4-Channel Tracking

1. Opening instructions

Please read the "Motor Introduction and Usage" in the four-way motor driver board information first to understand the motor parameters, wiring method, 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 motor, 7.4V lithium battery, 4-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)

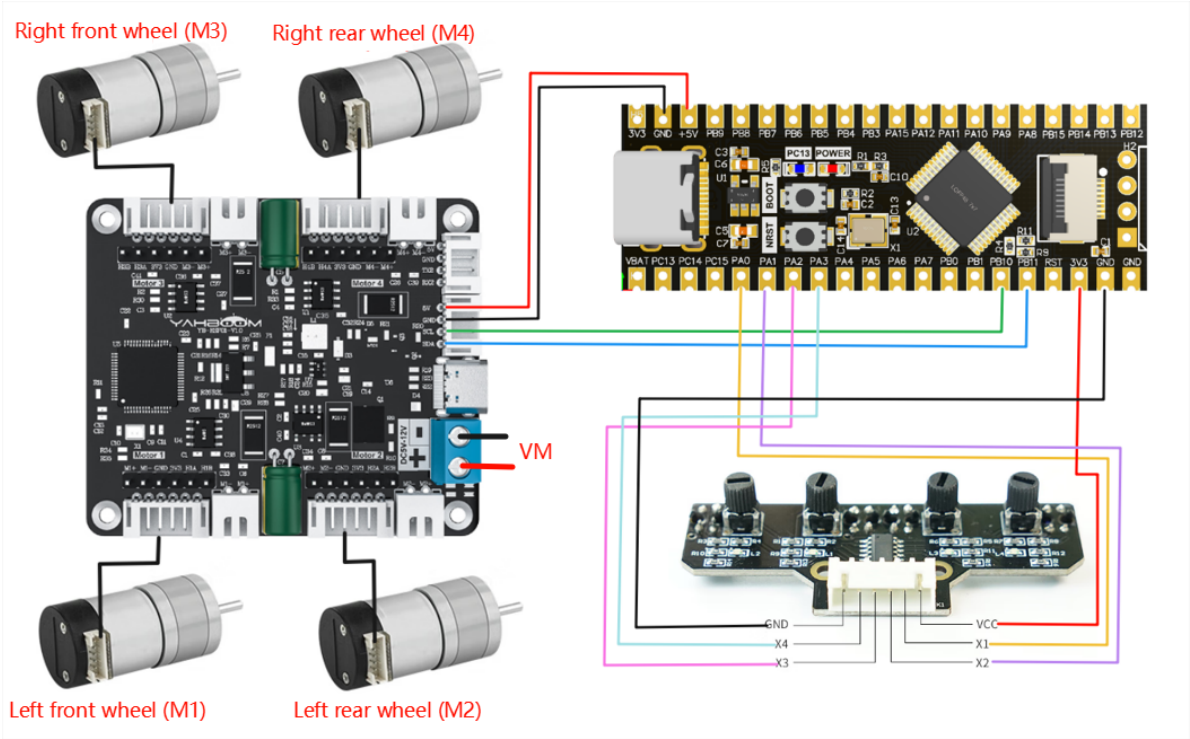
M2 -> lower left motor (left rear wheel of the car)

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+

4-channel patrol module	STM32C8T6
VCC	5V
GND	GND
X1	PA0
X2	PA1
X3	PA2
X4	PA3

3. Key code analysis

- 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
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    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.

- Four_linewalking.c

```
#include "Four_linewalking.h"

#define IRTrack_Trun_KP (450)
#define IRTrack_Trun_KI (0)
#define IRTrack_Trun_KD (0)

int pid_output_IRR = 0;
float err = 0;

int IRR_SPEED = 300;//初始直线速度    Initial linear velocity

float APP_IR_PID_Calc(float actual_value)
{

    float IRTrackTurn = 0;
    int8_t error;
    static int8_t error_last=0;
    static float IRTrack_Integral;//积分  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;
}

void Four_LineWalking_GPIO_Init(void)
{
    /*定义一个GPIO_InitTypeDef类型的结构体*/  /*Define a GPIO_InitTypeDef type
structure*/
    GPIO_InitTypeDef GPIO_InitStructure;
    /*开启外设时钟*/  /*Turn on peripheral clock*/
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    /*选择要控制的引脚*/    /*Select the pin to control*/

    GPIO_InitStructure.GPIO_Pin = LineWalk_L1_PIN | LineWalk_L2_PIN |
LineWalk_R1_PIN | LineWalk_R2_PIN;
    /*设置引脚模式为上拉输入*/ /*Set pin mode to pull-up input*/
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    /*设置引脚速率为50MHz */   /*Set the pin rate to 50MHz */
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    /*调用库函数，初始化PORT*/   /*Call library function to initialize PORT*/
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```

```
//获取X1X2X3X4的引脚电平   Get the pin levels of X1X2X3X4
void Four_GetLineWalking(int *LineL1, int *LineL2, int *LineR1, int *LineR2)
{
    *LineL1 = GPIO_ReadInputDataBit(GPIOA, LineWalk_L1_PIN);
    *LineL2 = GPIO_ReadInputDataBit(GPIOA, LineWalk_L2_PIN);
    *LineR1 = GPIO_ReadInputDataBit(GPIOA, LineWalk_R1_PIN);
    *LineR2 = GPIO_ReadInputDataBit(GPIOA, LineWalk_R2_PIN);
}

void Four_LineWalking(void)
{
    int LineL1 = 0, LineL2 = 0, LineR1 = 0, LineR2 = 0;
    Four_GetLineWalking(&LineL1, &LineL2, &LineR1, &LineR2);//获取黑线检测状态  Get
black line detection status
    //debug
//    printf("L1:%d L2:%d R1:%d R2:%d\r\n",LineL1, LineL2, LineR1, LineR2);

    // 0 0 X 0
    // 1 0 X 0
    // 0 1 X 0
    //处理右锐角和右直角的转动
    //Processing the right acute angle and the right right angle rotation
    if( (LineL1 == LOW || LineL2 == LOW) && LineR2 == LOW)
    {
        err=13;
        delay_ms(80);
    }
   // 0 X 0 0
   // 0 X 0 1
   // 0 X 1 0
   //处理左锐角和左直角的转动
    //Handling left acute angle and left right angle rotation
    else if ( LineL1 == LOW && (LineR1 == LOW || LineR2 == LOW))
    {
        err=-13;
        delay_ms(80);
    }
    // 0 X X X
   //最左边检测到
    //Most left detected
    else if( LineL1 == LOW )
    {
        err=-9;
        delay_ms(10);
    }
    // X X X 0
   //最右边检测到
    //Most right detected
    else if ( LineR2 == LOW)
    {
        err=9;
//      Contrl_Speed(500,500,-500,-500);
        delay_ms(10);
    }
    // X 0 1 X
   //处理左小弯
    //Processing of the left hand chicane
```

```
    else if (LineL2 == LOW && LineR1 == HIGH) //中间黑线上的传感器微调车左转  Sensor
on the black line in the center fine tunes the car to turn left
    {
        err=-1;
    }
    // X 1 0 X
   //处理右小弯
    //Processing of the right-hand chicane
    else if (LineL2 == HIGH && LineR1 == LOW) //中间黑线上的传感器微调车右转  The
sensor on the center black line fine tunes the car to turn right
    {
        err=1;
    }
    // X 0 0 X
   //处理直线
    //Processing straight lines
    else if(LineL2 == LOW && LineR1 == LOW) // 都是黑色，加速前进   It's all black,
so speed up.
    {
        err=0;
    }
    // 0 0 0 0
    else if(LineL1 == LOW && LineL2 == LOW && LineR1 == LOW && LineR2 == LOW) //
都是黑色，加速前进 It's all black, so speed up.
    {
        err = 0;
    }

    //当为1 1 1 1时小车保持上一个小车运行状态
    //When it is 1 1 1 1 the trolley keeps the previous trolley in operation

    pid_output_IRR = (int)(APP_IR_PID_Calc(err));

    Motion_Car_Control(IRR_SPEED, 0, pid_output_IRR);

}
```

`Four_LineWalking_GPIO_Init` initializes the GPIO pins for reading the signals of the four black line sensors.

`Four_GetLineWalking` gets the status of the four sensors (L1, L2, R1, R2).

`APP_IR_PID_Calc` calculates the PID control output and adjusts the movement of the car according to the sensor feedback (position error). 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

`Four_LineWalking` function determines whether the car needs to turn, go straight or turn around according to the status of the sensor. When the left or right sensor detects a black line, adjust the steering. When neither sensor detects a black line, keep going straight. Use PID control to adjust the speed and steering according to the sensor error.

- 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!

#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)
{
    bsp_init();

    TIM3_Init();
    //四路巡线模块初始化 Initialization of the 4-Way Patrol Module
    Four_LineWalking_GPIO_Init();
    //电机模块iic通信初始化  Motor module iic communication initialization
    IIC_Motor_Init();
    Set_Motor(MOTOR_TYPE);//设置电机参数  Setting motor parameters

    while(1)
    {
        Four_LineWalking();//四路巡线，启动！    Four-way line patrol, start!
    }

}
```

`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.

Initialize the underlying hardware through `bsp_init()`, initialize timer 3 through `TIM3_Init()`, configure the four-way line patrol sensor pins through `Four_LineWalking_GPIO_Init()`, initialize the motor I2C communication through `IIC_Motor_Init()`, and use `Set_Motor(MOTOR_TYPE)` to set the motor type and parameters. In the `while(1)` loop, `Four_LineWalking()` reads the line patrol sensor data, and after calculating the PID error with `APP_IR_PID_Calc()`, calls `Motion_Car_Control()` to adjust the motor speed and realize the automatic line patrol driving of the car.

4. Experimental phenomenon

After connecting the car and burning the program to the STM32, place the car on the map with white background and black lines, and unplug the serial port line that communicates with the four-way motor module. Adjust the four-way line patrol module and adjust the module knob so that the indicator light of the module is on when encountering a black line, and the indicator light is off when it is not a black line. After the adjustment is completed, disconnect the power supply and plug the serial port cable back in. Put the car in the middle, let the two middle probes of the four-way line patrol module be on the black line, then plug in the power supply, and the car will patrol the black line. The map adapted for this project is the high-difficulty map sold in this store,

which is adapted to the chassis of the four-wheel drive 310. If the line patrol effect of other chassis is not good, you need to modify the motor PID and line patrol PID in the code yourself.
Note: **This experiment requires adjusting the potentiometer of the 4-way infrared tracking module to achieve the best sensitivity of line patrol.**