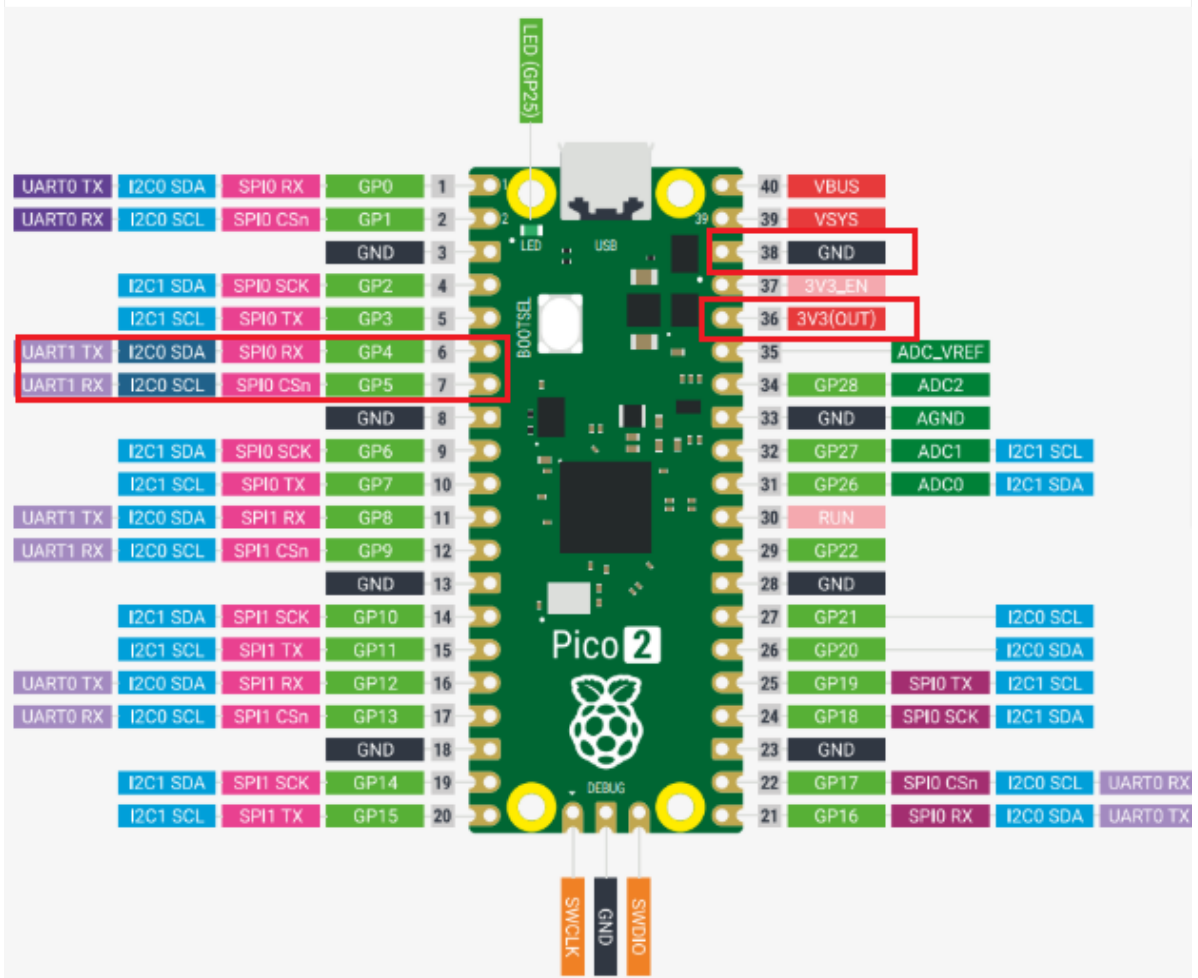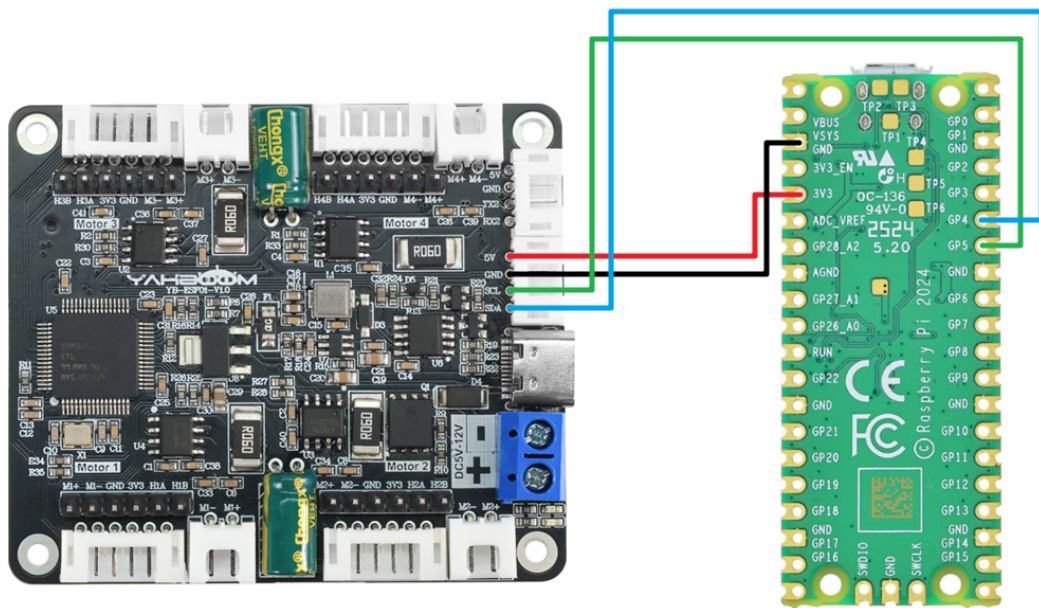# Drive motor and read encoder-IIC

## 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:**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| UART0 TX | I2C0 SDA | SPI0 RX | GP0 | 1 | | 40 | VBUS | | | |
| UART0 RX | I2C0 SCL | SPI0 CSn | GP1 | 2 | | 39 | VSYS | | | |
| | | | GND | 3 | | 38 | GND | | | |
| | I2C1 SDA | SPI0 SCK | GP2 | 4 | | 37 | 3V3_EN | | | |
| | I2C1 SCL | SPI0 TX | GP3 | 5 | | 36 | 3V3(OUT) | | | |
| UART1 TX | I2C0 SDA | SPI0 RX | GP4 | 6 | | 35 | ADC_VREF | | | |
| UART1 RX | I2C0 SCL | SPI0 CSn | GP5 | 7 | | 34 | GP28 | ADC2 | | |
| | | | GND | 8 | | 33 | GND | AGND | | |
| | I2C1 SDA | SPI0 SCK | GP6 | 9 | | 32 | GP27 | ADC1 | I2C1 SCL | |
| | I2C1 SCL | SPI0 TX | GP7 | 10 | | 31 | GP26 | ADC0 | I2C1 SDA | |
| UART1 TX | I2C0 SDA | SPI1 RX | GP8 | 11 | | 30 | RUN | | | |
| UART1 RX | I2C0 SCL | SPI1 CSn | GP9 | 12 | | 29 | GP22 | | | |
| | | | GND | 13 | | 28 | GND | | | |
| | I2C1 SDA | SPI1 SCK | GP10 | 14 | | 27 | GP21 | | I2C0 SCL | |
| | I2C1 SCL | SPI1 TX | GP11 | 15 | | 26 | GP20 | | I2C0 SDA | |
| UART0 TX | I2C0 SDA | SPI1 RX | GP12 | 16 | | 25 | GP19 | SPI0 TX | I2C1 SCL | |
| UART0 RX | I2C0 SCL | SPI1 CSn | GP13 | 17 | | 24 | GP18 | SPI0 SCK | I2C1 SDA | |
| | | | GND | 18 | | 23 | GND | | | |
| | I2C1 SDA | SPI1 SCK | GP14 | 19 | | 22 | GP17 | SPI0 CSn | I2C0 SCL | UART0 RX |
| | I2C1 SCL | SPI1 TX | GP15 | 20 | | 21 | GP16 | SPI0 RX | I2C0 SDA | UART0 TX |

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

| 4-channel motor drive board | Pico2 |
|---|---|
| SDA | GP4 |
| SCL | GP5 |
| GND | GND |
| 5V | 3V3 |

## 1.2 Instructions

Open the code using Thonny software, connect to the Raspberry Pi pico2, and click the stop button on the far right of the upper toolbar.
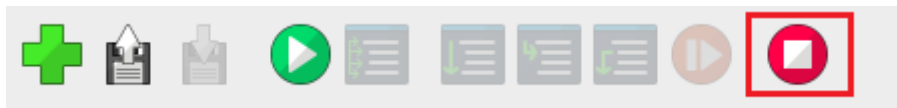


Then, you can see the information of the current firmware in pico2 pop up in the message bar at the bottom, which means that the software has recognized the serial port.

```
Shell ✕
>>>

MicroPython v1.24.0-preview.201.g269a0e0e1 on 2024-08-09; Raspberry Pi Pico2 with RP2350
Type "help()" for more information.
>>>
```

If the message bar here shows that the serial port cannot be recognized, you need to refer to the firmware flashing tutorial and flash the firmware to the motherboard so that the serial port can be recognized.

```
Shell ✕
  Couldn't find the device automatically.
  Check the connection (making sure the device is not in bootloader mode) or choose
  "Configure interpreter" in the interpreter menu (bottom-right corner of the window)
  to select specific port or another interpreter.
```

After successfully identifying the serial port and printing the firmware information, click the green button to start running the script.

# 1.3 Code analysis

```
#define UPLOAD_DATA 2  // 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 data

#define 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.

```python
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(1600)   # 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(1200)
        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(1600)
        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.

```python
# 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
        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, plug the motherboard into the USB port of the computer, and then click the red stop button on the software. Under normal circumstances, you can see the firmware information pop up in the message bar below.

Then, click the green run button, and 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 message bar constantly changing.

```
M1:-6    M2:0    M3:0    M4:0
M1:-28   M2:0    M3:0    M4:0
M1:-23   M2:0    M3:0    M4:0
M1:-18   M2:0    M3:0    M4:0
```