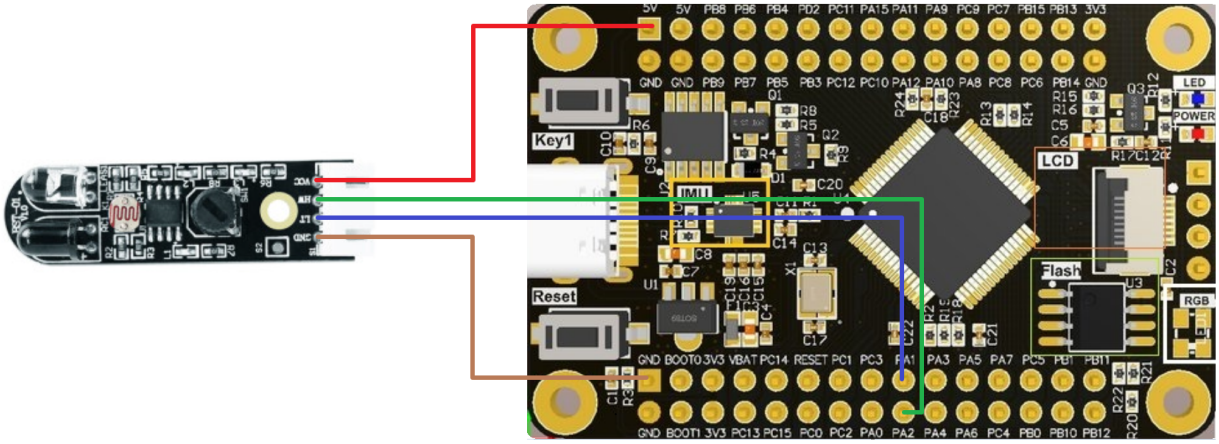
Infrared obstacle avoidance light tracing module: GPIO input

Hardware wiring



LED	Infrared obstacle avoidance light seeking module	STM32F103RCT6
	VCC	5V/3.3V
L2	LT	PA1
L1	HW	PA2
	GND	GND

Brief principle

Infrared obstacle avoidance:

Obstacle detected: light on The infrared obstacle avoidance light finding module corresponds to the output low level of the HW interface

No obstacle detected: light off The infrared obstacle avoidance and light finding module corresponds to the output high level of the HW interface

Photosensitive sensors:

Weak ambient light: the light is on The infrared obstacle avoidance light finding module corresponds to the LT interface output low level

Ambient light intensity: light off The infrared obstacle avoidance and light finding module corresponds to the LT interface output high level

Main code

main.c

```
#include "stm32f10x.h"
#include "UART.h"
#include "SysTick.h"
#include "Infrared_Seek_Light.h"

int main(void)
{

    SysTick_Init();//滴答定时器初始化
    UART1_Init();//UART1初始化
    Infrared_Seek_Light_Init();//红外避障寻光模块引脚初始化
    printf("Infrared_Seek_Light!\n");

    while(1)
    {
        Infrared_Seek_Light_State();//红外避障寻光模块 检测状态
        Delay_us(1000000);
    }
}
```

SysTick.c

```
#include "SysTick.h"

unsigned int Delay_Num;

void SysTick_Init(void)//滴答定时器初始化
{
    while(SysTick_Config(72));//设置重装载值 72 对应延时函数为微秒级
    //若将重装载值设置为72000 对应延时函数为毫秒级
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
}

void Delay_us(unsigned int NCount)//微秒级延时函数
{
    Delay_Num = NCount;
    SysTick->CTRL |= (1 << 0);//开启定时器
    while(Delay_Num);
    SysTick->CTRL &= ~(1 << 0);//定时器初始化后关闭，使用再开启
}

void SysTick_Handler(void)
{
    if(Delay_Num != 0)
    {
        Delay_Num--;
    }
```

```
}
```

SysTick.h

```
#ifndef __SYSTICK_H__
#define __SYSTICK_H__

#include "stm32f10x.h"

void SysTick_Init(void);//滴答定时器初始化
void Delay_us(unsigned int NCount);//微秒级延时函数

#endif
```

UART.c

```
#include "UART.h"

void UART1_Init(void)//UART1初始化
{
    USART_InitTypeDef USART_InitStructure;
    GPIO_InitTypeDef GPIO_InitStructure;

    /* Enable GPIO clock */
    /* 使能GPIOA AFIO时钟 TXD(PA9) RXD(PA10) */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO, ENABLE);

    /* Enable USART1 Clock */
    /* 使能串口1 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 , ENABLE);

    /* Configure USART1 Rx as input floating */
    /* 配置RXD引脚 PA10 浮空输入模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure USART1 Tx as alternate function push-pull */
    /* 配置TXD引脚 PA9 复用推挽输出模式 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* USART1 configured as follow:
    - BaudRate = 115200 baud
    - Word Length = 8 Bits
    - One Stop Bit
    - No parity
    - Hardware flow control disabled (RTS and CTS signals)
    - Receive and transmit enabled
```

```
    */
    USART_InitStructure.USART_BaudRate = 115200;//波特率设置
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;//8位数据位
    USART_InitStructure.USART_StopBits = USART_StopBits_1;//1位停止位
    USART_InitStructure.USART_Parity = USART_Parity_No;//无奇偶校验位
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;//无需硬件流控
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;//全双工 即发送也接
受
    USART_Init(USART1, &USART_InitStructure);

    /* Enable USART1 Receive and Transmit interrupts */
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
    USART_ITConfig(USART1, USART_IT_TXE, ENABLE);

    /* Enable the USART1 */
    /* 使能USART1 */
    USART_Cmd(USART1, ENABLE);
}

void NVIC_UART1_Init(void)//UART1 NVIC配置
{
    NVIC_InitTypeDef NVIC_InitStructure;

    /* Configure the NVIC Preemption Priority Bits */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);

    /* Enable the USART1 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}


void USART_SendString(USART_TypeDef* USARTx, char *pt)//给指定串口发送字符串
{
    while(*pt)
    {
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
        USART_SendData(USARTx,*pt);
        while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
        pt++;
    }
}

int fputc(int c, FILE *pt)//printf重定向
{
    USART_TypeDef* USARTx = USART1;
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TXE) == RESET);
    USART_SendData(USARTx, c);
    while(USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
```

```
    return 0;
}

void USART1_IRQHandler(void)
{
    unsigned char ch;
    while(USART_GetFlagStatus(USART1, USART_FLAG_RXNE) == SET)//接收到数据
    {
        ch = USART_ReceiveData(USART1);
        printf("%c\n",ch);
    }
}
```

UART.h

```
#ifndef __UART_H__
#define __UART_H__

#include "stm32f10x.h"
#include "stdio.h"

void UART1_Init(void);//UART1初始化
void USART_SendString(USART_TypeDef* USARTx, char *pt);//给指定串口发送字符串
int fputc(int c, FILE *pt);//printf重定向
void NVIC_UART1_Init(void);//UART1 NVIC配置

#endif
```

Infrared_Seek_Light.c

```
#include "Infrared_Seek_Light.h"

void Infrared_Seek_Light_Init(void)//红外避障寻光模块引脚初始化
{
    GPIO_InitTypeDef GPIO_InitStructure;

    /* GPIOA Periph clock enable */
    /* 使能 DPIOA 时钟 */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

    /* Configure PA1 PA2 in GPIO_Mode_IPU mode */
    //红外避障寻光模块对应 LT寻光输出:PA1 HW避障输出:PA2
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}

/*
说明:
            本例程只提供简单的单个模块测试功能，可以通过该例程调节循迹模块的灵敏度
            如果想实现寻光功能，则需要搭配两个红外避障寻光模块
```

```
            若想驱动小车：可结合自己的电机驱动代码文件
            对两个红外避障寻光模块的各个检测状态进行判断，根据判断加入电机驱动控制
*/

void Infrared_Seek_Light_State(void)//红外避障寻光模块 检测状态
{
    if(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_1) == Bit_RESET)
    {
        printf("Seek_Light module detected an obstacle!\n");
    }
    if(GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_2) == Bit_RESET)
    {
        printf("Infrared_Obstacle module detected an obstacle!\n");
    }
}
```

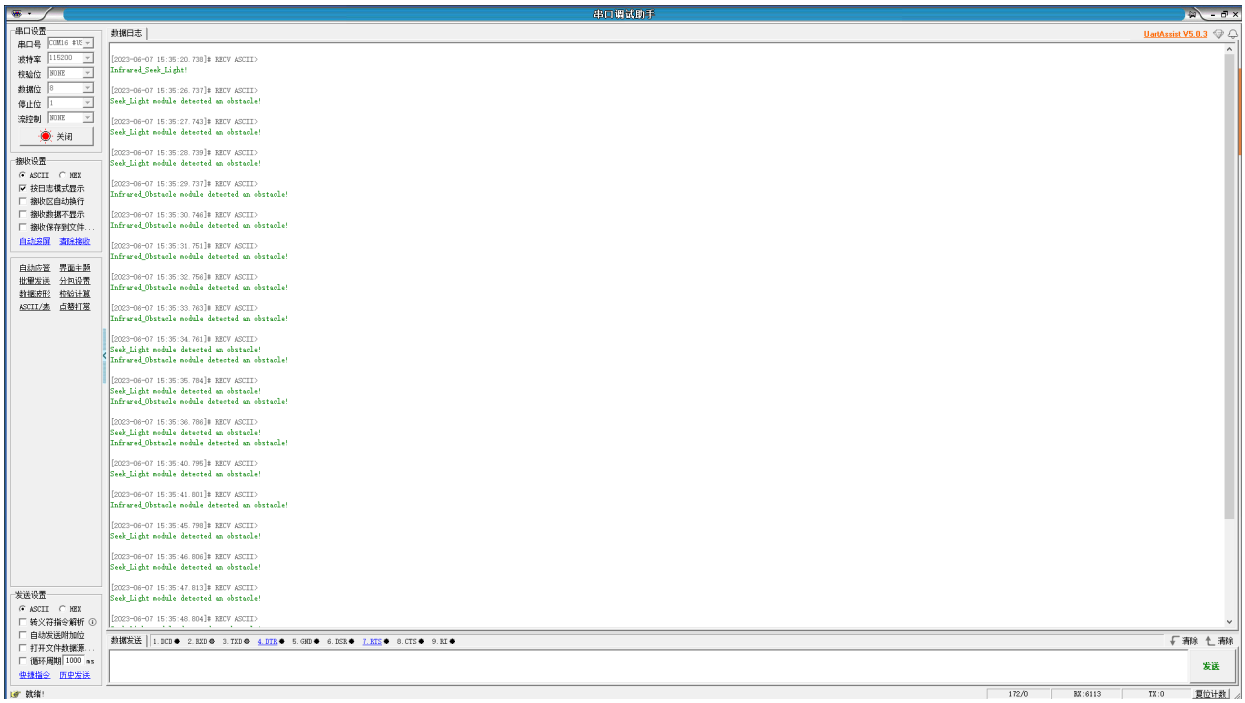
Infrared_Seek_Light.h

```
#ifndef __INFRARED_SEEK_LIGHT_H__
#define __INFRARED_SEEK_LIGHT_H__

#include "stm32f10x.h"
#include "UART.h"

void Infrared_Seek_Light_Init(void);//红外避障寻光模块引脚初始化
void Infrared_Seek_Light_State(void);//红外避障寻光模块 检测状态

#endif
```

Phenomenon

After downloading the program, press the Reset key once, and the downloaded program will run.

When you block the IR/photosensitive sensor corresponding to the infrared obstacle avoidance and light seeking module, the serial port will print the information that the corresponding sensor detects the obstacle.



Note:

This routine only provides a simple single module test function, through which the sensitivity of the tracking module can be adjusted; If you want to realize the light finding function, you need to be equipped with two infrared obstacle avoidance light finding modules.

If you want to drive the trolley: you can combine your own motor drive code file to judge the detection status of the two infrared obstacle avoidance and light finding modules, and add the motor drive control according to the judgment