Arduino Version

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

① **Move forward and backward.**

② **Move horizontally left and right.**

③ **Turn left and right in place.**

④ **Move horizontally left front and right rear.**

⑤ **Drift left and right in place.**

Each action is executed for 2 seconds, with a 1-second interval between different actions.

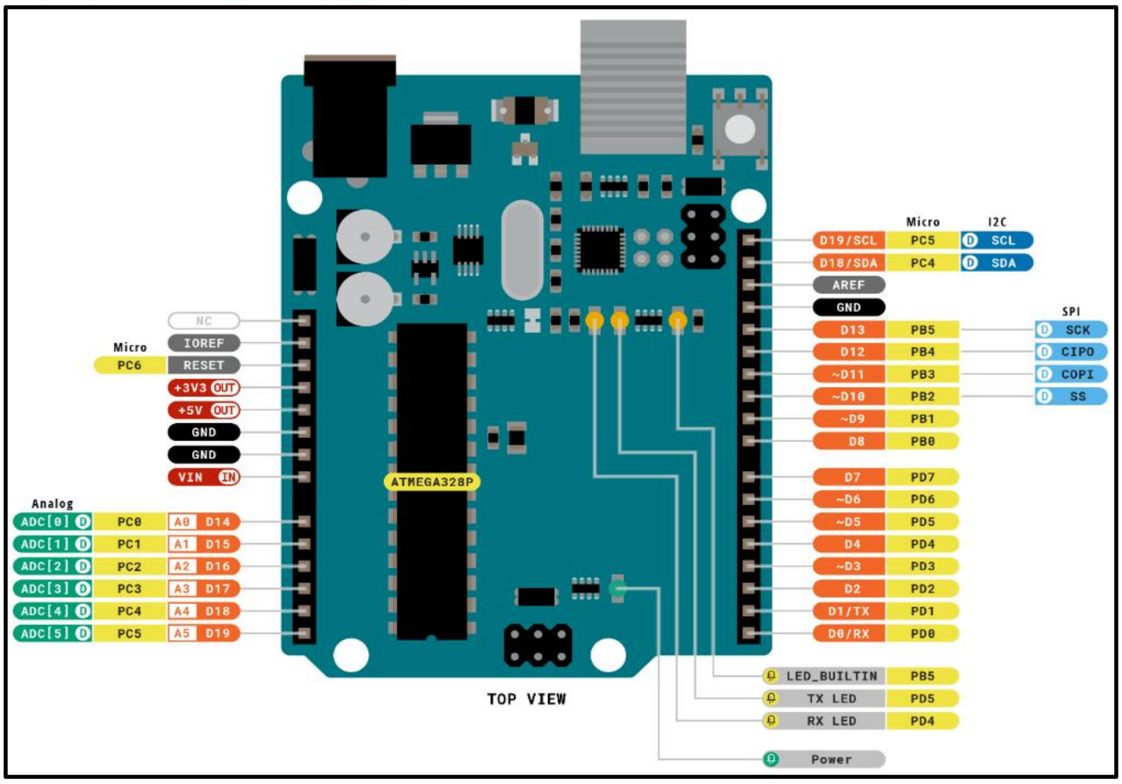
1. Hardware Introduction

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.

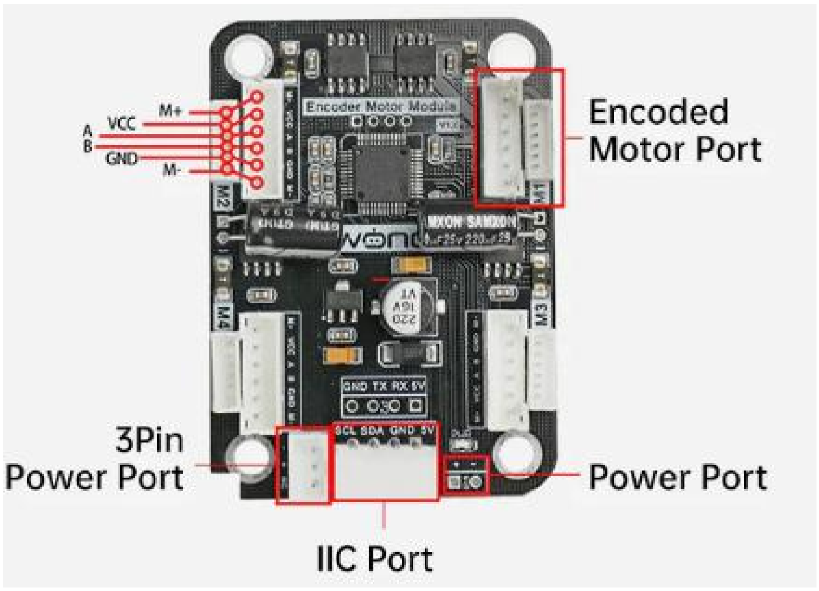
The following diagram illustrates the physical pin layout of Arduino UNO.

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 an SA8339 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:



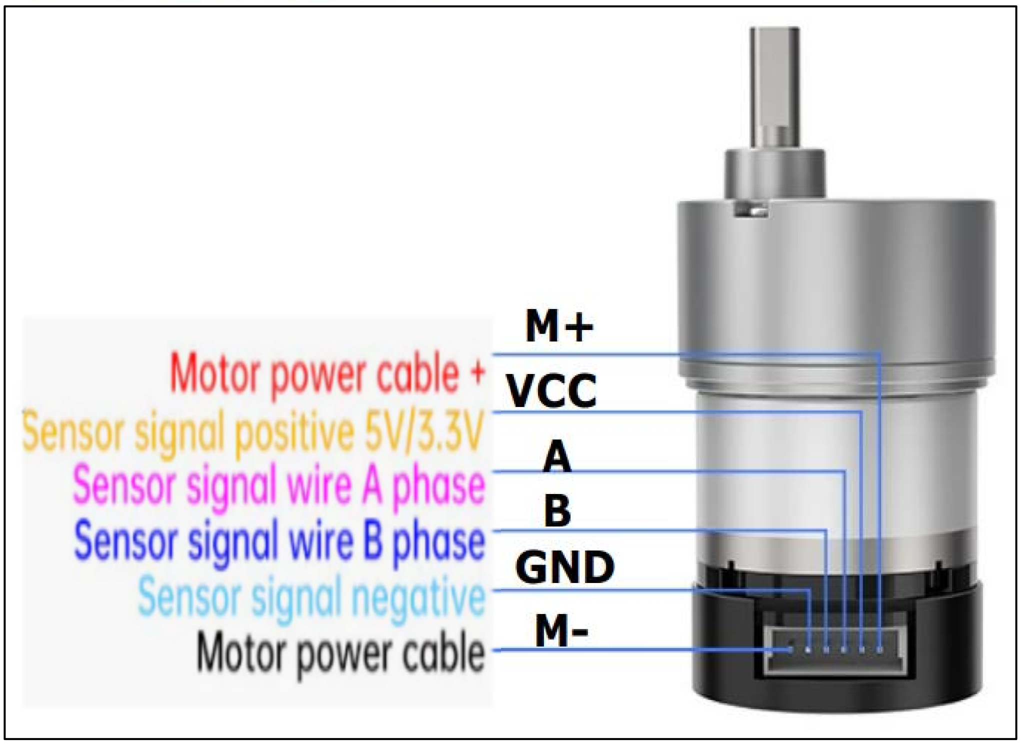
The introduction to the interface on the driver is as below:

Interface type	NO.	Function
Encoder motor interface	GND	Negative electrode of the Hall power
	A	A-phase pulse signal output terminal
	B	B-phase pulse signal output terminal
	VCC	Positive electrode of the Hall power
	M+	Positive electrode of the motor power supply
	M-	Positive electrode of the motor power supply
	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.	

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

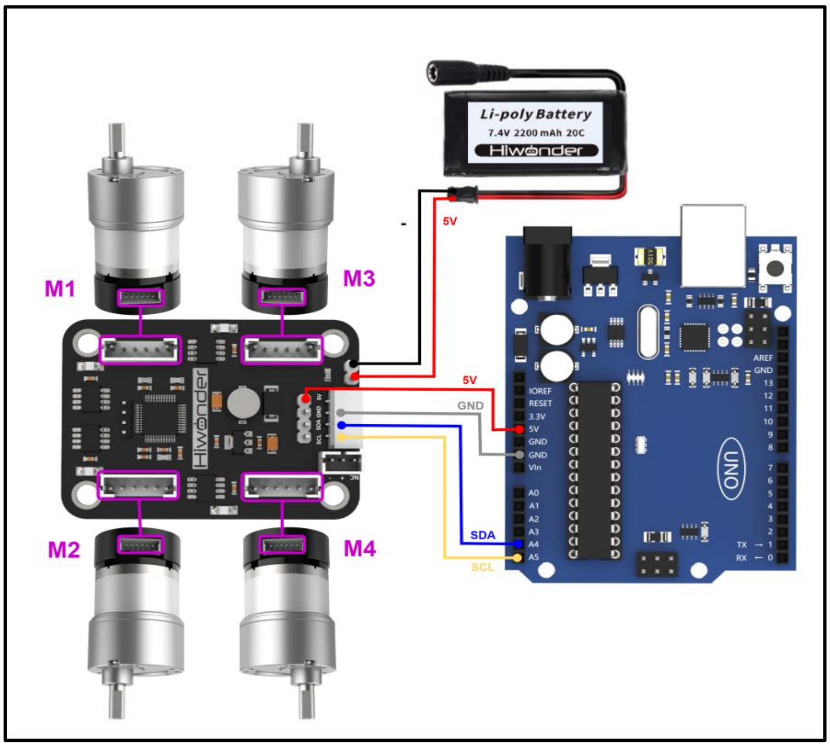
1.3 Encoder Geared Motor

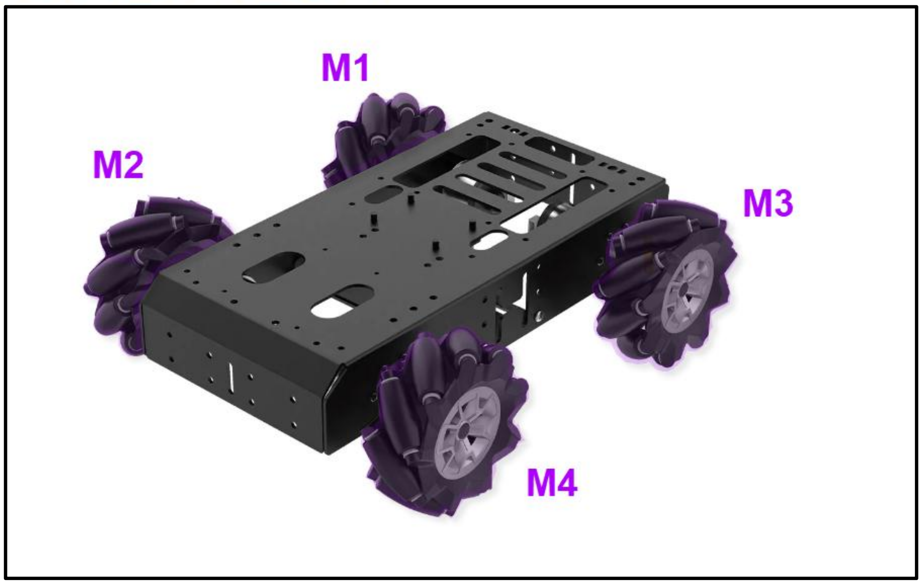
The motor model employed in this chassis is JGB37-520R131-12. Here's the breakdown: "J" signifies a DC motor, "GB" denotes an eccentric output shaft, "37" indicates the diameter of the reduction box, "520" represents the motor model, "R131" stands for the reduction ratio of 1:131, and "12" signifies the rated voltage of 12V. Please refer to the interface description illustrated in the figure below:


Motor power cable +
M+
Sensor signal positive 5V/3.3V
VCC
Sensor signal wire A phase
A
Sensor signal wire B phase
B
Sensor signal negative
GND
Motor power cable -
M-

2. Wiring

The Arduino UNO is outfitted with a 4-ch motor driver. It operates using an 7.4V 2200mAh lithium battery to power the motor. The image below is the Arduino UNO wiring diagram.





3. Environment Configuration and Program Download

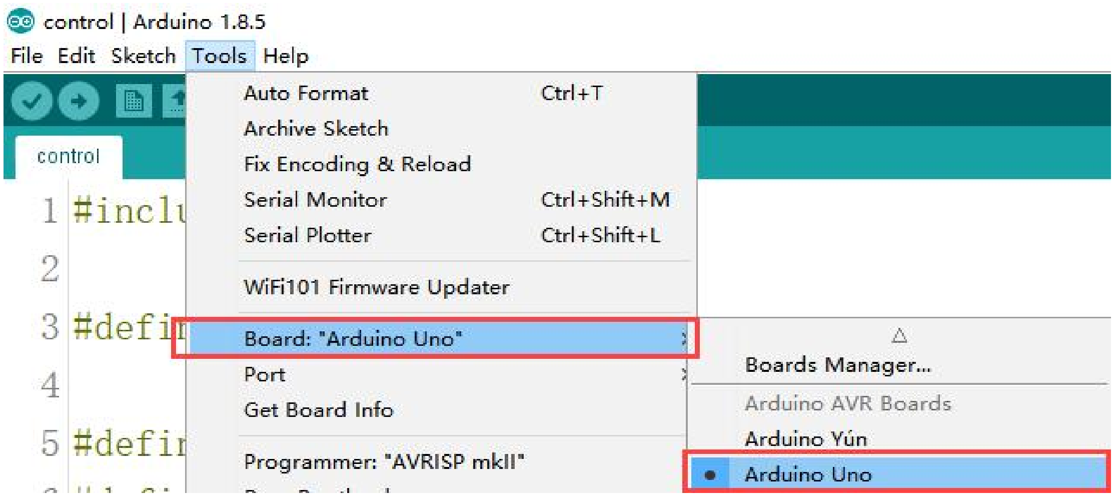
3.1 Environment Configuration

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".

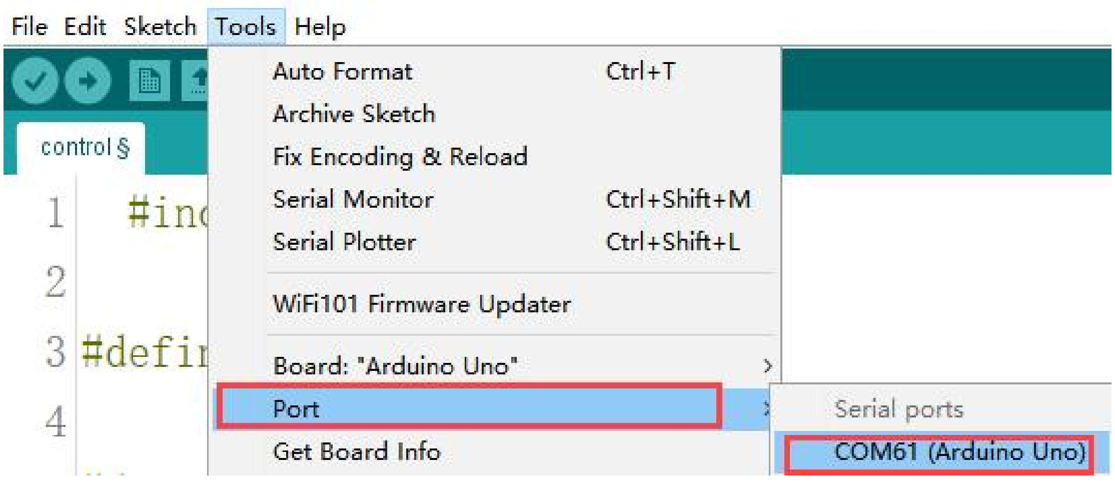
3.2 Program Running

Open the "car_move_demo.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".

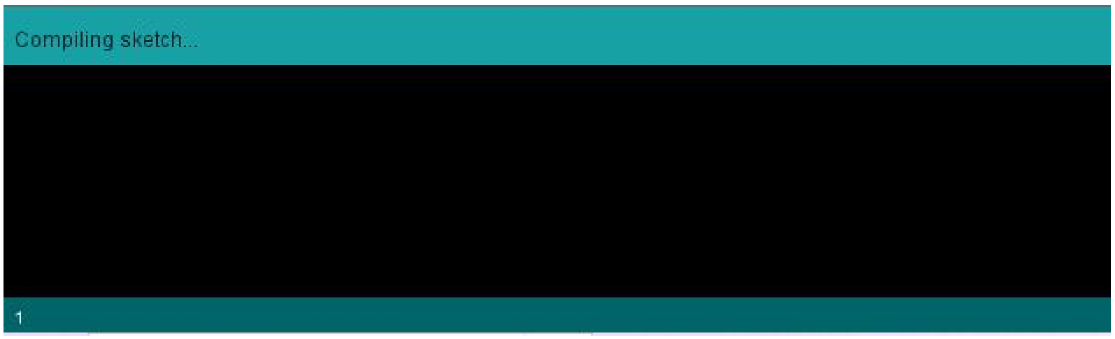


2) Select the USB port the Arduino currently connecting to your computer. The IDE will detect it automatically; in this case, choose “COM56”.

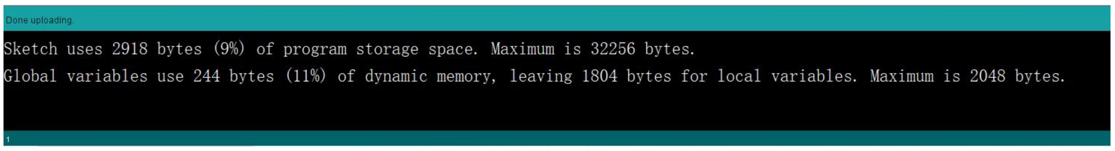


3) Connect Arduino UNO to the computer. Select “Arduino UNO” in the tool bar, and click-on ➔ to download the program.

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.



```
Done uploading.
Sketch uses 2918 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 244 bytes (11%) of dynamic memory, leaving 1804 bytes for local variables. Maximum is 2048 bytes.
```

3.3 Program Outcome

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

① Move forward and backward.

② Move horizontally left and right.

③ Turn left and right in place.

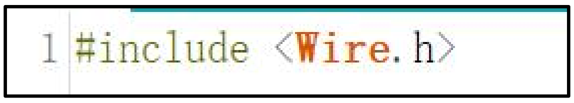
④ Move horizontally left front and right rear.

⑤ Drift left and right in place.

Each action is executed for 2 seconds, with a 1-second interval between different actions.

4. Program Analysis

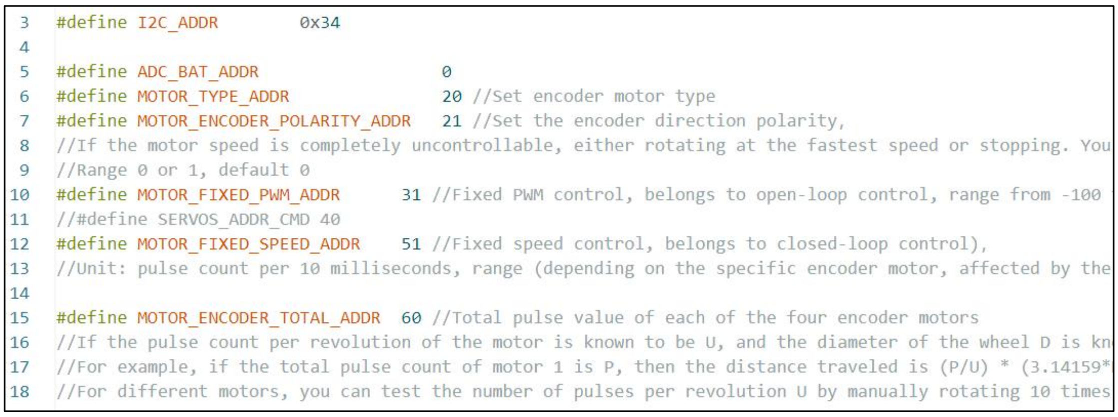
- **Import Necessary Library**



```
#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**



```
#define I2C_ADDR        0x34

#define ADC_BAT_ADDR                  0
#define MOTOR_TYPE_ADDR               20 //Set encoder motor type
#define MOTOR_ENCODER_POLARITY_ADDR   21 //Set the encoder direction polarity,
//If the motor speed is completely uncontrollable, either rotating at the fastest speed or stopping. You
//Range 0 or 1, default 0
#define MOTOR_FIXED_PWM_ADDR      31 //Fixed PWM control, belongs to open-loop control, range from -100
//#define SERVOS_ADDR_CMD 40
#define MOTOR_FIXED_SPEED_ADDR    51 //Fixed speed control, belongs to closed-loop control),
//Unit: pulse count per 10 milliseconds, range (depending on the specific encoder motor, affected by the

#define MOTOR_ENCODER_TOTAL_ADDR  60 //Total pulse value of each of the four encoder motors
//If the pulse count per revolution of the motor is known to be U, and the diameter of the wheel D is kn
//For example, if the total pulse count of motor 1 is P, then the distance traveled is (P/U) * (3.14159*
//For different motors, you can test the number of pulses per revolution U by manually rotating 10 times
```

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 address is designated as 20. It's essential to note that these two numbers represent the address positions for writing parameters, not the actual parameter values. Their values are hardware-specific, and the default can be retained here.

In the subsequent program, the "MOTOR_FIXED_SPEED_ADDR" I2C address is used to control the motor rotation. According to the address writing for speed control of the 4-ch encoder motor driver, it is set to 51. If you need to control the speed of the motors on the car chassis in the subsequent process, set it to 51 when writing to the IIC interface. Similarly, keep the default here.

- **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_JGB                   3
```

The 4-ch 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.

- **Set Motor and Control**

```
uint8_t MotorType = MOTOR_TYPE_JGB;
uint8_t MotorEncoderPolarity = 0;

void setup()
{

  Wire.begin();
  delay(200);
  WireWriteDataArray(MOTOR_TYPE_ADDR,&MotorType,1);
  delay(5);
  WireWriteDataArray(MOTOR_ENCODER_POLARITY_ADDR,&,1);
  delay(2000);
}
```

“MotorType = MOTOR_TYPE_JGB” and “MotorEncoderPolarity = 0” define the type of the motor and the polarity of the encoder.

”MOTOR_TYPE_JGB” is a predefined motor type constant. It represents a specific motor model JGB. “MotorEncoderPolarity” is set to 0, indicating that the polarity of the encoder is the default setting.

”Wire.begin()” indicates the start of communication via the I2C bus. Call this function to initialize it.

“WireWriteDataArray(MOTOR_TYPE_ADDR, &MotorType, 1)” and

“WireWriteDataArray(MOTOR_ENCODER_POLARITY_ADDR, &MotorEncoderPolarity, 1)” send the motor type and encoder polarity settings to the specified address via the I2C protocol.

- **Set Motion Action**

```
int8_t car_forward[4]={30,-30,-30,30};                    // forward
int8_t car_back[4]={-30,30,30,-30};                       // backward

int8_t car_left_transform[4]={-30,-30,-30,-30};           // move horizontally to the left
int8_t car_right_transform[4]={30,30,30,30};              // move horizontally to the right


int8_t car_turn_left[4]={30,-30,-10,10};                  // turn left
int8_t car_turn_right[4]={10,-10,-30,30};                 // turn right

int8_t car_turn_left_back[4]={-30,30,10,-10};             // left reverse
int8_t car_turn_right_back[4]={-10,10,30,-30};            // right reverse

int8_t car_turn_right_orin[4]={-30,30,-30,30};            // turn left in place
int8_t car_turn_left_orin[4]={30,-30,30,-30};             // turn right in place

int8_t car_Ob_translation_left_up[4]={0,-30,-30,0};       // move horizontally to the left front
int8_t car_Ob_translation_right_up[4]={30,0,0,30};        // move horizontally to the right front

int8_t car_Ob_translation_left_back[4]={-30,0,0,-30};     // 左move horizontally to the left rear
int8_t car_Ob_translation_right_back[4]={0,30,30,0};      // move horizontally to the right rear

int8_t car_drift_left[4]={-30,0,-30,0};                   // drift left
int8_t car_drift_right[4]={30,0,30,0};                    // drift right
```

In the above codes, the initial values of the motor speed required for the motor to move are set. The comment indicates which action corresponds to each parameter setting. Multiple actions are defined here for you to call. Take the “car_forward” action as an example. The “car_forward[4] = {30, -30, -30, 30}” indicates the set speeds of M1, M2, M3, and M4. The motors may be installed in reverse, it is necessary to set the initial rotation direction for forward and reverse based on the motor's rotation direction. Similarly, the speed settings for other actions are the same. It is recommended to set the speed range to [-50, 50].

- **Chassis Motion Execution Function**

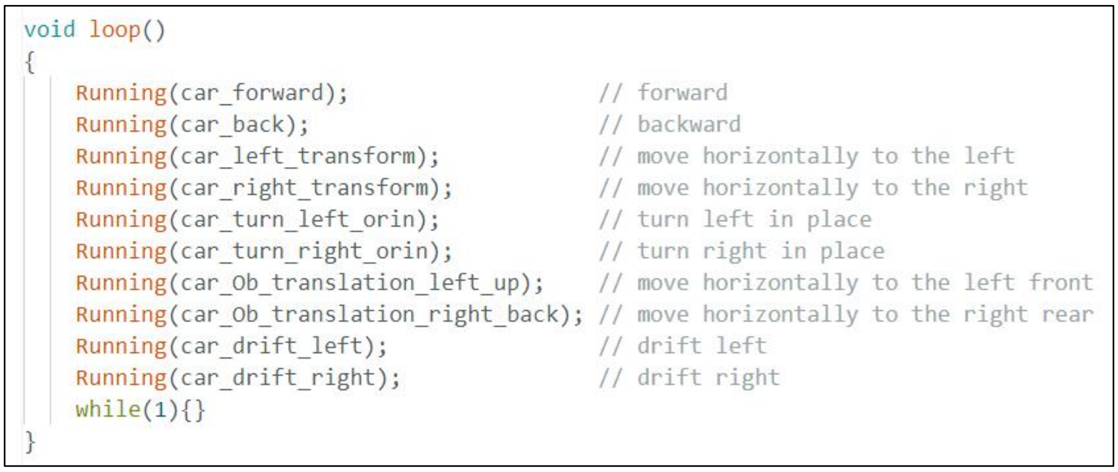
```
void Running(int8_t running_mode[4])
{
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,running_mode,4);  //execute corresponding action
  delay(2000);
  WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR,car_stop,4);      // stop
  delay(1000);
}
```

The “Running” function is used to execute the movement of the car chassis.

“running_mode[4]” is the passed-in motion mode. The corresponding parameters are set for each action in the previous action setting section. Different actions can be executed by passing in different motion mode parameters.

The “WireWriteDataArray(MOTOR_FIXED_SPEED_ADDR, running_mode, 4)” function is used to write the current speed of the four motors. “MOTOR_FIXED_SPEED_ADDR” is the address set for the motor to move. “running_mode” is the corresponding passed-in action parameter (the operating speed of the four motors). After delaying for 2000 milliseconds with "delay(2000)", the stop action "car_stop" is executed for 1 second.

- **Main Function**



```
void loop()
{
    Running(car_forward);                    // forward
    Running(car_back);                       // backward
    Running(car_left_transform);             // move horizontally to the left
    Running(car_right_transform);            // move horizontally to the right
    Running(car_turn_left_orin);             // turn left in place
    Running(car_turn_right_orin);            // turn right in place
    Running(car_Ob_translation_left_up);     // move horizontally to the left front
    Running(car_Ob_translation_right_back);  // move horizontally to the right rear
    Running(car_drift_left);                 // drift left
    Running(car_drift_right);                // drift right
    while(1){}
}
```

In the main function, the previously written “Running” function is executed. The corresponding action execution parameters are passed in to achieve different motion effects. The functions are executed in order from top to bottom. You can modify the passed-in parameters of different action execution functions. This allows you to obtain different motion effects.

5. Development Notices

1) Given that the Arduino UNO operates at a rated working voltage of 5V, the input voltage range is 7 to 12V. However, the I2C interface on the 4-ch motor driver module cannot be used directly to power the Arduino UNO. The 5V pin

of the I2C interface only supports voltage input, not output. Similarly, there are no additional sensors and component loads on the Mecanum wheel. Therefore, 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.

2) During the process of self-development, if there is no strong or weak protection circuit for the positive and negative input/output interfaces on the 4-ch encoder motor driver, do not directly connect them to the power supply positive and negative poles on the controller. This helps prevent the resistor, capacitor, or chip of the controller from burning.