



## HiSpark-智能小车套件 循迹模块编程指南

文档版本 00B01

发布日期 2020/8/3

**版权所有 © 上海海思技术有限公司。保留一切权利。**

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## **商标声明**



**HISILICON**、海思和其他海思商标均为上海海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## **注意**

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

## **上海海思技术有限公司**

地址：上海市青浦区金泽镇（西岑）水秀路 318 号 101 室 邮编：201718

网址：<http://www.hisilicon.com>



# 前言

## 概述

本文档主要介绍基于海思 WiFi 芯片 Hi3861 开发的 HiSpark-WiFi-IoT 套件演示指导书。

## 产品版本

与本文档相对应的主芯片版本如下。

产品名称	产品版本
Hi3861	V100R001C00SPC021

## 读者对象

本文档（本指南）主要适用于以下工程师：

- 软件开发工程师
- 硬件开发工程师

## 修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2020-08-3	00B01	第一次临时版本发布。





# 目 录

前 言.....	i
1 智能小车套件循迹模块功能实现 .....	4
1.1 循迹模块硬件准备 .....	4
1.2 循迹模块原理介绍 .....	5
1.3 循迹模块功能软件实现.....	5



# 1 智能小车套件循迹模块功能实现

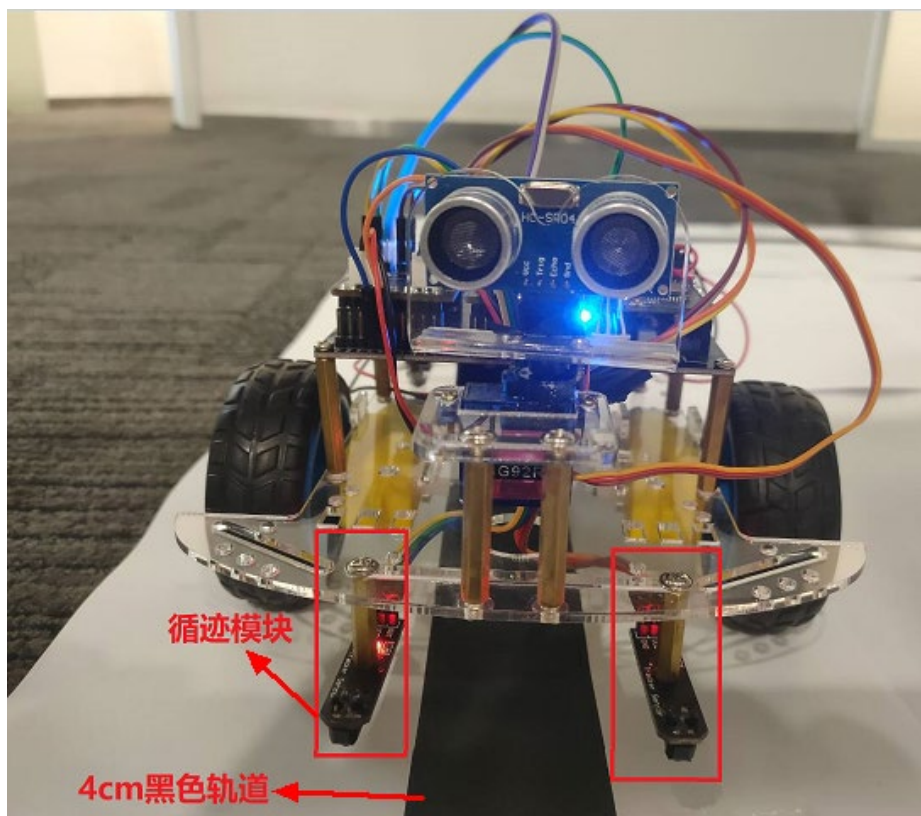
## 1.1 循迹模块硬件准备

图 1.1-1 循迹模块



图 1.1-2 循迹模块在小车上的安装

(两个循迹模块之间的间距需大于轨道宽度，一般 5~6 厘米即可)





## 1.2 循迹模块原理介绍

循迹所采用的模块为 TCRT5000 模块，传感器的红外发射二极管不断发射红外线，当发射出的红外线没有被反射回来或被反射回来但强度不够大时，此时模块输出端为低电平，指示二极管一直处于熄灭状态；当被检测物体出现在检测范围内时，红外线被反射回来且强度足够大，光敏三极管饱和，此时模块的输出端为高电平，指示二极管被点亮。

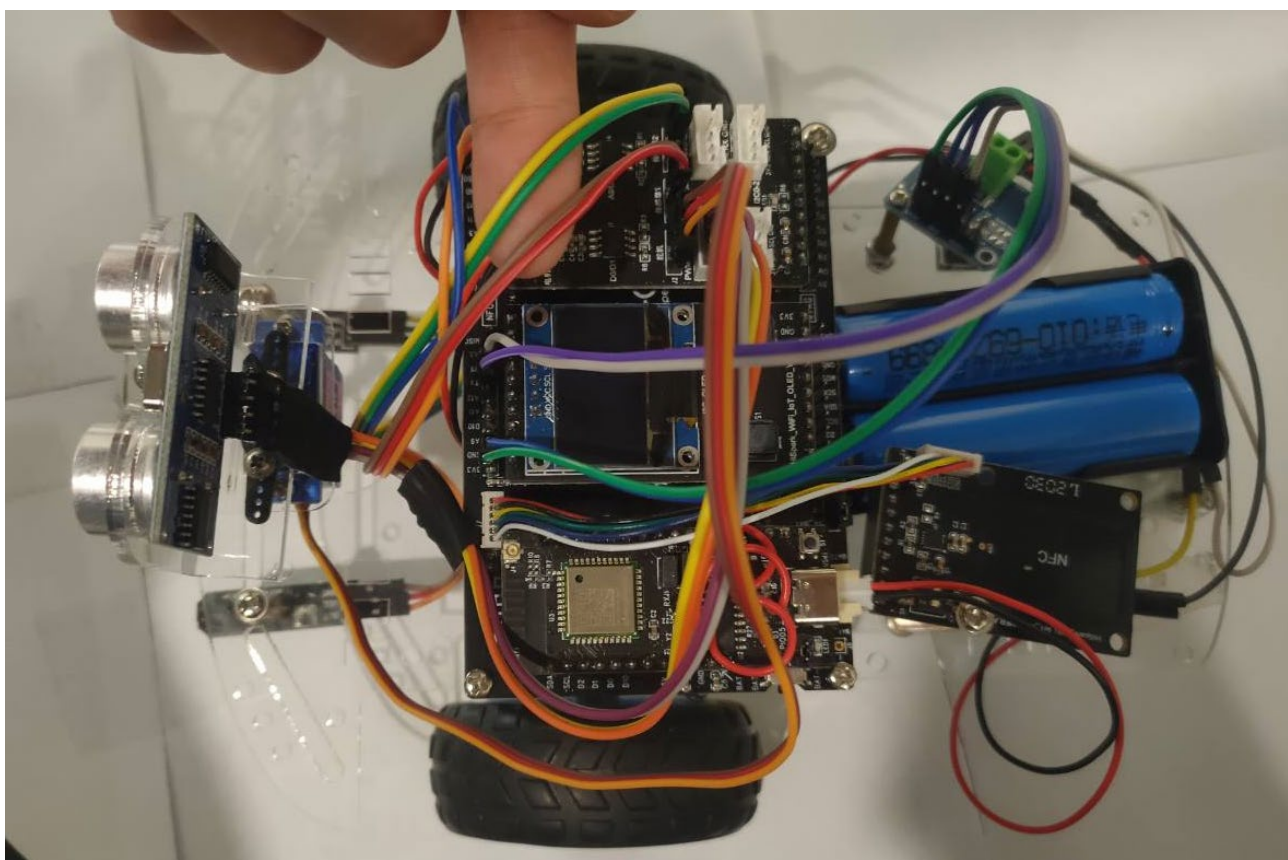
小车共配置两个循迹模块：

Out1 ----- GPIO\_11

Out2 ----- GPIO\_12

循迹模块在智能小车上的连线如下所示：

图 1.2-1 循迹模块接线图



## 1.3 循迹模块功能软件实现

(1) 首先进行 IO 引脚的初始化工作，主要为以下函数：

```
hi_u32 hi_io_set_func(hi_io_name id, hi_u8 val);
```

```
hi_u32 hi_gpio_set_dir(hi_gpio_idx id, hi_gpio_dir dir);
```



```
hi_u32 hi_io_set_pull(hi_io_name id, hi_io_pull val);
```

以下为各函数的详细介绍：

- `hi_u32 hi_io_set_func(hi_io_name id, hi_u8 val);`

**hi\_io\_name id:** gpio 硬件管脚编号，对应的是硬件管脚上的 GPIO pin 脚，枚举类型如下：

```
typedef enum {  
    HI_IO_NAME_GPIO_0,      /**< GPIO0 */  
    HI_IO_NAME_GPIO_1,      /**< GPIO1 */  
    HI_IO_NAME_GPIO_2,      /**< GPIO2 */  
    HI_IO_NAME_GPIO_3,      /**< GPIO3 */  
    HI_IO_NAME_GPIO_4,      /**< GPIO4 */  
    HI_IO_NAME_GPIO_5,      /**< GPIO5 */  
    HI_IO_NAME_GPIO_6,      /**< GPIO6 */  
    HI_IO_NAME_GPIO_7,      /**< GPIO7 */  
    HI_IO_NAME_GPIO_8,      /**< GPIO8 */  
    HI_IO_NAME_GPIO_9,      /**< GPIO9 */  
    HI_IO_NAME_GPIO_10,     /**< GPIO10 */  
    HI_IO_NAME_GPIO_11,     /**< GPIO11 */  
    HI_IO_NAME_GPIO_12,     /**< GPIO12 */  
    HI_IO_NAME_GPIO_13,     /**< GPIO13 */  
    HI_IO_NAME_GPIO_14,     /**< GPIO14 */  
    HI_IO_NAME_SFC_CSN,     /**< SFC_CSN */  
    HI_IO_NAME_SFC_IO1,     /**< SFC_IO1 */  
    HI_IO_NAME_SFC_IO2,     /**< SFC_IO2 */  
    HI_IO_NAME_SFC_IO0,     /**< SFC_IO0 */  
    HI_IO_NAME_SFC_CLK,     /**< SFC_CLK */  
    HI_IO_NAME_SFC_IO3,     /**< SFC_IO3 */  
    HI_IO_NAME_MAX,  
} hi_io_name;
```

**hi\_u8 val:** 对应 GPIO 引脚功能，如果该引脚复用，可以配置为其他的复用功能，如：UART/PWM/I2C/SPI/SDIO 等功能；

- `hi_u32 hi_gpio_set_dir(hi_gpio_idx id, hi_gpio_dir dir);`

**hi\_gpio\_idx id:** 硬件管脚编号，对应的是硬件管脚上的 GPIO pin 脚，枚举类型如下：,





```
typedef enum {  
    HI_GPIO_IDX_0, /**< GPIO0*/  
    HI_GPIO_IDX_1, /**< GPIO1*/  
    HI_GPIO_IDX_2, /**< GPIO2*/  
    HI_GPIO_IDX_3, /**< GPIO3*/  
    HI_GPIO_IDX_4, /**< GPIO4*/  
    HI_GPIO_IDX_5, /**< GPIO5*/  
    HI_GPIO_IDX_6, /**< GPIO6*/  
    HI_GPIO_IDX_7, /**< GPIO7*/  
    HI_GPIO_IDX_8, /**< GPIO8*/  
    HI_GPIO_IDX_9, /**< GPIO9*/  
    HI_GPIO_IDX_10, /**< GPIO10*/  
    HI_GPIO_IDX_11, /**< GPIO11*/  
    HI_GPIO_IDX_12, /**< GPIO12*/  
    HI_GPIO_IDX_13, /**< GPIO13*/  
    HI_GPIO_IDX_14, /**< GPIO14*/  
  
    HI_GPIO_IDX_MAX, /**< Maximum value, which cannot be used.CNcomment:最大值，不可输入  
    使用 CNend*/  
}  
hi_gpio_idx;
```

**hi\_gpio\_dir dir:** dir, GPIO 输出方向

```
typedef enum {  
    HI_GPIO_DIR_IN = 0, /**< Input.CNcomment:输入方向 CNend*/  
    HI_GPIO_DIR_OUT /**< Output.CNcomment:输出方向 CNend*/  
}  
hi_gpio_dir;
```

● **hi\_u32 hi\_io\_set\_pull(hi\_io\_name id, hi\_io\_pull val);**

**hi\_io\_name id:** gpio 硬件管脚编号，对应的是硬件管脚上的 GPIO pin 脚，枚举类型如下：

```
typedef enum {  
    HI_IO_NAME_GPIO_0,    /**< GPIO0 */  
    HI_IO_NAME_GPIO_1,    /**< GPIO1 */  
    HI_IO_NAME_GPIO_2,    /**< GPIO2 */  
    HI_IO_NAME_GPIO_3,    /**< GPIO3 */  
    HI_IO_NAME_GPIO_4,    /**< GPIO4 */  
    HI_IO_NAME_GPIO_5,    /**< GPIO5 */  
}
```



```
HI_IO_NAME_GPIO_6,    /**< GPIO6 */
HI_IO_NAME_GPIO_7,    /**< GPIO7 */
HI_IO_NAME_GPIO_8,    /**< GPIO8 */
HI_IO_NAME_GPIO_9,    /**< GPIO9 */
HI_IO_NAME_GPIO_10,   /**< GPIO10 */
HI_IO_NAME_GPIO_11,   /**< GPIO11 */
HI_IO_NAME_GPIO_12,   /**< GPIO12 */
HI_IO_NAME_GPIO_13,   /**< GPIO13 */
HI_IO_NAME_GPIO_14,   /**< GPIO14 */
HI_IO_NAME_SFC_CSN,   /**< SFC_CSN */
HI_IO_NAME_SFC_IO1,   /**< SFC_IO1 */
HI_IO_NAME_SFC_IO2,   /**< SFC_IO2 */
HI_IO_NAME_SFC_IO0,   /**< SFC_IO0 */
HI_IO_NAME_SFC_CLK,   /**< SFC_CLK */
HI_IO_NAME_SFC_IO3,   /**< SFC_IO3 */
HI_IO_NAME_MAX,
} hi_io_name;
```

**hi\_io\_pull val:** IO 上下拉功能，枚举类型如下：

```
typedef enum {
    HI_IO_PULL_NONE,    /**< Disabled.CNcomment:无拉 CNend */
    HI_IO_PULL_UP,      /**< Pull-up enabled.CNcomment:上拉 CNend */
    HI_IO_PULL_DOWN,    /**< Pull-down enabled.CNcomment:下拉 CNend */
    HI_IO_PULL_MAX,     /**< Invalid.CNcomment:无效值 CNend */
} hi_io_pull;
```

例如：循迹模块 IO 口初始化在 demo 中示例：

```
hi_io_set_func(HI_IO_NAME_GPIO_11, HI_IO_FUNC_GPIO_5_GPIO);
hi_gpio_set_dir(HI_GPIO_IDX_11, HI_GPIO_DIR_IN);
hi_io_set_pull(HI_IO_NAME_GPIO_11, HI_IO_PULL_UP);

hi_io_set_func(HI_IO_NAME_GPIO_12, HI_IO_FUNC_GPIO_5_GPIO);
hi_gpio_set_dir(HI_GPIO_IDX_12, HI_GPIO_DIR_IN);
hi_io_set_pull(HI_IO_NAME_GPIO_12, HI_IO_PULL_UP);
```



(2) 当循迹模块检测到白线时, 指示二极管灯亮, 以一定的速度直走, 对应 IO 口为高电平; 当检测到黑线时, 指示二极管灯灭, 此时对应的 IO 口有一个下降沿, 此时让对应边的电机停止。中断处理函数为: `hi_u32 hi_gpio_register_isr_function(hi_gpio_idx id, hi_gpio_int_type int_type, hi_gpio_int_polarity int_polarity, gpio_isr_callback func, hi_void *arg);`

**hi\_gpio\_idx id:** 见上文

**hi\_gpio\_int\_type int\_type:** GPIO 中断触发方式, 枚举类型如下:

```
typedef enum {  
    HI_INT_TYPE_LEVEL = 0, /*电平触发中断 CNend */  
    HI_INT_TYPE_EDGE    /*边沿触发中断 CNend */  
} hi_gpio_int_type;
```

**hi\_gpio\_int\_polarity int\_polarity:** IO 中断极性, 与 `hi_gpio_int_type` 配合使用, 枚举类型如下:

```
typedef enum {  
    HI_GPIO_EDGE_FALL_LEVEL_LOW = 0, /*低电平或下降沿触发中断 CNend */  
    HI_GPIO_EDGE_RISE_LEVEL_HIGH    /*高电平或上升沿触发中断 CNend */  
} hi_gpio_int_polarity;
```

**gpio\_isr\_callback func:** GPIO 中断回调函数

**hi\_void \*arg:** void

对应 demo 中的中断处理如下:

```
ret = hi_gpio_register_isr_function(HI_GPIO_IDX_11, HI_INT_TYPE_EDGE, HI_GPIO_EDGE_FALL_LEVEL_LOW, gpio11_irq_callback, HI_NULL);  
ret = hi_gpio_register_isr_function(HI_GPIO_IDX_12, HI_INT_TYPE_EDGE, HI_GPIO_EDGE_FALL_LEVEL_LOW, gpio12_irq_callback, HI_NULL);
```

```
extern hi_u32 g_car_speed_left;  
extern hi_u32 g_car_speed_right;  
extern hi_u8 count;  
static hi_u8 speed_left_flag = 0;  
static hi_u8 speed_right_flag = 0;  
hi_void gpio11_irq_callback(hi_void *param)  
{  
    g_car_speed_left = 60000;  
    count = 0;  
}  
hi_void gpio12_irq_callback(hi_void *param)  
{  
    g_car_speed_right = 60000;  
    count = 0;  
}
```



(3) demo 中小车的循迹方法:

```
hi_void trace_module(hi_void)
{
    hi_u8 current_car_modular_control_module = g_car_modular_control_module;
    hi_u8 current_car_control_mode = g_car_control_mode;
    hi_gpio_value m_left_value = HI_GPIO_VALUE0;
    hi_gpio_value m_right_value = HI_GPIO_VALUE0;

    hi_u32 timer_id1;
    hi_timer_create(&timer_id1);
    hi_timer_start(timer_id1, HI_TIMER_TYPE_PERIOD, 1, timer1_callback, 0);
    gpio_control(HI_IO_NAME_GPIO_0, HI_GPIO_IDX_0, HI_GPIO_DIR_OUT, HI_GPIO_VALUE1, HI_IO_FUNC_GPIO_0_GPIO);
    pwm_control(HI_IO_NAME_GPIO_1, HI_IO_FUNC_GPIO_1_PWM4_OUT, HI_PWM_PORT_PWM4, g_car_speed_left);
    gpio_control(HI_IO_NAME_GPIO_9, HI_GPIO_IDX_9, HI_GPIO_DIR_OUT, HI_GPIO_VALUE1, HI_IO_FUNC_GPIO_9_GPIO);
    pwm_control(HI_IO_NAME_GPIO_10, HI_IO_FUNC_GPIO_10_PWM1_OUT, HI_PWM_PORT_PWM1, g_car_speed_right);
    while (1) {
        hi_pwm_start(HI_PWM_PORT_PWM4, g_car_speed_left, 60000);
        hi_pwm_start(HI_PWM_PORT_PWM1, g_car_speed_right, 60000);

        hi_udelay(1);

        if ((current_car_modular_control_module != g_car_modular_control_module)
            || (current_car_control_mode != g_car_control_mode)) {
            break;
        }
    }
    hi_timer_delete(timer_id1);
}
```

设置周期定时器，每1ms进入一次回调函数

小车的左(右)轮以g\_car\_speed\_left(g\_car\_speed\_right)速度往前

```
#define car_speed_left 100
#define car_speed_right 100
hi_u32 g_car_speed_left = car_speed_left;
hi_u32 g_car_speed_right = car_speed_right;
hi_u8 count = 0;
hi_gpio_value io_status_left;
hi_gpio_value io_status_right;
hi_void timer1_callback(hi_u32 arg)
{
    hi_gpio_value io_status;
    if(g_car_speed_left != car_speed_left)
    {
        count++;
        if(count >=2)
        {
            hi_gpio_get_input_val(HI_GPIO_IDX_11, &io_status);
            if(io_status != HI_GPIO_VALUE0){
                g_car_speed_left = car_speed_left;
                printf("left speed change \r\n");
                count = 0;
            }
        }
    }
    if(g_car_speed_right != car_speed_right)
    {
        count++;
        if(count >=2)
        {
            hi_gpio_get_input_val(HI_GPIO_IDX_12, &io_status);
            if(io_status != HI_GPIO_VALUE0){
                g_car_speed_right = car_speed_right;
                printf("right speed change \r\n");
                count = 0;
            }
        }
    }
}
```

在转过合适角度后，恢复左轮速度

在转过合适角度后，恢复右轮速度