```
#include "stm32f10x.h"
#include "motor.h"
#include "interface.h"
#include "string.h"

//全局变量定义
uint16_t speed_count=0;//占空比计数器 50 次一周期
int8_t left_speed_duty=MIN_SPEED_DUTY;
int8_t right_speed_duty=MIN_SPEED_DUTY;

uint8_t tick_5ms = 0;//5ms 计数器，作为主函数的基本周期
uint8_t tick_1ms = 0;//1ms 计数器，作为电机的基本计数器

int8_t ctrl_comm = COMM_STOP;//控制指令
int8_t ctrl_comm_last = COMM_STOP;//上一次的指令
uint8_t continue_time=0;

char strSrc[20];

void RedRayInit(void);
void GPIO_Conf(void);
void RCC_Conf(void);
void TIM2_Init(void);
void MotorInit(void);
void SEARCHPath(void);

int main(void)
{
    uint16_t i,j;

    RCC_Conf();
    GPIO_Conf();
    RedRayInit();
    TIM2_Init();
    MotorInit();
    for(i=0;i<10000;i++)
        for(j=0;j<1000;j++);
    strSrc[0] = '\0';
    while (1)
    {
        if(tick_5ms >= 5)
        {
            tick_5ms = 0;
            //do something
```

```
            SEARCHPath();   //SearchRun();
            if(ctrl_comm_last != ctrl_comm)//指令发生变化
            {
                ctrl_comm_last = ctrl_comm;
                switch(ctrl_comm)
                {
                    case COMM_UP:    CarGo();break;
                    case COMM_UPL:   CarGoL();break;
                    case COMM_UPR:   CarGoR();break;
                    default : break;
                }
            }
        }
    }
}

void RCC_Conf(void)
{
    ErrorStatus HSEStartUPStatus;
    RCC_DeInit();
    RCC_HSEConfig(RCC_HSE_ON);
    HSEStartUPStatus=RCC_WaitForHSEStartUp();
    if(HSEStartUPStatus == SUCCESS)
    {
        RCC_HCLKConfig(RCC_SYSCLK_Div1);
        RCC_PCLK2Config(RCC_HCLK_Div1);
        RCC_PCLK1Config(RCC_HCLK_Div2);
        RCC_PLLConfig(RCC_PLLSource_HSE_Div1,RCC_PLLMul_9);
        RCC_PLLCmd(ENABLE);
        while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
        while(RCC_GetSYSCLKSource()!=0x08);
    }

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC,ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
}

void GPIO_Conf(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;

    GPIO_InitStructure.GPIO_Pin=LED_PIN;
```

```
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_2MHz;
    GPIO_Init(GPIOC,&GPIO_InitStructure);
}

//红外光电对管初始化--寻迹使用
void RedRayInit(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
    //L2
    GPIO_InitStructure.GPIO_Pin = SEARCH_L2_PIN;//配置使能 GPIO 管脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;//配置 GPIO 模式,输入上拉
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;//配置 GPIO 端口速度
    GPIO_Init(SEARCH_L2_GPIO , &GPIO_InitStructure);
    //L1
    GPIO_InitStructure.GPIO_Pin = SEARCH_L1_PIN;//配置使能 GPIO 管脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;//配置 GPIO 模式,输入上拉
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;//配置 GPIO 端口速度
    GPIO_Init(SEARCH_L1_GPIO , &GPIO_InitStructure);
    //M
    GPIO_InitStructure.GPIO_Pin = SEARCH_M_PIN;//配置使能 GPIO 管脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;//配置 GPIO 模式,输入上拉
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;//配置 GPIO 端口速度
    GPIO_Init(SEARCH_M_GPIO , &GPIO_InitStructure);
    //R1
    GPIO_InitStructure.GPIO_Pin = SEARCH_R1_PIN;//配置使能 GPIO 管脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;//配置 GPIO 模式,输入上拉
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;//配置 GPIO 端口速度
    GPIO_Init(SEARCH_R1_GPIO , &GPIO_InitStructure);
    //R2
    GPIO_InitStructure.GPIO_Pin = SEARCH_R2_PIN;//配置使能 GPIO 管脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;//配置 GPIO 模式,输入上拉
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;//配置 GPIO 端口速度
    GPIO_Init(SEARCH_R2_GPIO , &GPIO_InitStructure);
}

//打开 103 的启动文件 starup_stm32f10x_md.s,这里面只有 TIM1-TIM4 的定时器中断事件,
TIM6 是没有中断响应这个功能的
void TIM2_Init(void)
{
    TIM_TimeBaseInitTypeDef  TIM2_TimeBaseStructure;
    NVIC_InitTypeDef NVIC_InitStructure;

    NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
```

```
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_Init(&NVIC_InitStructure);

    TIM_TimeBaseStructInit(&TIM2_TimeBaseStructure);
    TIM2_TimeBaseStructure.TIM_Prescaler = 72 - 1;  //分频之后的时钟频率为 1MHz
    TIM2_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM2_TimeBaseStructure.TIM_Period = 100 - 1;       //定时 0.1ms 频率 10kHz
    TIM2_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseInit(TIM2, &TIM2_TimeBaseStructure);

    TIM_UpdateRequestConfig(TIM2, TIM_UpdateSource_Regular);    //将 TIM2 的更新
事件设为其触发输出源

    TIM_Cmd(TIM2, ENABLE);
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
}

void TIM2_IRQHandler(void)
{
    if(TIM_GetITStatus(TIM2, TIM_IT_Update)== SET)  //!=RESET
    {
        TIM_ClearITPendingBit(TIM2, TIM_FLAG_Update);
        tick_1ms++;
        if(tick_1ms > 10)
        {   //1ms 定时到
            tick_1ms = 0;
            speed_count++;
            tick_5ms++;
            if(speed_count > 50)
            {
                speed_count = 0;
            }
            CarMove();
        }
    }
}
//五寻迹传感器
void SEARCHPath(void)
{
    uint8_t nowL2,nowL1,nowR1,nowR2;    //now-当前测试状态

    nowL2 = SEARCH_L2_IO;
```

```
    nowL1 = SEARCH_L1_IO;
    nowR1 = SEARCH_R1_IO;
    nowR2 = SEARCH_R2_IO;

    if(nowL1 == 1 || nowL2 ==1) //直行,偏左,需向右偏小微调
        ctrl_comm = COMM_UPL;
    else if(nowR1 == 1 || nowR2 == 1)   //直行,偏右,需向左偏微调
        ctrl_comm = COMM_UPR;
    else    //直行路
        ctrl_comm = COMM_UP;
}
```