8-Channel Tracking

1. Opening Notes

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.

Motor: The case and code take the 310 motor of our store as an example.

2. Experimental Preparation

National Race Chassis V2 four-wheel drive version, 4*310 motors, 7.4V lithium battery, 8-channel patrol module, STM32F103C8T6 core board.

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

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

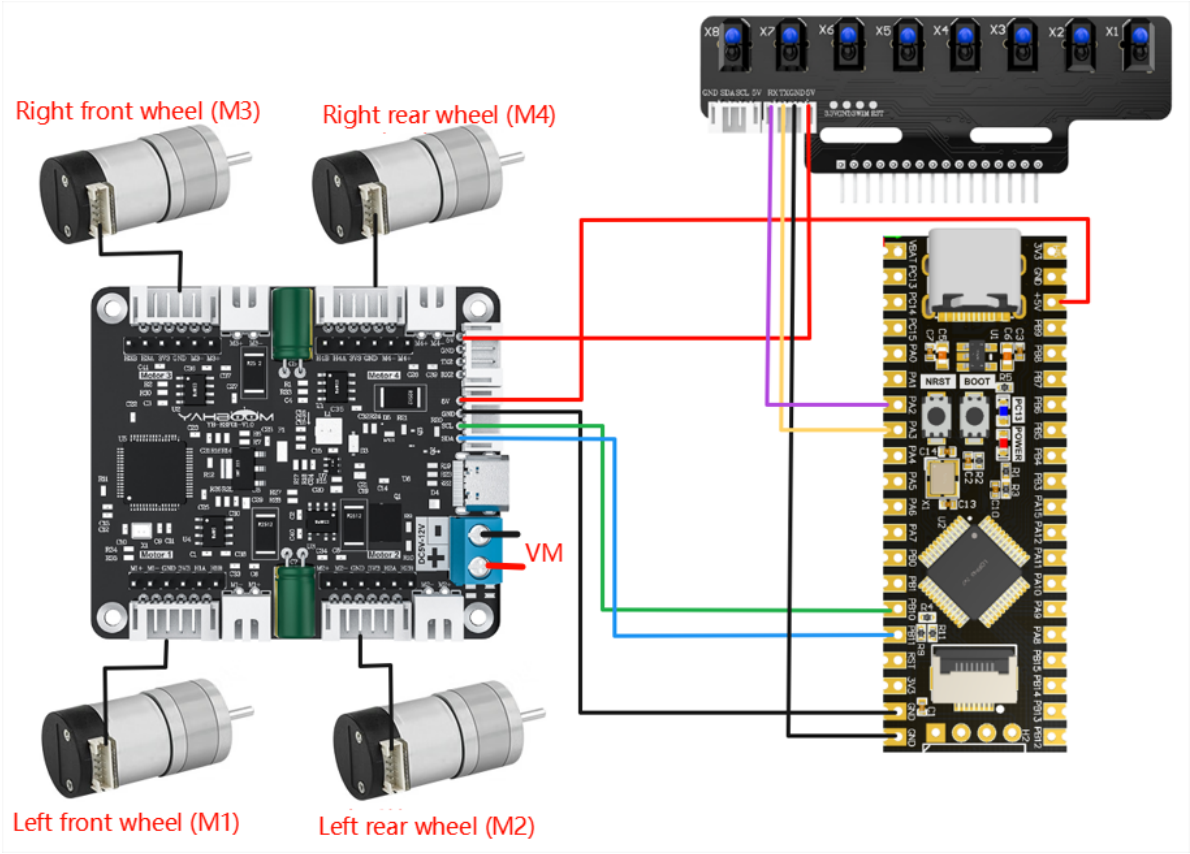
M2 -> lower left motor (left rear wheel of the car)

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

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

Hardware wiring:

Overall wiring



Wiring pins

Four-way motor driver board	STM32C8T6
5V	5V
GND	GND
SCL	PB10
SDA	PB11

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+

8-channel patrol module	STM32C8T6
VCC	5V
GND	GND
TX	PA3
RX	PA2

3. Key code analysis

- app_usart.c

```
void send_control_data(u8 adjust,u8 aData,u8 dData)
{
    u8 send_buf[8] = "$0,0,0#";
    if(adjust == 1)//校准命令   Calibration command
    {
        send_buf[1] = '1';
    }
    else
    {
        send_buf[1] = '0';
    }
    if(aData == 1)//模拟值数据   Analog data
    {
        send_buf[3] = '1';
        g_Amode_Data = 1;
    }
    else
    {
        send_buf[3] = '0';
        g_Amode_Data = 0;
    }
    if(dData == 1)//数字值数据   Digital data
    {
        send_buf[5] = '1';
        g_Dmode_Data = 1;
    }
    else
    {
        send_buf[5] = '0';
        g_Dmode_Data = 0;
    }

    USART2_Send_ArrayU8(send_buf,strlen((char*)send_buf));

}
```

The `send_control_data` function modifies the control command according to the passed parameters and sends it through the `USART2` serial port, controlling whether to send calibration commands, analog value data or digital value data.

- bsp_motor_iic.c

```
//配置电机  Configure the motor
void Set_motor_type(uint8_t data)
{
    i2cWrite(Motor_model_ADDR,MOTOR_TYPE_REG,2,&data);
}

//配置死区  Configuring Dead Zone
void Set_motor_deadzone(uint16_t data)
{
    static uint8_t buf_tempzone[2];

    buf_tempzone[0] = (data>>8)&0xff;
    buf_tempzone[1] = data;

    i2cWrite(Motor_model_ADDR,MOTOR_DeadZONE_REG,2,buf_tempzone);
}

//配置磁环线 Configuring magnetic loop
void Set_Pluse_line(uint16_t data)
{
    static uint8_t buf_templine[2];

    buf_templine[0] = (data>>8)&0xff;
    buf_templine[1] = data;

    i2cWrite(Motor_model_ADDR,MOTOR_PluseLine_REG,2,buf_templine);
}

//配置减速比 Configure the reduction ratio
void Set_Pluse_Phase(uint16_t data)
{
    static uint8_t buf_tempPhase[2];

    buf_tempPhase[0] = (data>>8)&0xff;
    buf_tempPhase[1] = data;

    i2cWrite(Motor_model_ADDR,MOTOR_PlusePhase_REG,2,buf_tempPhase);
}


//配置直径  Configuration Diameter
void Set_Wheel_dis(float data)
{
    static uint8_t bytes[4];

    float_to_bytes(data,bytes);

    i2cWrite(Motor_model_ADDR,WHEEL_DIA_REG,4,bytes);
}
//只能控制带编码器类型的电机 Can only control motors with encoders
//传入参数:4个电机的速度      Input parameters: speed of 4 motors
void control_speed(int16_t m1,int16_t m2 ,int16_t m3,int16_t m4)
{
    static uint8_t speed[8];

    speed[0] = (m1>>8)&0xff;
    speed[1] = (m1)&0xff;
```

```
    speed[2] = (m2>>8)&0xff;
    speed[3] = (m2)&0xff;

    speed[4] = (m3>>8)&0xff;
    speed[5] = (m3)&0xff;

    speed[6] = (m4>>8)&0xff;
    speed[7] = (m4)&0xff;

    i2cWrite(Motor_model_ADDR,SPEED_Control_REG,8,speed);

}
```

Define the function of writing configuration parameters to the four-way motor driver board and the motor control function, which are used to set key parameters such as motor type, dead zone, number of magnetic ring lines, reduction ratio and wheel diameter, and control the speed of the four motors.

- app_motor.c

```
// 返回当前小车轮子轴间距和的一半  Returns half of the current sum of the wheel axle distances
static float Motion_Get_APB(void)
{
    return Car_APB;
}

void Set_Motor(int MOTOR_TYPE)
{
    if(MOTOR_TYPE == 1)
    {
        ...
    }

    else if(MOTOR_TYPE == 2)
    {
        Set_motor_type(2);//配置电机类型  Configure motor type
        delay_ms(100);
        Set_Pluse_Phase(20);//配置减速比 查电机手册得出 Configure the reduction ratio. Check the motor manual to find out
        delay_ms(100);
        Set_Pluse_line(13);//配置磁环线 查电机手册得出  Configure the magnetic ring wire. Check the motor manual to get the result.
        delay_ms(100);
        Set_Wheel_dis(48.00);//配置轮子直径,测量得出      Configure the wheel diameter and measure it
        delay_ms(100);
        Set_motor_deadzone(1900);//配置电机死区,实验得出  Configure the motor dead zone, and the experiment shows
        delay_ms(100);
    }

    ...

}

//直接控制速度   Directly control speed
```

```
void Motion_Car_Control(int16_t V_x, int16_t V_y, int16_t V_z)
{
    float robot_APB = Motion_Get_APB();
    speed_lr = 0;
    speed_fb = V_x;
    speed_spin = (V_z / 1000.0f) * robot_APB;
    if (V_x == 0 && V_y == 0 && V_z == 0)
    {
        control_speed(0,0,0,0);
        return;
    }

    speed_L1_setup = speed_fb + speed_spin;
    speed_L2_setup = speed_fb + speed_spin;
    speed_R1_setup = speed_fb  - speed_spin;
    speed_R2_setup = speed_fb  - speed_spin;

    if (speed_L1_setup > 1000) speed_L1_setup = 1000;
    if (speed_L1_setup < -1000) speed_L1_setup = -1000;
    if (speed_L2_setup > 1000) speed_L2_setup = 1000;
    if (speed_L2_setup < -1000) speed_L2_setup = -1000;
    if (speed_R1_setup > 1000) speed_R1_setup = 1000;
    if (speed_R1_setup < -1000) speed_R1_setup = -1000;
    if (speed_R2_setup > 1000) speed_R2_setup = 1000;
    if (speed_R2_setup < -1000) speed_R2_setup = -1000;


 //printf("%d\t,%d\t,%d\t,%d\r\n",speed_L1_setup,speed_L2_setup,speed_R1_setup,s
peed_R2_setup);

    control_speed(speed_L1_setup, speed_L2_setup, speed_R1_setup,
speed_R2_setup);

}
```

The `Set_Motor` function is initialized according to the motor type (MOTOR_TYPE), including setting the motor type, reduction ratio, magnetic loop, wheel diameter, and motor dead zone. The `Motion_Car_Control` function calculates the speed values of the four motors according to the input forward speed (`V_x`), lateral speed (`V_y`), and rotation speed (`V_z`) to achieve motion control of the car. The function obtains half of the wheel axle spacing of the car through `Motion_Get_APB`, which is used to calculate the rotation speed compensation (`speed_spin`) and adjust the speed difference of the left and right motors. If the speed is all zero, stop the motor; otherwise, calculate and limit the speed value between (-1000,1000), and finally call `control_speed` to control the motor speed to ensure that the car moves in the predetermined direction and speed.

- app_irtracking.c

```
#include "app_irtracking.h"

#define IRTrack_Trun_KP (500)
#define IRTrack_Trun_KI (0)
#define IRTrack_Trun_KD (0)

int pid_output_IRR = 0;
u8 trun_flag = 0;
```

```
#define IRR_SPEED             300  //巡线速度    Patrol speed

float PID_IR_Calc(int8_t actual_value)
{

    float IRTrackTurn = 0;
    int8_t error;
    static int8_t error_last=0;
    static float IRTrack_Integral;

    error=actual_value;

    IRTrack_Integral +=error;

    //位置式pid    Positional pid
    IRTrackTurn=error*IRTrack_Trun_KP
                          +IRTrack_Trun_KI*IRTrack_Integral
                          +(error - error_last)*IRTrack_Trun_KD;
    return IRTrackTurn;
}

//x1-x8 从左往右数   x1-x8 count from left to right
void LineWalking(void)
{
    static int8_t err = 0;
    static u8 x1,x2,x3,x4,x5,x6,x7,x8;

    x1 = IR_Data_number[0];
    x2 = IR_Data_number[1];
    x3 = IR_Data_number[2];
    x4 = IR_Data_number[3];
    x5 = IR_Data_number[4];
    x6 = IR_Data_number[5];
    x7 = IR_Data_number[6];
    x8 = IR_Data_number[7];

//优先判断是否到直角或锐角  Prioritize whether to right angles or acute angles
     if(x1 == 0 && x2 == 0  && x3 == 0&& x4 == 0 && x5 == 0 && x6 == 1  && x7 ==
1 && x8 == 1) // 0000 0111
    {
        err = -15;
        delay_ms(100);
    }
    else if(x1 == 1 && x2 == 1  && x3 == 1&& x4 == 0 && x5 == 0 && x6 == 0  &&
x7 == 0 && x8 == 0) // 1110 0000
    {
        err = 15;
        delay_ms(100);
    }

  else if(x1 == 0 &&  x2 == 0  && x7 == 0 && x8 == 0 ) //俩边都亮，直跑    Both
sides are lit. Run straight.
    {
        err = 0;
        if(trun_flag == 1)
        {
            trun_flag = 0;//走到圈了    Walking in circles.
```

```
        }
    }


    else if(x1 == 1 && x2 == 1  && x3 == 1&& x4 == 0 && x5 == 1 && x6 == 1  &&
x7 == 1 && x8 == 1) // 1110 1111
    {
        err = -1;
    }
    else if(x1 == 1 && x2 == 1  && x3 == 0&& x4 == 0 && x5 == 1 && x6 == 1  &&
x7 == 1 && x8 == 1) // 1100 1111
    {
        err = -2;
    }
    else if(x1 == 1 && x2 == 0  && x3 == 0&& x4 == 1 && x5 == 1 && x6 == 1  &&
x7 == 1 && x8 == 1) // 1001 1111
    {
        err = -8;
    }


    else if(x1 == 0 && x2 == 1  && x3 == 1&& x4 == 1 && x5 == 1 && x6 == 1  &&
x7 == 1 && x8 == 1) // 0111 1111
    {
        err = -10;
    }


    else if(x1 == 1 && x2 == 1  && x3 == 1&& x4 == 1 && x5 == 0 && x6 == 1  &&
x7 == 1 && x8 == 1) // 1111 0111
    {
        err = 1;
    }
    else if(x1 == 1 && x2 == 1  && x3 == 1&& x4 == 1 && x5 == 0 && x6 == 0  &&
x7 == 1 && x8 == 1) // 1111 0011
    {
        err = 2;
    }
    else if(x1 == 1 && x2 == 1  && x3 == 1&& x4 == 1 && x5 == 1 && x6 == 0  &&
x7 == 0 && x8 == 1) // 1111 1001
    {
        err = 8;
    }
        else if(x1 == 1 && x2 == 1  && x3 == 1&& x4 == 1 && x5 == 1 && x6 == 1
 && x7 == 1 && x8 == 0) // 1111 1110
    {
        err = 10;
    }

    else if(x1 == 1 &&x2 == 1 &&x3 == 1 && x4 == 0 && x5 == 0 && x6 == 1 && x7
== 1&& x8 == 1) //直走 go straight
    {
        err = 0;
    }

    //剩下的就保持上一个状态        The rest will stay the same.

    pid_output_IRR = (int)(PID_IR_Calc(err));
```

```
    Motion_Car_Control(IRR_SPEED, 0, pid_output_IRR);

}
```

The `LineWalking` function is used to obtain the status of the 8 infrared sensors, and the deviation value `err` is calculated according to different sensor combinations to determine whether the car needs to turn or go straight. Then, the `PID_IR_Calc` function is used to calculate the PID output according to the error to correct the direction of movement of the car. Finally, the `Motion_Car_Control` function controls the forward and turning of the car according to the calculated PID value and the preset line patrol speed `IRR_SPEED`.

- PID_IR_Calc: Position PID calculation, the calculated result is used to control the movement of the car. If the line patrol effect is not good, you can set KP and KD to 0 first, then slowly increase KP, and finally try to increase the value of KD

- main.c

```
//巡线要想在高难度巡线地图上运行，则需要修改电机的PID才能达到更好的巡线效果
//这个工程是使用四驱310底盘来调的效果，这里使用的电机PID为：P:1.9,I:0.2,D:0.8
//IIC驱动四路电机模块无法更改PID值，因此需要使用电脑串口助手使用串口的命令去修改PID值
//! 其余底盘使用这个电机PID和巡线PID，效果可能没有四驱310的好，需要自行去调节！

//Patrol to run on difficult patrol maps, it is necessary to modify the PID of
the motor in order to achieve better patrol results
//This project is the use of four-wheel drive 310 chassis to adjust the effect of
the motor PID used here: P: 1.9, I: 0.2, D: 0.8
//IIC drive four-way motor module can not change the PID value, so you need to
use the computer serial port assistant to use the serial port command to modify
the PID value
//! The rest of the chassis use this motor PID and patrol PID, the effect may not
be as good as the 4WD 310, you need to adjust yourself!

#include "AllHeader.h"

#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)
{

    //硬件初始化 Hardware Initialization
    BSP_init();

    Set_Motor(MOTOR_TYPE);//设置电机参数  Setting motor parameters

    send_control_data(0,0,1); //设置只接收数值型数据  Set to receive only numeric
data

    while(1)
    {
        LineWalking(); //开始八路巡线  Starting 8-channel patrols.
    }
```

```
}
```

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

Call the `BSP_init()` function to initialize the hardware settings, and use `Set_Motor(MOTOR_TYPE)` to set the motor type and parameters. In the `while(1)` loop, use `Set_Motor(MOTOR_TYPE)` to set the motor type and parameters. In the `while(1)` loop, repeatedly call the `LineWalking()` function to perform line patrol operations.

4. Experimental Phenomenon

Before the experiment, calibrate the 8-channel patrol module. Connect the car, burn the program to STM32, and place the car on the map with white background and black lines. The map adapted for this project is the high-difficulty map sold in our store, which is adapted to the chassis of the four-wheel drive 310. If the patrol effect of other chassis is not good, you need to modify the motor PID and patrol PID in the code yourself. After connecting the power supply, the car adjusts its movement state in real time according to the feedback of the sensor and PID control on the eight-way patrol module to realize the patrol function.