Arduino Version

After the program is downloaded, the tank chassis performs actions in the following set order:

① **Forward for 2 seconds.**

② **Back for 2 seconds.**

③ **Turn left for 2 seconds.**

④ **Forward for another 2 seconds.**

⑤ **Stop.**

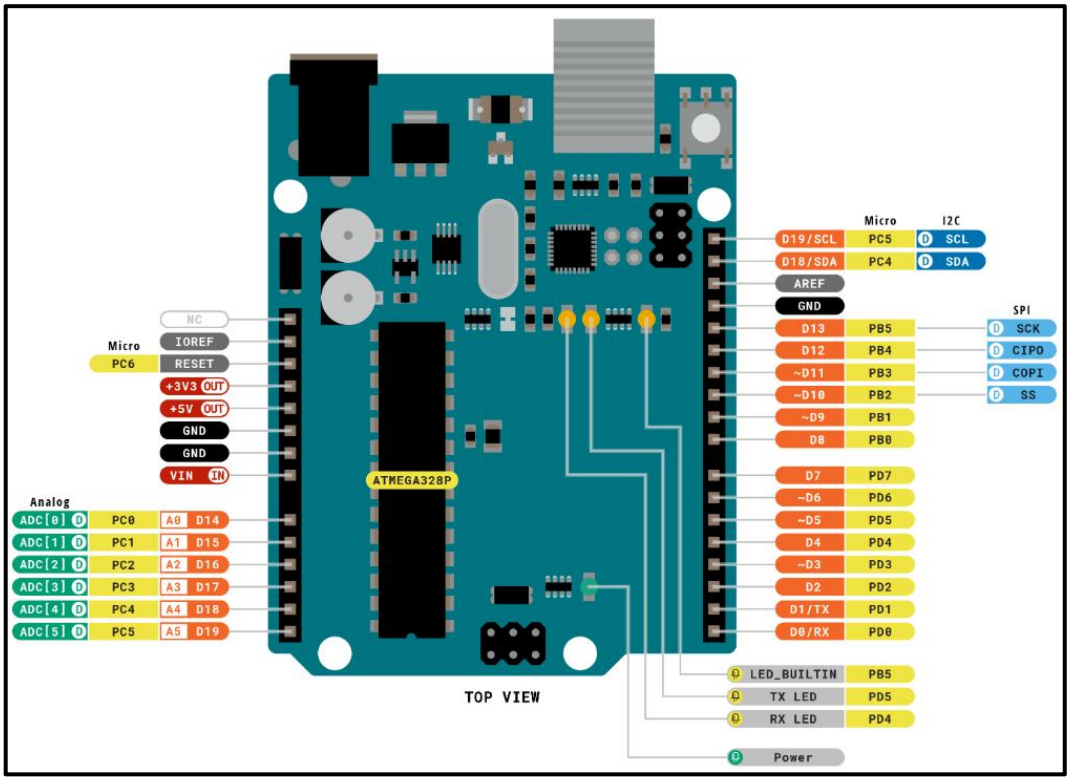
1. Hardware Introduction

1.1 Arduino UNO Controller

Arduino is a convenient, flexible, and user-friendly open-source electronic prototyping platform. It features 14 digital input/output pins (with 6 capable of PWM output), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power socket, an ICSP header, and a reset button.

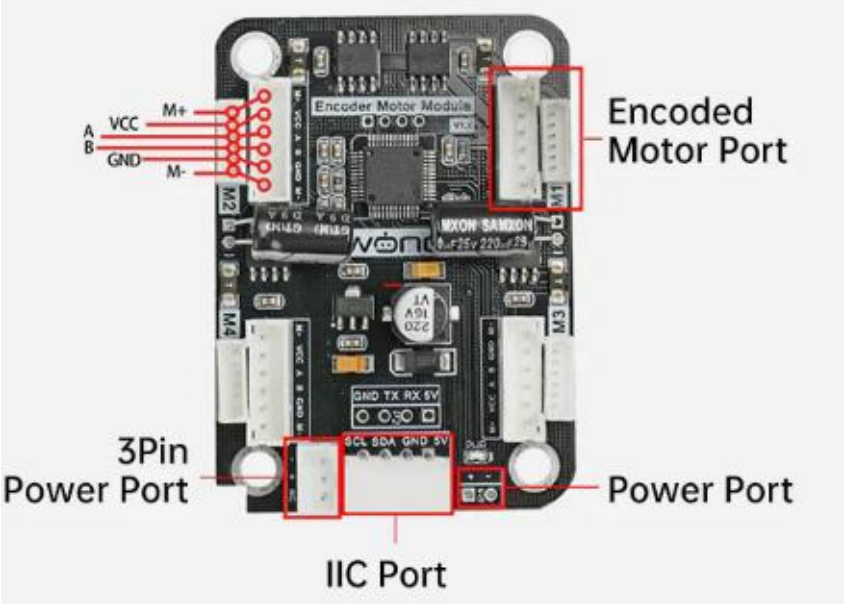
The following diagram illustrates the physical pin layout of Arduino UNO.

Please refer to your specific Arduino UNO controller for accurate details.



1.2 4-Channel Encoder Motor Driver

This is a motor drive module designed to work with a microcontroller for driving TT motors or magnetic encoder motors. Each channel is equipped with a YX-4055AM motor drive chip, and its voltage range is DC 3V-12V. The specific voltage depends on the voltage requirements of the connected motor. The interface distribution is illustrated in the figure below:



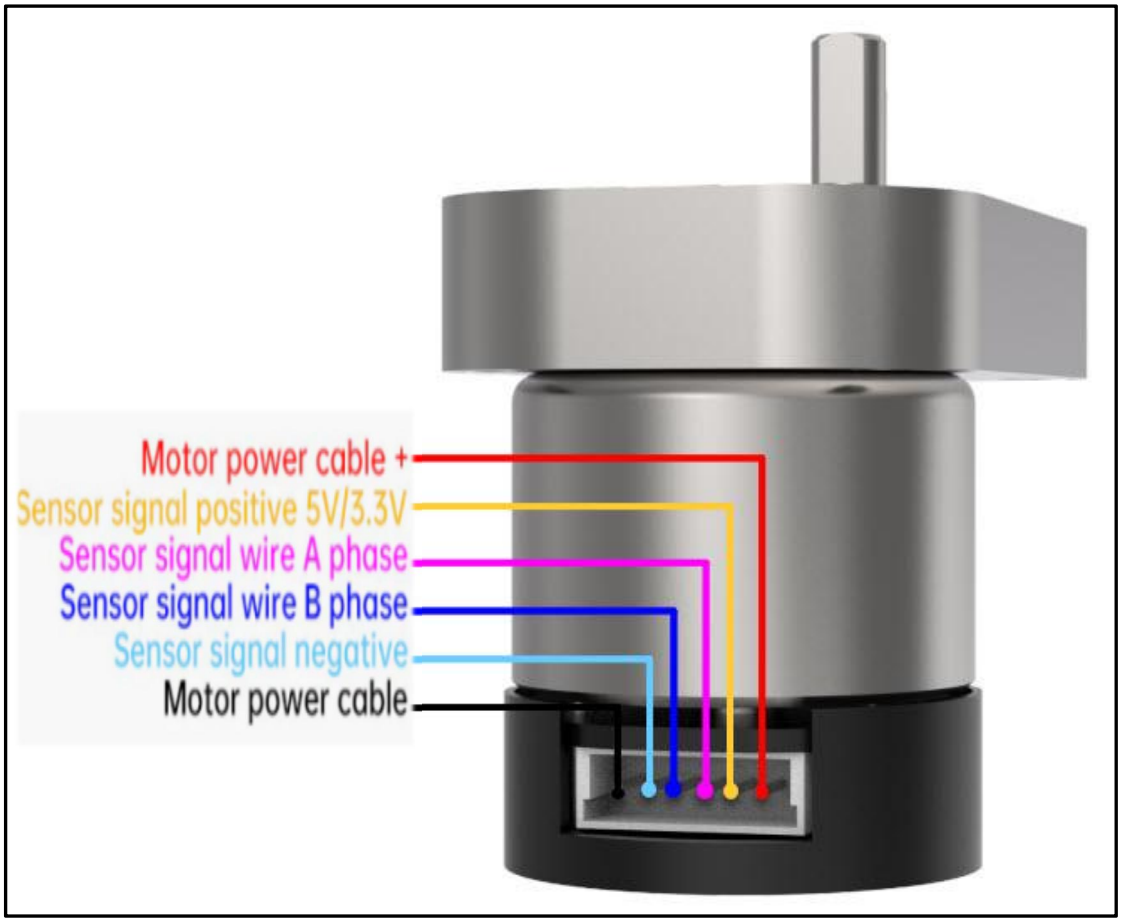
The introduction to the interface on the driver is as below:

<table>
<tr><th>Interface type</th><th>NO.</th><th>Function</th></tr>
<tr><td rowspan="8">Encoder motor interface</td><td>GND</td><td>Negative electrode of the Hall power</td></tr>
<tr><td>A</td><td>A-phase pulse signal output terminal</td></tr>
<tr><td>B</td><td>B-phase pulse signal output terminal</td></tr>
<tr><td>VCC</td><td>Positive electrode of the Hall power</td></tr>
<tr><td>M+</td><td>Positive electrode of the motor power supply</td></tr>
<tr><td>M-</td><td>Positive electrode of the motor power supply</td></tr>
<tr><td colspan="2">Note:
The voltage between VCC and GND is determined based on the power supply voltage of the microcontroller used. Typically, 3.3V or 5V is used.
When the spindle rotates clockwise, the output pulse signal of channel A is ahead of channel B; when the spindle rotates counterclockwise, the signal of channel A is behind channel B.
The voltage between M+ and M- is determined based on the voltage requirements of the motor used.</td></tr>
</table>

IIC	SCL	Clock line
	SDA	Bi-directional data line
	GND	Power ground line
	5V	5V DC output
3Pin power port	-	Power negative electrode
	+	Power positive input
	NC	Empty
Power port	+	Power positive input
	-	Power negative electrode

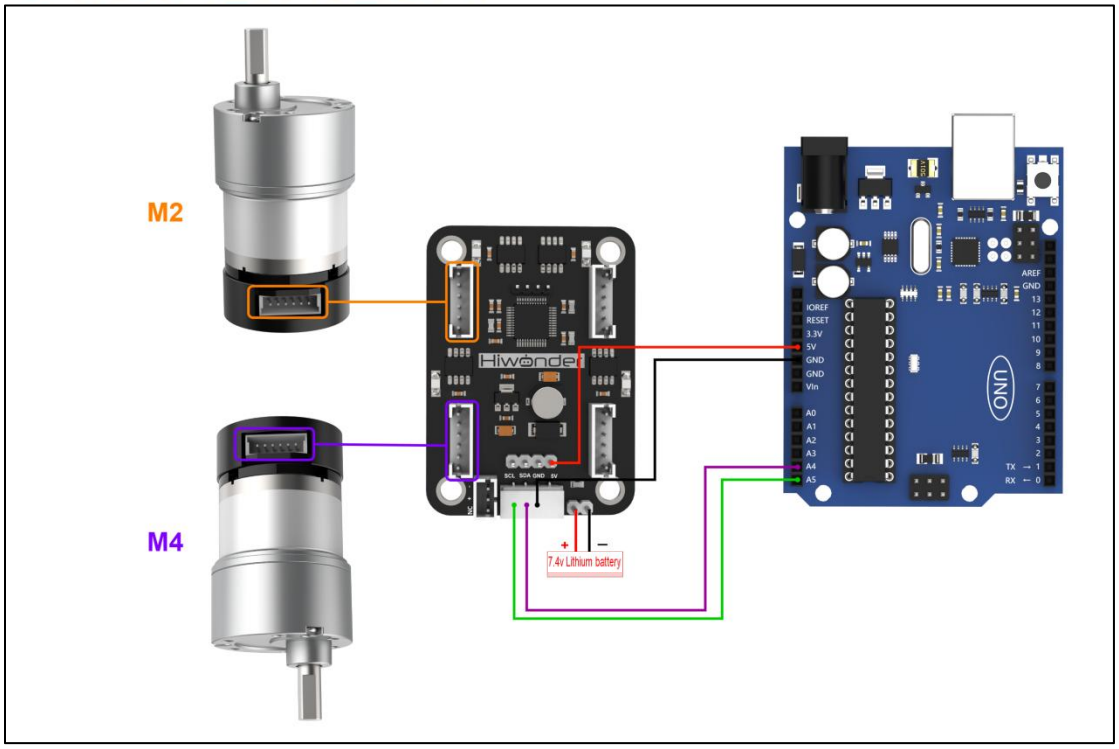
1.3 Encoder Geared Motor

The motor model employed in this chassis is JGB3865-520R45-12. Here's the breakdown: "J" signifies a DC motor, "GB" denotes an eccentric output shaft, "38" indicates the diameter of the reduction box, "520" represents the motor model, "R45" stands for the reduction ratio of 1:45, and "12" signifies the rated voltage of 12V. Please refer to the interface description illustrated in the figure below:



2. Wiring

The Arduino UNO is outfitted with a 4-ch motor driver. It operates using an 11.1V 6000mAh lithium battery to power the motor. The image below is the Arduino UNO wiring diagram. Please refer to the diagram to connect the motor driver module to the steering servo.



3. Environment Configuration and Program Download

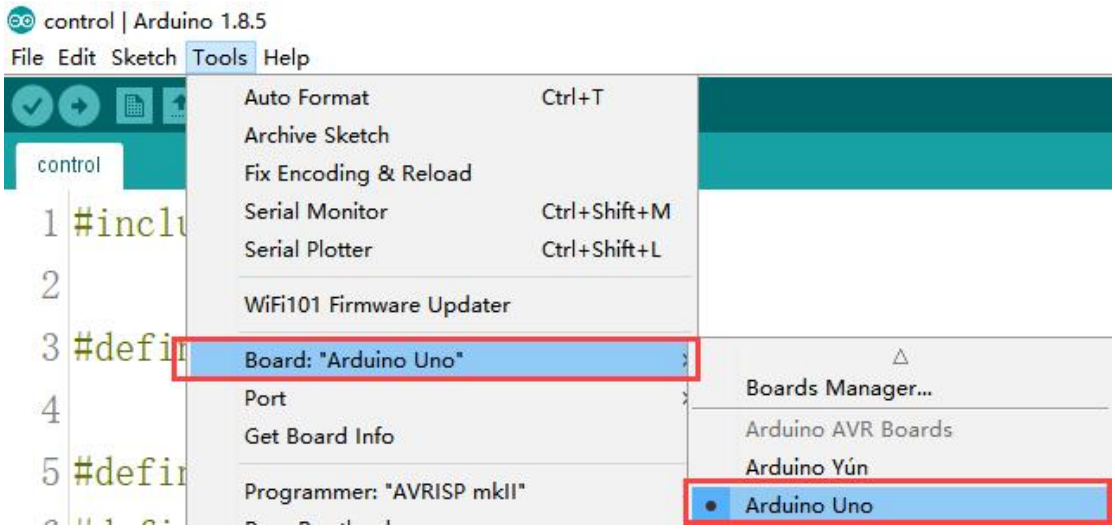
3.1 Environment Configuration

Prior to downloading, ensure that the Arduino IDE is installed on your computer. The software package is located in the “2.Software/2.1 Arduino IDE”.

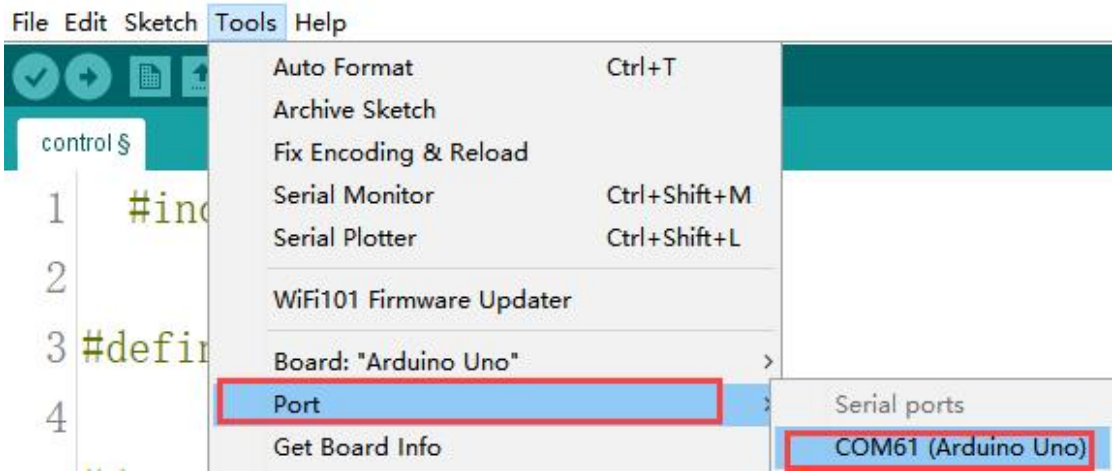
3.2 Program Running

Open the “control.ino” program saved in “3.Program File/Arduino Version” using Arduino IDE.

1) Choose the Arduino development board type. In this case, select “Arduino Uno”.

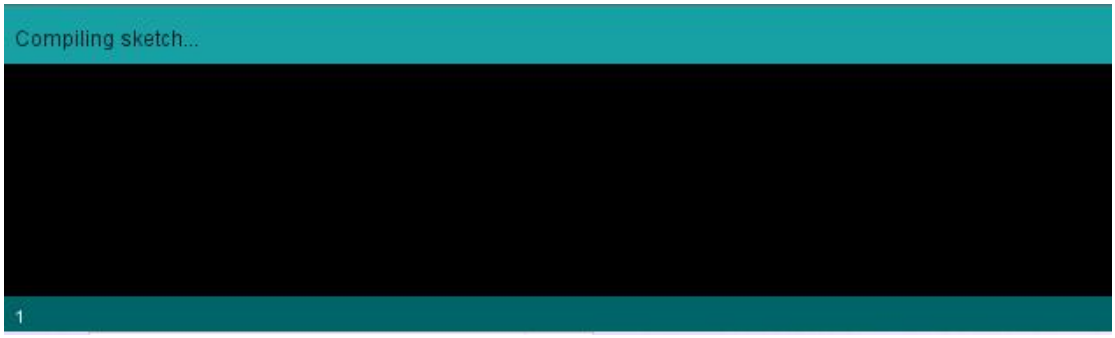


2) Select the USB port the Arduino currently connecting to your computer. The IDE will detect it automatically; in this case, choose “COM56”.

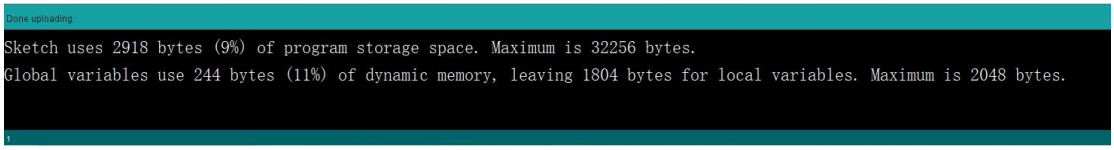


3) Connect Arduino UNO to the computer. Select “Arduino UNO” in the tool bar, and click-on  to download the program.

4) The software will compile the program automatically. Please wait until the compilation process is successfully completed.



5) Wait for the program to finish uploading.



3.3 Program Outcome

Once the program is downloaded, the car chassis executes the following actions in sequence: 1. Move forward for 2 seconds; 2. Move backward for 2 seconds; 3. Turn left for 2 seconds; 4. Move forward for 2 seconds; 5. Stop.

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 Library**

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

The library is integrated into the Arduino IDE. To add it, navigate to “Sketch -> Include Library”. It incorporates write methods for I2C communication, enabling the control of motor rotation.

- **Initialize Communication Address**

```
#include <Wire.h>

#define I2C_ADDR        0x34

#define ADC_BAT_ADDR                    0x00
#define MOTOR_TYPE_ADDR                 0x14 //Set encoder motor type
#define MOTOR_ENCODER_POLARITY_ADDR     0x15 //Set the encoder direction polarity,
//If the motor speed is completely uncontrollable, either rotating at the fastest speed
//Range 0 or 1, default 0
#define MOTOR_FIXED_PWM_ADDR      0x1F //Fixed PWM control, belongs to open-loop contro
//#define SERVOS_ADDR_CMD 40
#define MOTOR_FIXED_SPEED_ADDR    0x33 //Fixed speed control, belongs to closed-loop co
//Unit: pulse count per 10 milliseconds, range (depending on the specific encoder motor

#define MOTOR_ENCODER_TOTAL_ADDR  0x3C //Total pulse value of each of the four encoder
//If the pulse count per revolution of the motor is known to be U, and the diameter of
//For example, if the total pulse count of motor 1 is P, then the distance traveled is
//For different motors, you can test the number of pulses per revolution U by manually
```

Define the I2C communication address and address codes for different types of encoded motors as macros, facilitating subsequent calls. The I2C communication address connected to the driver board is set as 0x34; this value is hardware-specific, and the default can be retained here. The encoded motor type is designated as 0x14, and its direction polarity is defined as 0x15. It's essential to note that these two numbers represent the address positions for writing parameters, not the actual parameter values. These values are hardware-specific, and for simplicity, the default values can be maintained in this context.

- **Initialize Motor Type**

```
//motor type specific values
#define MOTOR_TYPE_WITHOUT_ENCODER        0
#define MOTOR_TYPE_TT                     1
#define MOTOR_TYPE_N20                    2
#define MOTOR_TYPE_JGB37_520_12V_110RPM   3 //Magnetic ring rotates 44 pulses per revolution,
```

The 4-channel motor driver module is versatile and supports different motor types, such as TTL, N20, and JGB motors. Macro definitions are used to specify these types. For this development, JGB motors are employed, and their motor type is macro-defined as 3.

- **I2C Communication Motor Control**

```
bool WireWriteDataArray(  uint8_t reg,uint8_t *val,unsigned int len)    //Send data
{
    unsigned int i;

    Wire.beginTransmission(I2C_ADDR);   //Set the starting address of the task
    Wire.write(reg);    //Write the header address
    for(i = 0; i < len; i++) {
        Wire.write(val[i]);   //Write the content passed in to I2C
    }
    if( Wire.endTransmission() != 0 ) {     //Determine if the write is connected
        return false;
    }

    return true;
}
```

In the above program, the “WireWriteDataArray” function is used to write messages to the motor. In this case, mainly use I2C communication to send a data set. The motor driver module can control the rotation of four motors at the

same time. Encapsulate the rotation speed of the four motors into an array. Send it to the motor driver module. After receiving the data, the driver module controls the motor rotation according to the set speed. Of course, in the tank car chassis, only two motors need to be controlled for rotation, as shown in the following figure.

```
uint8_t MotorType = MOTOR_TYPE_JGB37_520_12V_110RPM;    //Motor mode setting
uint8_t MotorEncoderPolarity = 0;     //Motor polarity setting
void setup()
{
  Wire.begin();
  delay(200);
  WireWriteDataArray(MOTOR_TYPE_ADDR,&MotorType,1);
  delay(5);
  WireWriteDataArray(MOTOR_ENCODER_POLARITY_ADDR,&MotorEncoderPolarity,1);

}
int8_t car_forward[4]={0,23,0,-23};   //Forward
int8_t car_retreat[4]={0,-23,0,23};   //Backward
int8_t car_turnLeft[4]={0,20,0,20}; //Turn left in place
int8_t car_stop[4]={0,0,0,0};
```

In the initialization program, send the motor mode and motor magnetic settings to the motor driver module. Set the speed of the motors. Then, set the speed for forward and backward movement of the chassis to 23, and the turning speed to 20. In this case, only motor 2 and motor 4 are controlled. Set the speed of motors 1 and 3 to 0. You can modify the values based on the actual situation. The positive and negative values represent clockwise and counterclockwise rotation respectively. The absolute value of the number determines the speed of the motor.

- **Main Function**

```
void loop()
{
  /* Car moves forward */
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_forward,4);
  delay(2000);
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_stop,4);
  delay(1000);
  /* Car moves backward */
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_retreat,4);
  delay(2000);
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_stop,4);
  delay(1000);
  /* Car turns left in place */
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_turnLeft,4);
  delay(600);
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_stop,4);
  delay(1000);
  /* Car moves forward in the current direction */
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_forward,4);
  delay(2000);
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_stop,4);
  while(1);
}
```

In the main function, the rotation of the motors can be controlled by sending, delaying, and sending. This achieves the control of the chassis movement.

The “WireWriteDataArray” is used to write the speed of the chassis control. The parameter “MOTOR_FIXED_SPEED_ADDR” can be referred to in the comments of the program. It represents the value of the speed control address. “car_forward”, “car_retreat”, “car_turnLeft”, and “car_stop” represent the control states of the four speeds of the chassis, which are forward, backward, turning left in place, and stopping. The corresponding motor speed values in these parameters can be adjusted to control the chassis for different movement effects.

5. Development Notices

1. Powering the 4-ch encoder motor driver solely through the I2C interface may result in low voltage and unstable motor driving function. Therefore, it is essential to select a suitable power supply based on the rated voltage of the motor.

2. When programming, the voltage difference between the M+ and M- pins of the motor needs to be controlled. This enables the motor's clockwise and counterclockwise rotation.

3. The Arduino UNO board does not integrate a motor driver chip. It can only output a single digital signal. The voltage difference signal pins for the M+ and M- of the encoder motor (two different level signals) cannot be provided. Therefore, it must be used in conjunction with a 4-ch motor driver module to achieve motor clockwise and counterclockwise rotation.

4. The Arduino UNO does not contain the chip required to drive the 4-ch encoder motor driver. Therefore, an additional motor driver chip is needed to control the motor rotation. Our motor driver board can be used, which is equipped with the YX-4055AM driver chip that can drive the motor to rotate. After I2C communication, the driver chip controls the motor to rotate. For more detailed information, please search the internet.

5. Given that the Arduino UNO operates at a rated working voltage of 5V, the I2C interface (5V, GND, SCL, SDA) on the the motor driver module cannot be directly used for power supply. This 5V interface only supports voltage input, not output. It is recommended to use the 5V pin on the serial port of the motor driver module to power the Arduino. The 5V pin on the serial port can support both input and output of 5V voltage.