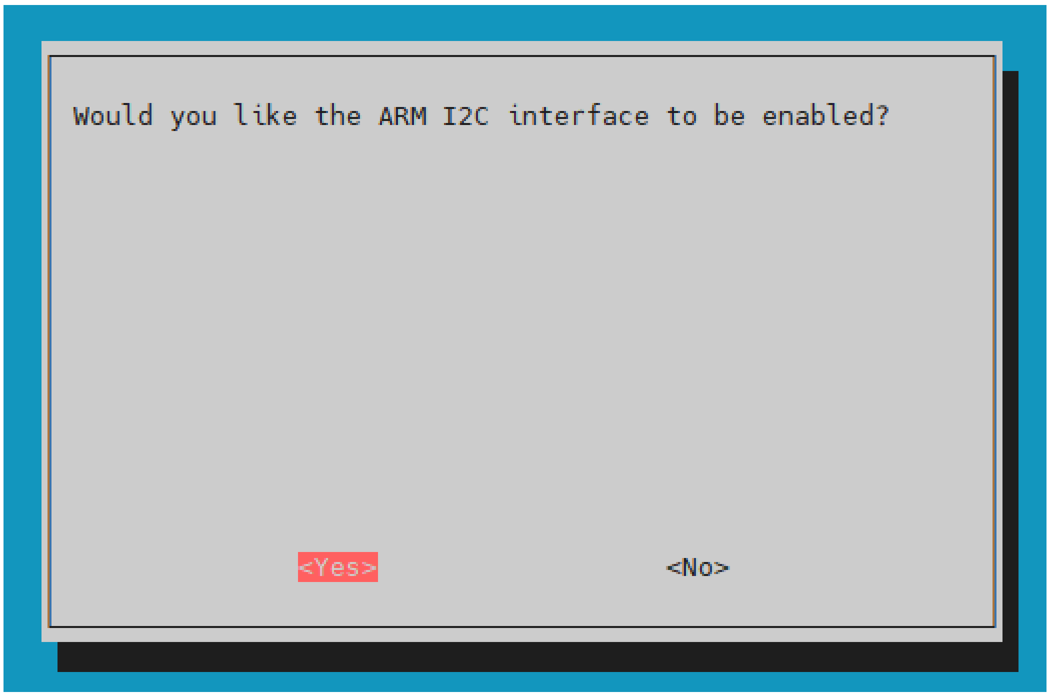
```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

 1 System Options       Configure system settings
 2 Display Options      Configure display settings
 3 Interface Options    Configure connections to peripherals
 4 Performance Options  Configure performance settings
 5 Localisation Options Configure language and regional settings
 6 Advanced Options     Configure advanced settings
 8 Update               Update this tool to the latest version
 9 About raspi-config   Information about this configuration tool

        <Select>                        <Finish>
```

Choose**Interfacing Options**



```
┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

I1 SSH         Enable/disable remote command line access using SSH
I2 RPi Connect Enable/disable Raspberry Pi Connect
I3 VNC         Enable/disable graphical remote desktop access
I4 SPI         Enable/disable automatic loading of SPI kernel module
I5 I2C         Enable/disable automatic loading of I2C kernel module
I6 Serial Port Enable/disable shell messages on the serial connection
I7 1-Wire      Enable/disable one-wire interface
I8 Remote GPIO Enable/disable remote access to GPIO pins

        <Select>                        <Back>
```

Select I2C and press **Enter** key



```
Would you like the ARM I2C interface to be enabled?
```

Select Yes and press Enter to confirm. This will turn on I2C.

1.2.2 Run file

Use file transfer software, such as WinSCP. You need to search and download the software yourself.

Transfer the py file to the root directory of the Raspberry Pi board through the software, then open the terminal and run the command:

```
sudo python ~/IIC.py
```

1.3 Code analysis

```
UPLOAD_DATA = 1  #1:接收总的编码器数据 2:接收实时的编码器
                 #1: Receive total encoder data 2: Receive real-time encoder

MOTOR_TYPE = 1  #1:520电机 2:310电机 3:测速码盘TT电机 4:TT直流减速电机 5:L型520电机
                #1:520 motor 2:310 motor 3:speed code disc TT motor 4:TT DC
reduction motor 5:L type 520 motor
```

- UPLOAD_DATA: used to set the data of the motor encoder. Set 1 to the total number of encoder pulses and 2 to the real-time pulse data of 10ms.
- MOTOR_TYPE: used to set the type of motor used. Just modify the corresponding numbers according to the comments according to the motor you are currently using. You don't need to modify the rest of the code.

If you need to drive the motor and observe the data, just modify the two numbers at the beginning of the program. No changes are required to the rest of the code.

```
def set_motor_parameter():
    if MOTOR_TYPE == 1:
        set_motor_type(1)  # 配置电机类型
        time.sleep(0.1)
        set_pluse_phase(30)  # 配置减速比，查电机手册得出
        time.sleep(0.1)
        set_pluse_line(11)  # 配置磁环线，查电机手册得出
        time.sleep(0.1)
        set_wheel_dis(67.00)  # 配置轮子直径，测量得出
        time.sleep(0.1)
        set_motor_deadzone(1900)  # 配置电机死区，实验得出
        time.sleep(0.1)

    elif MOTOR_TYPE == 2:
        set_motor_type(2)
        time.sleep(0.1)
        set_pluse_phase(20)
        time.sleep(0.1)
        set_pluse_line(13)
        time.sleep(0.1)
        set_wheel_dis(48.00)
        time.sleep(0.1)
        set_motor_deadzone(1600)
        time.sleep(0.1)

    elif MOTOR_TYPE == 3:
        set_motor_type(3)
        time.sleep(0.1)
        set_pluse_phase(45)
        time.sleep(0.1)
        set_pluse_line(13)
        time.sleep(0.1)
        set_wheel_dis(68.00)
        time.sleep(0.1)
        set_motor_deadzone(1250)
        time.sleep(0.1)

    elif MOTOR_TYPE == 4:
        set_motor_type(4)
        time.sleep(0.1)
        set_pluse_phase(48)
        time.sleep(0.1)
        set_motor_deadzone(1000)
        time.sleep(0.1)

    elif MOTOR_TYPE == 5:
        set_motor_type(1)
        time.sleep(0.1)
        set_pluse_phase(40)
        time.sleep(0.1)
        set_pluse_line(11)
        time.sleep(0.1)
        set_wheel_dis(67.00)
        time.sleep(0.1)
        set_motor_deadzone(1900)
        time.sleep(0.1)
```

This is used to store the parameters of the Yahboom motor. By modifying the MOTOR_TYPE parameter above, one-click configuration can be achieved.

In normally, do not modify the code here when using the Yahboom motor.

If you are using your own motor, or if a certain data needs to be modified according to your needs, you can check the course《1.2 Control command》 to understand the specific meaning of each command.

```
if __name__ == "__main__":
    try:
        t = 0
        print("please wait...")
        set_motor_parameter() # 设置自己的电机参数  Set your own motor parameters

        while True:
            t += 10
            M1 = t
            M2 = t
            M3 = t
            M4 = t

            if MOTOR_TYPE == 4:
                control_pwm(M1*2, M2*2, M3*2, M4*2)
            else:
                control_speed(M1, M2, M3, M4)#直接发送命令控制电机  Send commands 
directly to control the motor

            if t> 1000 or t < -1000:
                t = 0

            if UPLOAD_DATA == 1:
                now_string = read_all_encoder()  # 读取累计编码器数据   Read the 
accumulated encoder data
                print(now_string)
            elif UPLOAD_DATA == 2:
                offset_string = read_10_encoder()  # 读取实时编码器数据 Read real-
time encoder data
                print(offset_string)
            time.sleep(0.1)
```

In the loop program, the speed of the four motors will be slowly increased from 0 to 1000. If the motor type is 4, that is, the motor without encoder, the motor's PWM is directly controlled.

At the same time, the data sent by the driver board is read and printed out at the same time.

```
# 读取编码器数据    Read encoder data
def read_10_encoder():
    global encoder_offset
    formatted_values = []
    for i in range(4):
        reg = READ_TEN_M1_ENCODER_REG + i
```

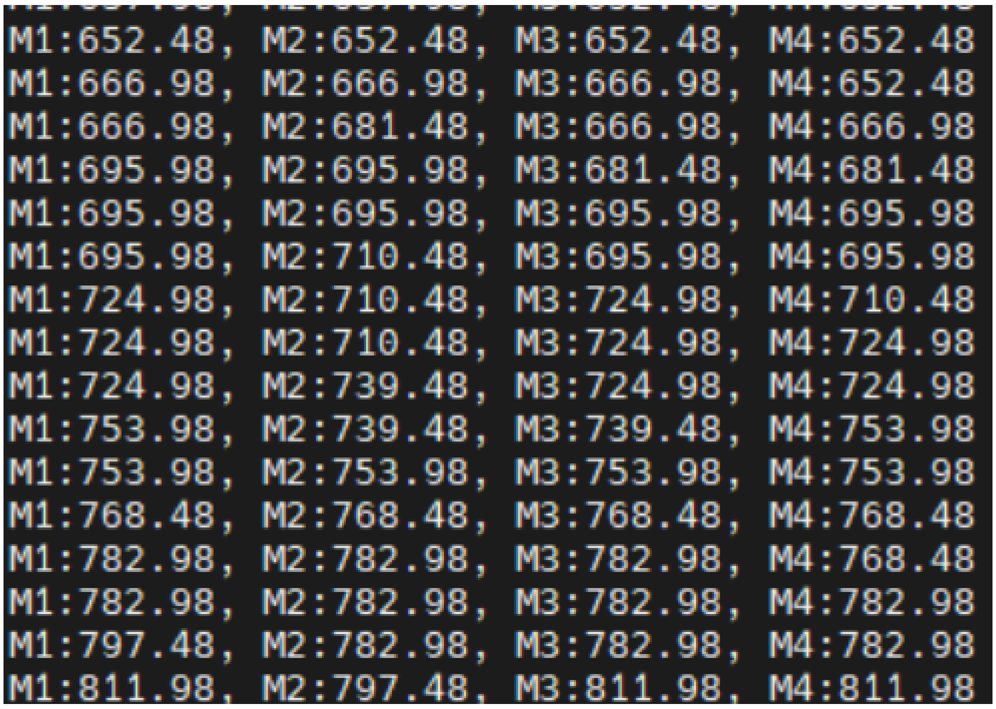
```python
        buf = i2c_read(MOTOR_MODEL_ADDR, reg, 2)
        encoder_offset[i] = (buf[0] << 8) | buf[1]
        if encoder_offset[i] & 0x8000:  # 检查最高位（符号位）是否为 1  Check if the highest bit (sign bit) is 1
            encoder_offset[i] -= 0x10000 # 将其转为负数 Turn it into a negative number
        formatted_values.append("M{}:{}".format(i + 1, encoder_offset[i]))
    return ", ".join(formatted_values)

def read_all_encoder():
    global encoder_now
    formatted_values = []
    for i in range(4):
        high_reg = READ_ALLHIGH_M1_REG + (i * 2)
        low_reg = READ_ALLLOW_M1_REG + (i * 2)
        high_buf = i2c_read(MOTOR_MODEL_ADDR, high_reg, 2)
        low_buf = i2c_read(MOTOR_MODEL_ADDR, low_reg, 2)

        high_val = high_buf[0] <<8 | high_buf[1]
        low_val = low_buf[0] <<8 | low_buf[1]

        encoder_val = (high_val << 16) | low_val

        # 处理符号扩展，假设 32 位有符号整数   Handles sign extension, assuming 32-bit signed integers
        if encoder_val >= 0x80000000:  # 如果大于 2^31，说明应该是负数  If it is greater than 2^31, it should be a negative number
            encoder_val -= 0x100000000  # 将其转为负数  Turn it into a negative number
        encoder_now[i] = encoder_val
        formatted_values.append("M{}:{}".format(i + 1, encoder_now[i]))
    return ", ".join(formatted_values)
```

After getting the data from the driver board, it is shifted, and the shift is required to get the correct data.

1.4 Experimental phenomenon

After the wiring is correct, place the program in the root directory, and then run the command `sudo python ~/IIC.py`. You can see that the motor will gradually speed up, then stop, and repeat.

At the same time, you can see the printed motor value in the terminal constantly changing.



```
M1:652.48, M2:652.48, M3:652.48, M4:652.48
M1:666.98, M2:666.98, M3:666.98, M4:652.48
M1:666.98, M2:681.48, M3:666.98, M4:666.98
M1:695.98, M2:695.98, M3:681.48, M4:681.48
M1:695.98, M2:695.98, M3:695.98, M4:695.98
M1:695.98, M2:710.48, M3:695.98, M4:695.98
M1:724.98, M2:710.48, M3:724.98, M4:710.48
M1:724.98, M2:710.48, M3:724.98, M4:724.98
M1:724.98, M2:739.48, M3:724.98, M4:724.98
M1:753.98, M2:739.48, M3:739.48, M4:753.98
M1:753.98, M2:753.98, M3:753.98, M4:753.98
M1:768.48, M2:768.48, M3:768.48, M4:768.48
M1:782.98, M2:782.98, M3:782.98, M4:768.48
M1:782.98, M2:782.98, M3:782.98, M4:782.98
M1:797.48, M2:782.98, M3:782.98, M4:782.98
M1:811.98, M2:797.48, M3:811.98, M4:811.98
```