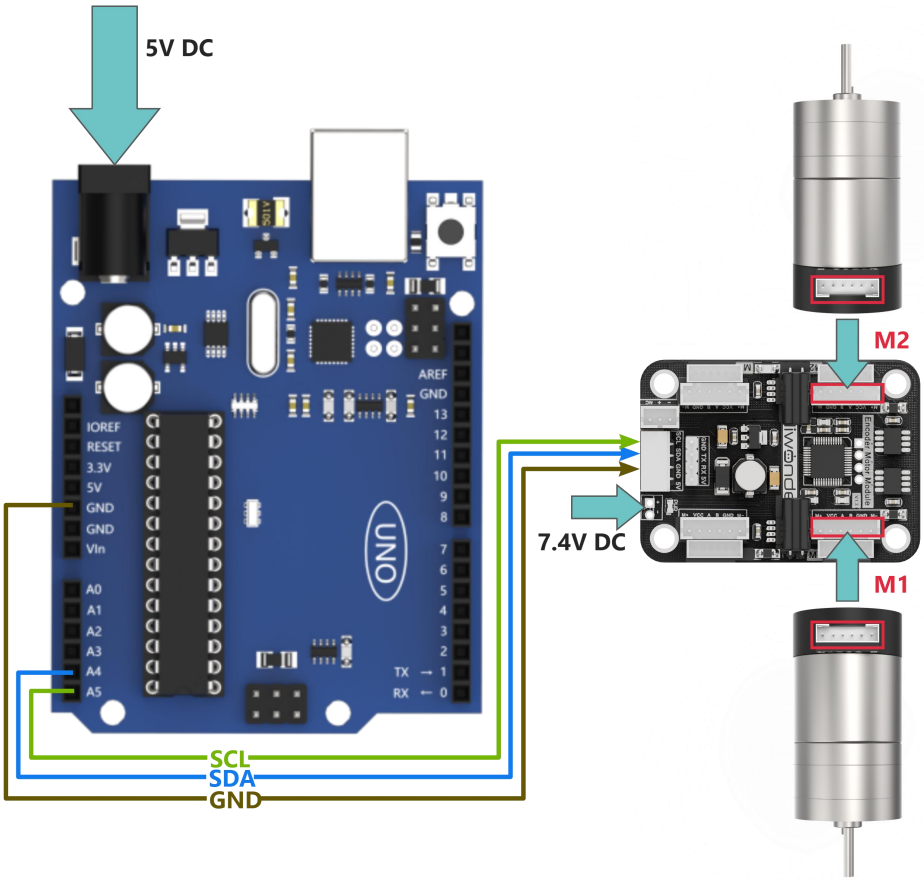
Arduino Development

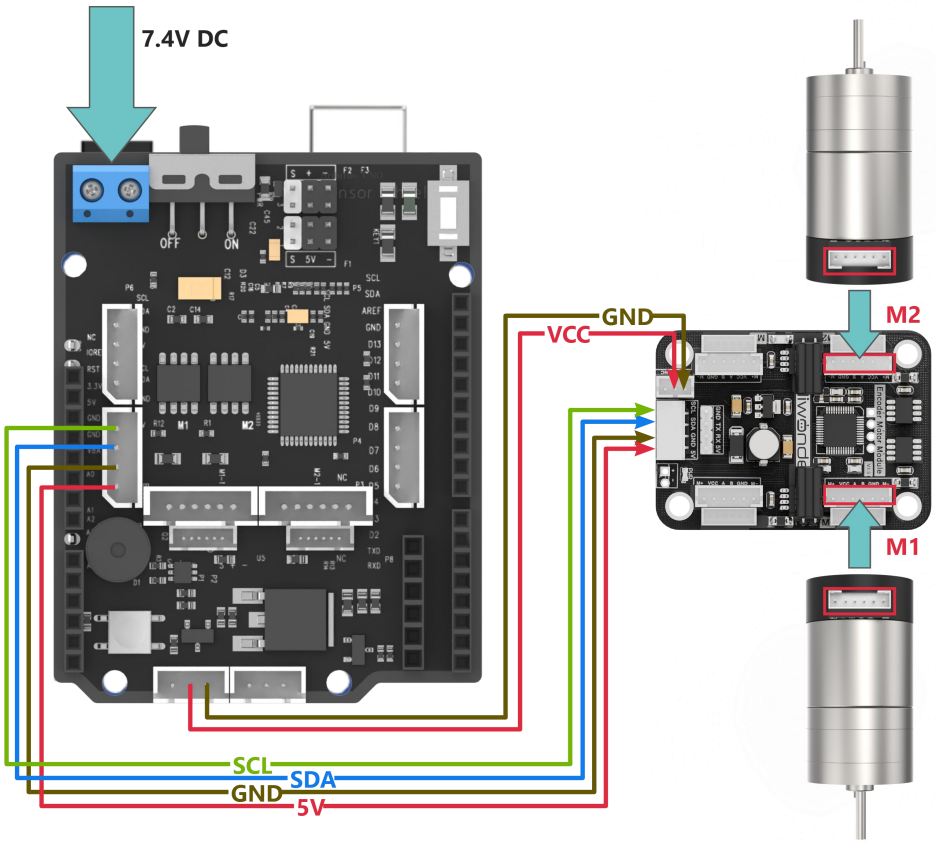
1. Hardware Connection

This lesson employs the Arduino UNO for control demonstration. The specific connection method is shown below:

Option 1: Power the main controller and motor driver module separately



Option 2: Power the main controller and motor driver module using different voltages via a DC-DC buck converter.



Software Preparation:

Install and debug the Arduino IDE, based on the instruction in "Set Development Environment" in this folder.

Note:

1) Please solder the pins on the power interface. Connect the power supply via a male-to-female connector, with 5V to 5V and GND to GND.

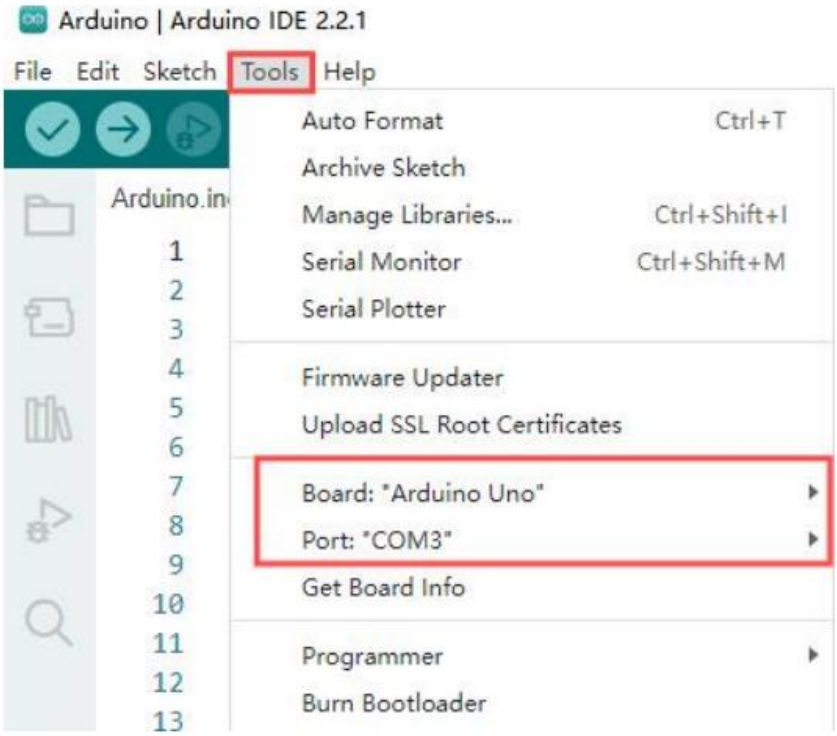
2) This diagram only shows the connection of one motor. If you want to connect more motors, you can refer to this wiring method.

2. Program Download

1) Connect the Arduino UNO to your computer with a USB cable.

2) Open the Arduino program located in the same folder as this lesson.

3) Select the correct development board and port. This lesson uses UNO as 12 an example, with the port number being 3. Please select them based on the actual situation.



4) Click  to upload the program to the development board.

3. Program Outcome

After powering on, Arduino will print the power voltage of the 4-ch encoder motor driver on the serial port. The motors will be controlled to rotate clockwise for 3 seconds and counterclockwise for 3 seconds. Then, the total accumulated pulse value of the four encoder motors will be printed on the serial port. Next, the motors will stop. The total pulse value of all encoder

motors will be reset

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**

The Wire library is used for I2C communication between the Ard uino and the 4-ch encoder motor driver.

```
#include <Wire.h>
```

- **Initialize communication address**

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.

```
#define I2C_ADDR        0x34

#define ADC_BAT_ADDR                0x00
#define MOTOR_TYPE_ADDR             0x14 //编码电机类型设置(set encoder motor type)
#define MOTOR_ENCODER_POLARITY_ADDR  0x15 //设置编码方向极性，(set the encoder direction polarity)
//如果发现电机转速根本不受控制，要么最快速度转动，要么停止。可以将此地址的值重新设置一下(If the motor speed is completely uncontrollable,
//either it rotates at the fastest speed or stops. You can reset the value of this address)
//范围0或1，默认0 (range 0 or 1, default 0)
#define MOTOR_FIXED_PWM_ADDR     0x1F //固定PWM控制，属于开环控制，范围（-100~100）(Fixed PWM control belongs to open loop control. It ranges from -100 to 100)
#define MOTOR_FIXED_SPEED_ADDR    0x33 //固定转速控制，属于闭环控制，(Fixed speed control belongs to closed loop control)
//单位：脉冲数每10毫秒，范围（根据具体的编码电机来，受编码线数，电压大小，负载大小等影响，一般在±50左右）(Unit: pulse count per 10 milliseconds,
//range (depending on the specific encoder motor, affected by the number of encoding lines, voltage size, load size, etc., generally around ±50))

#define MOTOR_ENCODER_TOTAL_ADDR  0x3C //4个编码电机各自的总脉冲值(total pulse value of 4 encoder motors)
//如果已知电机每转一圈的脉冲数为U，又已知轮子的直径D，那么就可以通过脉冲计数的方式得知每个轮子行进的距离(If the pulse count per revolution of the motor is known as U,
//and the diameter of the wheel is known as D, then the distance traveled by each wheel can be obtained through pulse counting)
//比如读到电机1的脉冲总数为P，那么行进的距离为(P/U) * (3.14159*D)(For example, if the total pulse count of motor 1 is P, then the distance traveled is (P/U) * (3.14159*D))
//对于不同的电机可以自行测试每圈的脉冲数U，可以手动旋转10圈读出脉冲数，然后取平均值得出(For different motors,
//you can test the pulse count per revolution U by manually rotating 10 times and reading the pulse count, and then take the average value to get)
```

- **Module initialization**

First, the I2C communication is initialized. Then, the serial port is initialized

with a baud rate of 9600. Next, the motor type is initialized. The 4-ch encoder motor driver supports various types of motors, including TTL, N20, and JGB motors. All these specific types are defined. In this case, JGB motors are used. Therefore, the motor model is set to 3. The motor polarity is set to 0 simultaneously. If it is set to 1, the motors will rotate clockwise continuously. This makes it impossible to control via the code.

```
uint8_t MotorType = MOTOR_TYPE_JGB37_520_12V_110RPM; //设定电机类型(set motor type)
uint8_t MotorEncoderPolarity = 0;
void setup()
{
  Wire.begin();//初始化I2C，以Ardunio UNO为例I2C口为：A4(SCL)、A5(CLK)(Initialize I2C, for example, the I2C port is A4 (SCL) and A5 (CLK) for Ardunio UNO)
  Serial.begin(9600);//初始化串口，波特率设置为9600(Initialize the serial port, and set the baud rate to 9600)
  printf_begin();//printf输出初始化(initialize printf output)
  delay(200);
  WireWriteDataArray(MOTOR_TYPE_ADDR,&MotorType,1);//在电机类型地址中写入电机类型编号(write motor type number in the motor type address)
  delay(5);
  WireWriteDataArray(MOTOR_ENCODER_POLARITY_ADDR,&MotorEncoderPolarity,1);

}
```

- **Variable and array definition**

In this case, the I2C communication is mainly called to send data collection. The motor driver module can control the rotation of four motors simultaneously. Encapsulate the rotation speed of the four motors in an array and send it to the motor driver module. After receiving the data, the driver module will control the motor to rotate based on the set speed.

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

```
/*用数组传递电机速度，正数为设置前进速度，负数为设置后退速度(Use an array to pass the motor speed, positive is to set forward speed, negative is to set reverse speed)
  以p1、p2为例：p1=4个电机以50的速度前进    p2=4个电机以20的速度后退(Take p1 and p2 as examples: p1 = 4 motors move forward at a speed of 50; p2 = 4 motors move backward at a speed of 20)*/
int8_t p1[4]={50,50,50,50};
int8_t p2[4]={-20,-20,-20,-20};
int8_t stop[4]={0,0,0,0};
int8_t EncodeReset[16]={0};
int32_t EncodeTotal[4]; //用于暂存电机累积转动量的值，正转递增，反转递减(Used to temporarily store the accumulated rotation of the motor,
//increasing during forward rotation and decreasing during reverse rotation)
```

- **Main function**

In the main function, use the “WireRead DataArray()” function to read the power voltage of the 4-ch encoder motor driver. Call the “WireWriteDataArray( )” function to write the motor control speed. The motors will be controlled to

rotate clockwise for 3 seconds and counterclockwise for 3 seconds. Then, the total accumulated pulse value of the four encoder motors will be printed on the serial port. The “MOTOR_FIXED_PWM_ADDR” represents the PWM speed control address. The“MOTOR_FIXED_SPEED_ADDR” represents the fixed speed control address. The “MOTOR_ ENCODER_TOTAL_ADDR” is read to obtain the total pulse value of each of the 4 encoder motors. Then, write a stop array with four elements of 0 to control the motors to stop rotating. The p1, p2, and stop arrays are used to control the motor speed. The corresponding motor speed value in the array can be adjusted to achieve different motion effects. Next,write 0 to the “MOTOR_ENCODER_TOTAL_ADDR”. This can clear the pulse values of the motors.

```
void loop()
{
  u16 v;//用于暂存电压值(used to temporarily store the voltage value)
  WireReadDataArray(ADC_BAT_ADDR,data,2);//读取电压地址中存放的电压(read the voltage stored in the voltage address)
  v = data[0]+ (data[1]<<8);
  Serial.print("V = ");Serial.print(v);Serial.println("mV     ");//将电压转换为mV(convert voltage to mV)

  /*在电机转速地址中写入电机的转动方向和速度：WireWriteDataArray（转速控制地址，电机转速数组，电机个数）(Write the motor rotation direction and speed to the motor speed address:
  //WireWriteDataArray (speed control address, motor speed array, and number of motors))*/
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,p1,4);  //正转(rotate clockwise)
  delay(3000);

  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,p2,4);  //反转(rotate counterclockwise)
  delay(3000);

  //PWM控制，注意：PWM控制是一个持续控制的过程，若有延时则会打断电机的运行（打印也会有一定的延时）(PWM control; note:
  //PWM control is a continuous control process. If there is a delay, it will interrupt the motor operation (there will also be a certain delay in printing))


  //WireWriteDataArray(MOTOR_FIXED_PWM_ADDR,s1,4);

  WireReadDataArray(MOTOR_ENCODER_TOTAL_ADDR,(uint8_t*)EncodeTotal,16);//读取电机累积转动量(read accumulated motor rotation value)
  /*打印4个电机的累计转动量(print the accumulated rotation of the 4 motors)*/
  printf("Encode1 = %ld  Encode2 = %ld  Encode3 = %ld  Encode4 = %ld  \r\n", EncodeTotal[0], EncodeTotal[1], EncodeTotal[2], EncodeTotal[3]);

  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,stop,4);  //停止(stop)
  delay(2000);
  //清空编码电机的脉冲值，向该地址写入0即可(Clear the pulse value of the encoder motors, and write 0 to the address)
  WireWriteDataArray(MOTOR_ENCODER_TOTAL_ADDR,EncodeReset,16);

  WireReadDataArray(MOTOR_ENCODER_TOTAL_ADDR,(uint8_t*)EncodeTotal,16);
  printf("Encode1 = %ld  Encode2 = %ld  Encode3 = %ld  Encode4 = %ld  \r\n", EncodeTotal[0], EncodeTotal[1], EncodeTotal[2], EncodeTotal[3]);
}
```