八路巡线

1.开篇说明

请先阅读四路电机驱动板资料中的《电机介绍以及用法》，了解清楚自己现使用的电机参数、接线方式、供电电压。以免造成烧坏主板或者电机的后果。

电机：案例及代码以本店的310电机为例。

2.实验准备

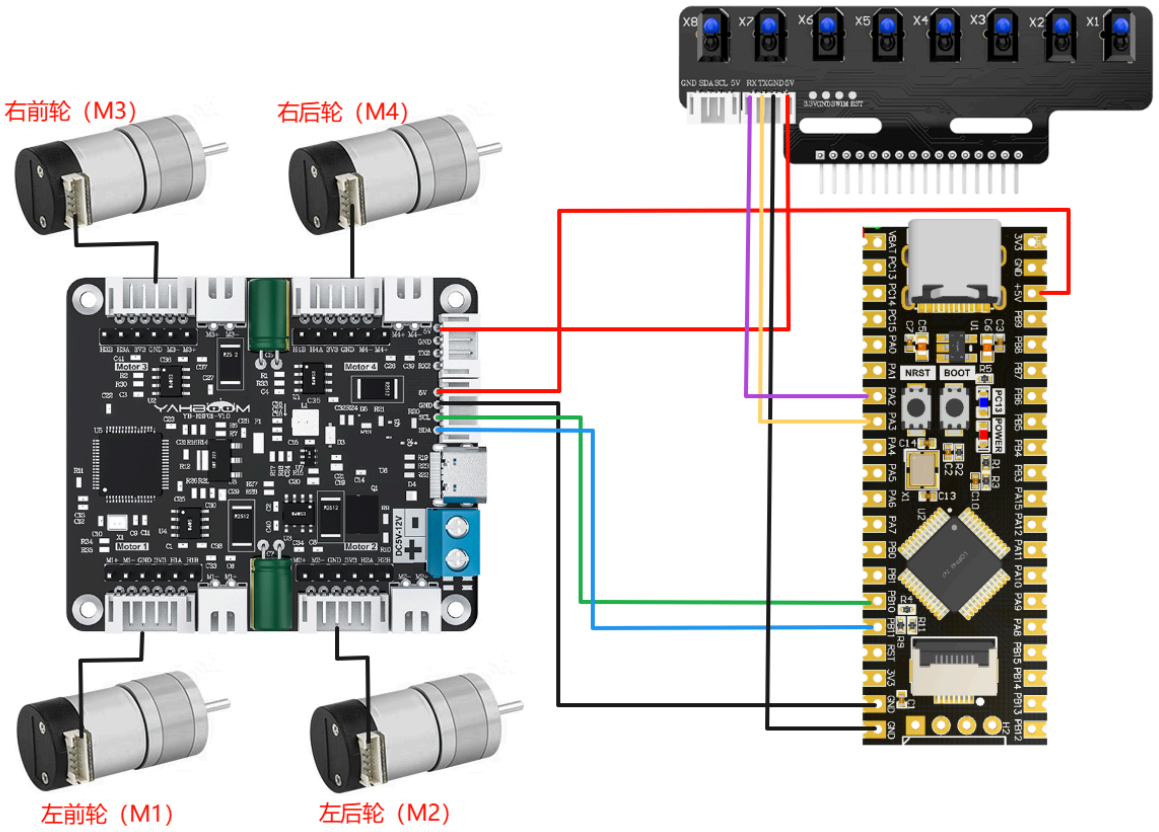
国赛底盘V2四驱版本、4*310电机、7.4V锂电池、八路巡线模块、STM32F103C8T6核心板。

4个电机接口对应小车的关系如下：

M1 -> 左上电机(小车的左前轮)
M2 -> 左下电机(小车的左后轮)
M3 -> 右上电机(小车的右前轮)
M4 -> 右下电机(小车的右后轮)

硬件接线：

整体接线



接线引脚

四路电机驱动板	STM32C8T6
5V	5V
GND	GND
SCL	PB10
SDA	PB11

下面以M1电机为例，其他电机依此类推

电机	四路电机驱动板(Motor)
M2	M1-
VCC	3V3
A	H1A
B	H1B
GND	GND
M1	M1+

八路巡线模块	STM32C8T6
VCC	5V

八路巡线模块	STM32C8T6
GND	GND
TX	PA3
RX	PA2

3.关键码解析

- 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));

}
```

`send_control_data` 函数根据传入的参数修改控制命令，并通过 `USART2` 串口发送，控制是否发送校准命令、模拟值数据或数字值数据。

- 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);

}
```

定义向四路电机驱动板写入配置参数的函数和电机控制函数，用于设置电机类型、死区、磁环线数、减速比和轮子直径等关键参数，并控制四个电机的速度。

- 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,sp
eed_R2_setup);

    control_speed(speed_L1_setup, speed_L2_setup, speed_R1_setup,
speed_R2_setup);

}
```

`Set_Motor` 函数依据电机类型 (MOTOR_TYPE) 进行初始化，包括设置电机类型、减速比、磁环线、轮子直径和电机死区。`Motion_Car_Control` 函数根据输入的前进速度 (`V_x`)、侧向速度 (`V_y`) 和旋转速度 (`V_z`)，计算四个电机的速度值，实现小车的运动控制。函数通过 `Motion_Get_APB` 获取小车轮子轴间距的一半，用于计算旋转速度补偿 (`speed_spin`)，并调整左右电机的速度差异。若速度全为零，则停止电机；否则，计算并限制速度值在 (-1000,1000) 之间，最终调用 `control_speed` 控制电机转速，确保小车按预定方向和速度运动。

- 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);

}
```

通过 `LineWalking` 函数获取 8 个红外传感器的状态，根据不同的传感器组合计算偏差值 `err`，判断小车是否需要转向或直行。然后，利用 `PID_IR_Calc` 函数根据误差计算 PID 输出，用于修正小车的运动方向。最后，`Motion_Car_Control` 函数根据计算出的 PID 值和预设的巡线速度 `IRR_SPEED` 控制小车的前进和转向。

- PID_IR_Calc：位置式PID计算，计算出来的结果用于控制小车运动。如果巡线效果不好，可以将KP和KD先置0，然后慢慢增加KP，最后再尝试加KD的值

- 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 eight-way patrols.
```

```
    }
}
```

`MOTOR_TYPE`：用于设置使用的电机类型，根据自己现使用的电机对照着注释修改对应的数字。

调用 `BSP_init()` 函数初始化硬件设置，并使用 `Set_Motor(MOTOR_TYPE)` 设置电机类型和参数。在 `while(1)` 循环中，并使用 `Set_Motor(MOTOR_TYPE)` 设置电机类型和参数。在 `while(1)` 循环中，反复调用 `LineWalking()` 函数执行巡线操作。

4.实验现象

实验前，先对八路巡线模块进行校准。将小车接好线，给STM32烧录程序后，把小车放在白底黑线的地图上，这个工程适配的地图是本店售卖的高难度地图，适配的是四驱310的底盘。其他的底盘的巡线效果不好的话，需要自己修改代码中的电机PID与巡线PID。打开电源开关，小车根据八路巡线模块上传感器的反馈和PID控制实时调整自己的运动状态，实现巡线功能。