蓝牙遥控

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

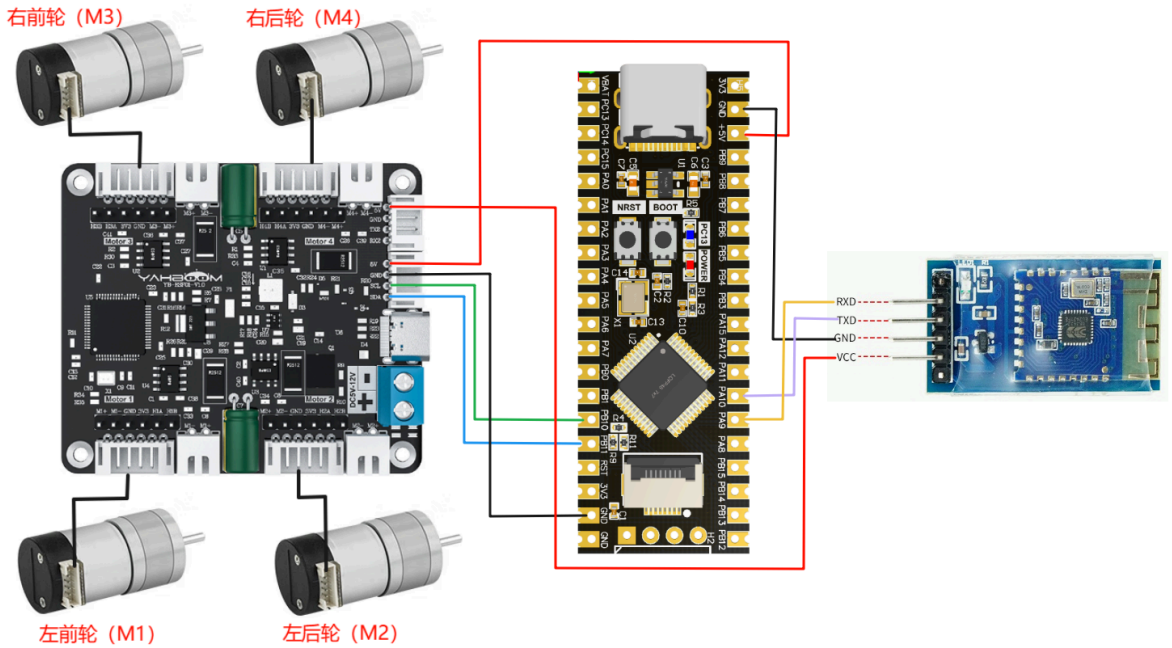
国赛底盘V2四驱版本、4*310电机、7.4V锂电池、蓝牙5.0模块、STM32F103C8T6核心板。

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+

蓝牙模块	STM32C8T6
VCC	3V3
GND	GND
RXD	PA9
TXD	PA10

3.关键码解析

- 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);
}

...

//控制带编码器类型的电机   Control the motor with encoder type
//传入参数:4个电机的pwm PWM of 4 motors
//此函数可以结合实时编码器的数据，来实现control_speed的功能   This function can combine
the data of real-time encoder to realize the function of control_speed
void control_pwm(int16_t m1,int16_t m2 ,int16_t m3,int16_t m4)
{
    static uint8_t pwm[8];

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

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

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

    pwm[6] = (m4>>8)&0xff;
```

```
    pwm[7] = (m4)&0xff;

    i2cWrite(Motor_model_ADDR,PWM_Control_REG,8,pwm);

}
```

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，通过i2c 通信向电机驱动板发送 PWM 信号，从而调节电机的转速和方向。

- Motor.c

```
void Forward(void)
{
    control_pwm(1500,1500,1500,1500);
}
void Backward(void)
{
    control_pwm(-1500,-1500,-1500,-1500);
}
void Turnleft(void)
{
    control_pwm(0,0,2000,2000);
}
void Turnright(void)
{
    control_pwm(2000,2000,0,0);
}
void Stop(void)
{
    control_pwm(0,0,0,0);
}
void SpinLeft(void)
{
    control_pwm(-1500,-1500,1500,1500);
}
void SpinRight(void)
{
    control_pwm(1500,1500,-1500,-1500);
}

void Data_Analyse(void)//解析串口中断串口接收的数据
{
        if(rxd_flag == 1)
        {
            switch(rxd_buf[0])
            {
                case '1':
                    printf("Forward!\n");
                    Car_state = 1;
                    break;
                case '2':
                    printf("Backward!\n");
                    Car_state = 2;
                    break;
```

```
                case '3':
                    printf("Left!\n");
                    Car_state = 3;
                    break;
                case '4':
                    printf("Right!\n");
                    Car_state = 4;
                    break;
                case '0':
                    printf("Stop!\n");
                    Car_state = 0;
                    break;
            }
            switch(rxd_buf[2])
            {
                case '1':
                    printf("SpinLeft!\n");
                    Car_state = 5;
                    break;
                case '2':
                    printf("SpinRight!\n");
                    Car_state = 6;
                    break;
            }
            rxd_flag = 0;
        }
}

void Car_Function(unsigned int Car_state)//控制小车不同状态
{
    switch(Car_state)
    {
            case 0:
                    printf("Stop\n");
                    Stop();
                    break;
            case 1:
                    printf("Forward\n");
                    Forward();
                    break;
            case 2:
                    printf("Backward\n");
                    Backward();
                    break;
            case 3:
                    printf("Turnleft\n");
                    Turnleft();
                    break;
            case 4:
                    printf("Turnright\n");
                    Turnright();
                    break;
            case 5:
                    printf("SpinLeft\n");
                    SpinLeft();
                    break;
```

```
            case 6:
                    printf("SpinRight\n");
                    SpinRight();
                    break;
    }
}
```

定义了几个运动控制函数，通过 `control_pwm()` 控制四个电机，实现小车的前进、后退、转向、停止和原地旋转；`Data_Analyse()` 解析串口指令并更新 `Car_state`，`Car_Function()` 根据 `Car_state` 调用相应的运动函数，使小车按照接收到的指令执行对应动作。

- main.c

```
#include "stm32f10x.h"
#include "SysTick.h"
#include "Uart.h"
#include "bsp_motor_iic.h"
#include "Motor.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)
{
    delay_init();       //配置定时器Configure the timer
    Uart1_Configuration();          //配置UART1串口Configure UART1 serial port
    Uart1_NVIC_Configuration(); //配置UART1串口中断Configure UART1 serial port interrupt

    IIC_Motor_Init();

    #if MOTOR_TYPE == 1
    ...

    #elif 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(1300);//配置电机死区,实验得出  Configure the motor dead zone, and the experiment shows
    delay_ms(100);

    ...

    #endif
```

```
    while(1)
    {
            Data_Analyse();//解析串口中断接收的数据    Analyze data received by the
serial port interrupt
            Car_Function(Car_state);//控制小车不同状态  Control different states of
the car
    }
}
```

`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。

初始化定时器、UART 串口和 IIC 电机驱动模块，并根据 `MOTOR_TYPE` 使用 `Set_motor_type()` 等函数设置电机类型和参数。在主循环中，调用 `Data_Analyse()` 不断解析串口接收的数据，并调用 `Car_Function()` 根据解析结果控制小车的运动状态，如前进、后退、转向或停止。

4.实验现象

将小车接好线，给STM32烧录程序后，把小车放在地上，插上电源。打开app，选择4WD车型，连接蓝牙后就可以通过app控制小车前进后退、左转右转、左旋右旋。

附录：

APP下载：https://www.yahboom.com/download_app

YahboomRobot APP具体操作