# Electric PTZ control



## 1. Introduction

**Please read the "Introduction to Motors and Usage" in the four-way motor driver board information first to understand the motor parameters, wiring methods, and power supply voltage you are currently using. To avoid burning the motherboard or motor.**

## 2. Experimental preparation

Wheeled car chassis V1 four-wheel drive version, 4*310 motors, 7.4V lithium battery, two-dimensional electric PTZ, MSPM0 robot expansion board (optional), MSPM0G3507 core board (Yahboom).

### The relationship between the 4 motor interfaces and the car is as follows:

M1 -> upper left motor (left front wheel of the car)

M2 -> lower left motor (left rear wheel of the car)

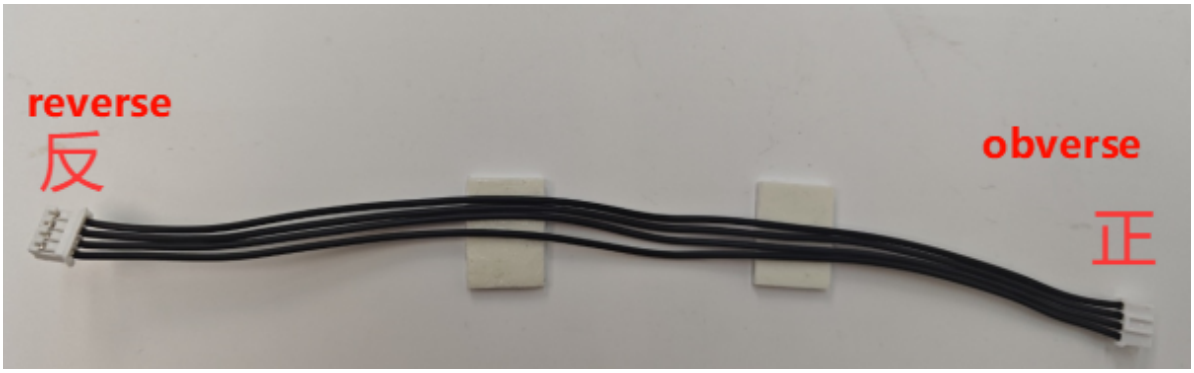
M3 -> upper right motor (right front wheel of the car)

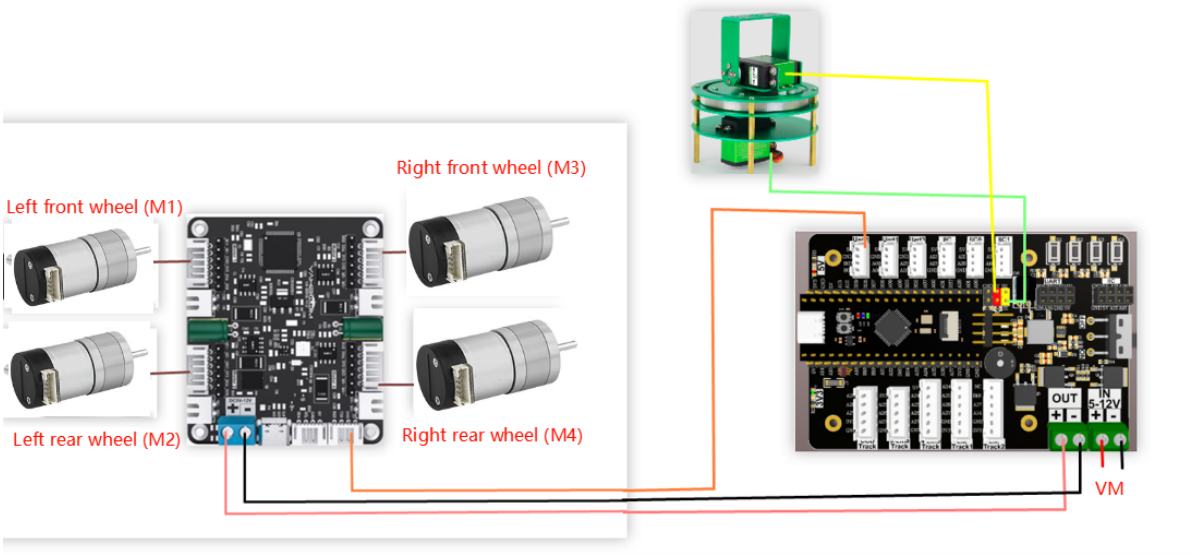
M4 -> lower right motor (right rear wheel of the car)

### Hardware wiring:

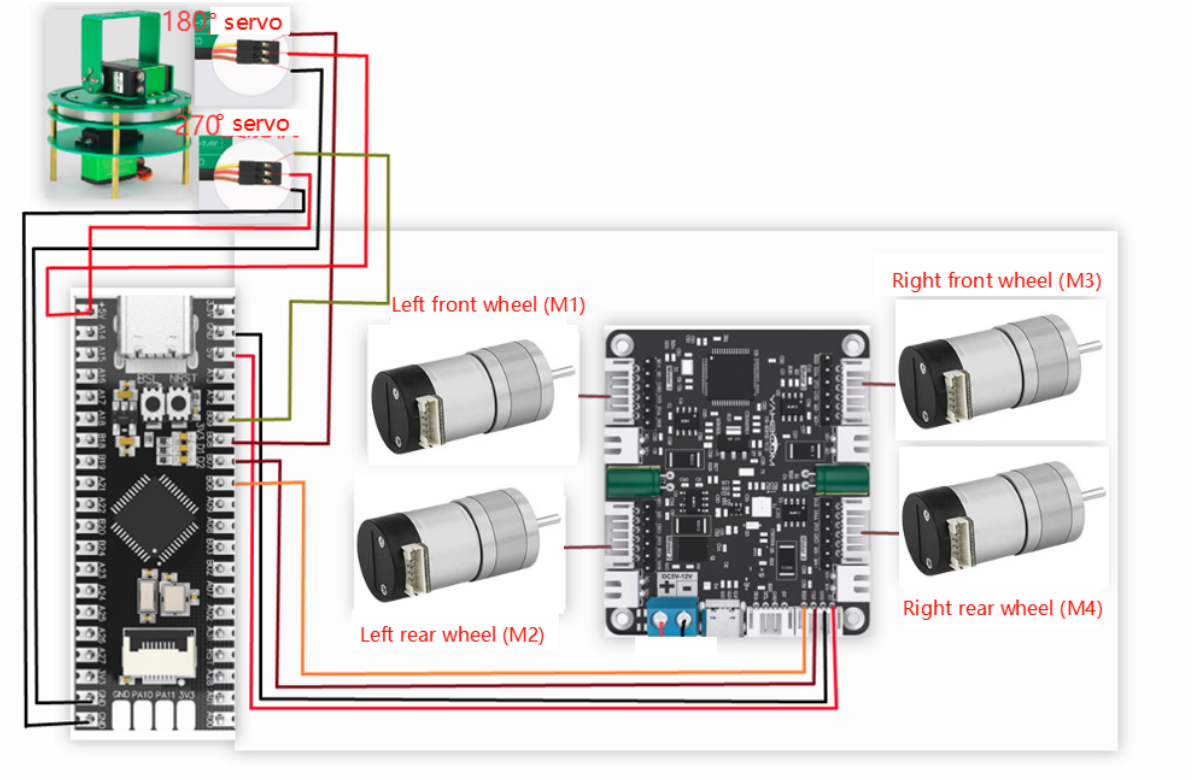
#### Wiring using MSPM0 robot expansion board

**Note: The wire used to connect the MSPM0 robot expansion board to the four-way motor drive module is: XPH2.0-4pin cable, double-ended all black, reverse (200mm), the direction of the reverse cable holder is shown in the figure below**




## Wiring using MSPM0G3507 core board (Yahboom)



### Wiring pins

| Four-way motor driver board | MSPM0G3507 core board (Yahboom) |
| --- | --- |
| RX2 | PB6 |
| TX2 | PB7 |
| GND | GND |
| 5V | 5V |

Take M1 motor as an example below, and other motors are similar

| Motor | Four-way motor driver board (Motor) |
| --- | --- |
| M2 | M1- |
| VCC | 3V3 |
| A | H1A |
| B | H1B |
| GND | GND |
| M1 | M1+ |

| 180° servo | MSPM0G3507 core board (Yahboom) |
| --- | --- |
| SIG | PB8 |
| VCC | 5V |
| GND | GND |

| 270° servo | MSPM0G3507 core board (Yahboom) |
| --- | --- |
| SIG | PB9 |
| VCC | 5V |
| GND | GND |

# 3. Key code analysis

- bsp_servo.c

```
/**************************************************************************
 * 函 数 名 称：Set_Servo_Angle
 * 函 数 说 明：设置270角度
 * 函 数 形 参：angle=要设置的角度，范围0-270
 * 函 数 返 回：无
 * Function Name: Set_Servo_Angle
 * Function Description: Set 270 angles
 * Function Form Parameters: angle=The angle to be set, range 0-270
```

```
 * Function Return: None
**************************************************************************/
void Set_Servo270_Angle(unsigned int angle)
{
      uint32_t period = 400;

      if(angle > 270)
      {
            angle = 270; // 限制角度在0到270度之间    Limit angle between 0 and 270
degrees
      }

      Servo_Angle270 = angle;

      // 计算PWM占空比  Calculate PWM duty cycle
      // 0.5ms对应的计数 = 10       0.5ms corresponding count = 10
      // 2.5ms对应的计数 = 50       2.5ms corresponding count = 50
      float min_count = 10.0f;
      float max_count = 50.0f;
      float range = max_count - min_count;
      float ServoAngle = min_count + (((float)angle / 270.0f) * range);

      DL_TimerG_setCaptureCompareValue(PWM_Servo_INST, (unsigned int)(ServoAngle
+ 0.5f), GPIO_PWM_Servo_C1_IDX);
}


/**************************************************************************
 * 函 数 名 称：Set_Servo_Angle
 * 函 数 说 明：设置180角度
 * 函 数 形 参：angle=要设置的角度，范围0-180
 * 函 数 返 回：无
 * Function Name: Set_Servo_Angle
 * Function Description: Set 180 angles
 * Function Form Parameters: angle=The angle to be set, range 0-180
 * Function Return: None
**************************************************************************/
void Set_Servo_Angle(unsigned int angle)
{
      uint32_t period = 400;

      if(angle > 180)
      {
            angle = 180; // 限制角度在0到180度之间    Limit angle between 0 and 180
degrees
      }

      Servo_Angle = angle;

      // 计算PWM占空比  Calculate PWM duty cycle
      // 0.5ms对应的计数 = 10       0.5ms corresponding count = 10
      // 2.5ms对应的计数 = 50       2.5ms corresponding count = 50
      float min_count = 10.0f;
      float max_count = 50.0f;
      float range = max_count - min_count;
      float ServoAngle = min_count + (((float)angle / 180.0f) * range);
```

```
    DL_TimerG_setCaptureCompareValue(PWM_Servo_INST, (unsigned int)(ServoAngle
+ 0.5f), GPIO_PWM_Servo_C0_IDX);
}
```

Set_Servo270_Angle: Control the angle of the 270° servo

Set_Servo_Angle: Control the angle of the 180° servo

- app_motor_usart.c

```
//发送电机类型    Transmitter motor type
void send_motor_type(motor_type_t data)
{
    sprintf((char*)send_buff,"$mtype:%d#",data);
    Send_Motor_ArrayU8(send_buff, strlen((char*)send_buff));

}

//发送电机死区    Send motor dead zone
void send_motor_deadzone(uint16_t data)
{
    sprintf((char*)send_buff,"$deadzone:%d#",data);
    Send_Motor_ArrayU8(send_buff, strlen((char*)send_buff));
}

//发送电机磁环脉冲  Send motor magnetic ring pulse
void send_pulse_line(uint16_t data)
{
    sprintf((char*)send_buff,"$mline:%d#",data);
    Send_Motor_ArrayU8(send_buff, strlen((char*)send_buff));
}

//发送电机减速比   Transmitting motor reduction ratio
void send_pulse_phase(uint16_t data)
{
    sprintf((char*)send_buff,"$mphase:%d#",data);
    Send_Motor_ArrayU8(send_buff, strlen((char*)send_buff));
}

//发送轮子直径    Send wheel diameter
void send_wheel_diameter(float data)
{
    sprintf((char*)send_buff,"$wdiameter:%.3f#",data);
    Send_Motor_ArrayU8(send_buff, strlen((char*)send_buff));
}

//控制速度  Controlling Speed
void Contrl_Speed(int16_t M1_speed,int16_t M2_speed,int16_t M3_speed,int16_t
M4_speed)
{

sprintf((char*)send_buff,"$spd:%d,%d,%d,%d#",M1_speed,M2_speed,M3_speed,M4_speed
);
    Send_Motor_ArrayU8(send_buff, strlen((char*)send_buff));
}
```

Configure the motor parameters of the 4-way motor driver board

Contrl_Speed: Control the speed of the 4-way motor

- main.c

```
#define MOTOR_TYPE 2      //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
int main(void)
{
    uint8_t i=0;
    uint16_t j=0;
    SYSCFG_DL_init();

     //清除串口中断标志 Clear the serial port interrupt flag
    NVIC_ClearPendingIRQ(MYUART_INST_INT_IRQN);//打印串口   Print Serial Port
    NVIC_ClearPendingIRQ(UART_1_INST_INT_IRQN);//与四路驱动板通信串口 Serial port for communication with quad driver boards
    //使能串口中断    Enable serial port interrupt
    NVIC_EnableIRQ(MYUART_INST_INT_IRQN);
    NVIC_EnableIRQ(UART_1_INST_INT_IRQN);
    DL_TimerA_startCounter(PWM_Servo_INST);//启动控制舵机定时器  Start control servo timer

    ...

    #elif MOTOR_TYPE == 2
    send_motor_type(2);
    delay_ms(100);
    send_pulse_phase(20);
    delay_ms(100);
    send_pulse_line(13);
    delay_ms(100);
    send_wheel_diameter(48.00);
    delay_ms(100);
    send_motor_deadzone(1600);
    delay_ms(100);

    ...

    #endif

    /*现象：两个舵机都先归中，然后小车前进两秒后停下。接着两个舵机分别转动，最后归中。*/
    /*Phenomenon: Both servos are centered first,
    then the cart moves forward for two seconds and stops.
    Then both servos rotate separately and finally return to the center.*/

    Set_Servo270_Angle(155);//270
    delay_ms(100);
    Set_Servo_Angle(90); //180

    delay_ms(1000);
    Contrl_Speed(300,300,300,300);

    delay_ms(2000);
    Contrl_Speed(0,0,0,0);
```

```
    Set_Servo270_Angle(60);
    delay_ms(1500);
    Set_Servo270_Angle(250);
    delay_ms(1500);
    Set_Servo_Angle(0);
    delay_ms(1500);
    Set_Servo_Angle(170);
    delay_ms(1500);

    Set_Servo270_Angle(155);
    delay_ms(100);
    Set_Servo_Angle(90);

}
```

MOTOR_TYPE: used to set the type of motor used. Modify the corresponding number according to the comments based on the motor you are currently using.

USART_Init: Initialize the serial port for communicating with the 4-way motor driver board

Set_Servo270_Angle: Control the 270° servo to the specified angle

Set_Servo_Angle: Control the 180° servo to the specified angle

# 4. Experimental phenomenon

After connecting the car and burning the program to MSPM0, put the car on the ground and plug in the power supply. After running, the two servos will return to the center first. Then the car will move forward for 2 seconds and then stop. After stopping, the 270° servo and 180° servos will rotate.