Motor Control

Control the 4 motors connected to the Robduino expansion board.

Device connection

Hardware connection

Use Type-B data cable to connect Arduino Uno and computer.

Software connection

Open the "Arduino IDE" software and select the model and serial port number corresponding to the development board.

Control motor

The motor is directly driven by the underlying driver chip on the Robduino expansion board. The underlying driver chip and the Arduino Uno board use I2C communication, and the I2C address is 0x40.

Control principle

The angle of motor rotation is controlled by changing the duty cycle of the PWM signal that controls the motor signal pin.

Common parameters

Period: The duration of a complete PWM waveform;

Duty cycle: The ratio of the duration of the high level to the cycle time;

Frequency: The reciprocal of the period is called the frequency, that is, the number of PWM cycles generated per second;

```
Set the period of the PWM control signal to 20ms, that is, the frequency of 50Hz; the high level time of the
pulse determines the speed of the motor rotation
```

Control pins

Peripheral module	I2C register first address number (STC8GK2K64S4)	Control motor direction
L1 (upper left corner motor)	11	Control forward (forward)
L1 (upper left corner motor)	10	Control reverse (backward)
L2 (lower left corner motor)	8	Control forward (forward)
L2 (lower left corner motor)	9	Control reverse (backward)
R1 (upper right corner motor)	13	Control forward rotation (forward)
R1 (upper right corner motor)	12	Control reverse rotation (backward)
R2 (lower right corner motor)	14	Control forward rotation (forward)
R2 (lower right corner motor)	15	Control reverse rotation (backward)

Code analysis

Here we only briefly introduce the code content. For detailed code, please refer to the corresponding code file, which is provided in the download area!

- Include `Wire` and `Adafruit_PWMServoDriver` libraries

```
#include <Wire.h>                     // 包含Wire(I2C)通讯库 Include Wire library
#include <Adafruit_PWMServoDriver.h>  // 包含Adafruit PWMServoDriver库 Include Adafruit PWMServoDriver library
```

- Define the motor control pins, I2C address, motor frequency and initial speed

```
// 定义电机控制引脚 Define motor control pins
#define Motor_L1_F_PIN 11  // 控制小车左前方电机前进 Control the motor on the left front of the car
#define Motor_L1_B_PIN 10  // 控制小车左前方电机后退 Control the motor back on the left front of the car
#define Motor_L2_F_PIN 8   // 控制小车左后方电机前进 Control car left rear motor forward
#define Motor_L2_B_PIN 9   // 控制小车左后方电机后退 Control the car left rear motor back
#define Motor_R1_F_PIN 13  // 控制小车右前方电机前进 Control the right front motor of the car to move forward
#define Motor_R1_B_PIN 12  // 控制小车右前方电机后退 Control the motor back on the right front of the car
#define Motor_R2_F_PIN 14  // 控制小车右后方电机前进 Control car right rear motor forward
#define Motor_R2_B_PIN 15  // 控制小车右后方电机后退 Control car right rear motor back

// 定义底层芯片I2C设备地址 Define the IIC address of the Bottom-layer driver chip
#define Bottom_Layer_Driver_ADDR 0x40

// 定义PWM频率 Define PWM frequency
#define PWM_FREQUENCY 200

// 定义电机速度 Define motor speed
int MotorSpeed = 50;
```

- Create an instance of the Adafruit_PWMServoDriver class

```
// 创建Adafruit_PWMServoDriver类的实例 Create an instance of the Adafruit_PWMServoDriver class
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(Bottom_Layer_Driver_ADDR);
```

- Initialization Code

```
void setup() {
  Wire.begin();                    // 初始化I2C通讯 Initialize I2C communication
  delay(1000);                     // 如果小车功能异常，可以增加这个延时 If the function is abnormal, you can
increase the delay
  pwm.begin();                     // PWM初始化 Initialize the Pulse Width Modulation (PWM) library
  pwm.setPWMFreq(PWM_FREQUENCY);   // 设置PWM频率 Set the PWM frequency
}
```

- Looping code

```
void loop() {
  // 控制小车左前方电机前进500毫秒 Control car left front motor forward 500 milliseconds
  setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, MotorSpeed);
  delay(500);
  setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);

  // 控制小车左后方电机前进500毫秒 Control car left rear motor forward 500 milliseconds
  setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, MotorSpeed);
  delay(500);
  setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);

  // 控制小车右前方电机后退500毫秒 Control car right front motor back 500 milliseconds
  setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -MotorSpeed);
  delay(500);
  setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);

  // 控制小车右后方电机后退500毫秒 Control car right rear motor back 500 milliseconds
  setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -MotorSpeed);
  delay(500);
  setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
  delay(5000);
}
```

Experimental results

After compiling the program successfully, upload the code to the Arduino Uno development board.

After the program is started, the car will repeat the following pattern: the first motor on the left side of the car will rotate forward for 500 milliseconds and then stop; the second motor on the left side will rotate forward for 500 milliseconds and then stop; the first motor on the right side will rotate reversely for 500 milliseconds and then stop; the second motor on the right side will rotate reversely for 500 milliseconds and then stop; all motors stop for 5 seconds.

Note: The drive motor needs an external battery pack to drive normally. In addition, it is recommended that the user set up the car to avoid damaging the car!

```
The burning program cannot use other programs to occupy the serial port or an external serial communication
module (for example: WiFi camera module), otherwise the program cannot be burned or an error message will be
prompted!
```