HiSpark-智能小车套件 超声波测距编程指南

文档版本 00B01

发布日期 2020/8/3

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

1 智能小车套件超声波测距功能实现

1.1 超声波测距硬件准备

图 1.1-1 超声波测距模块

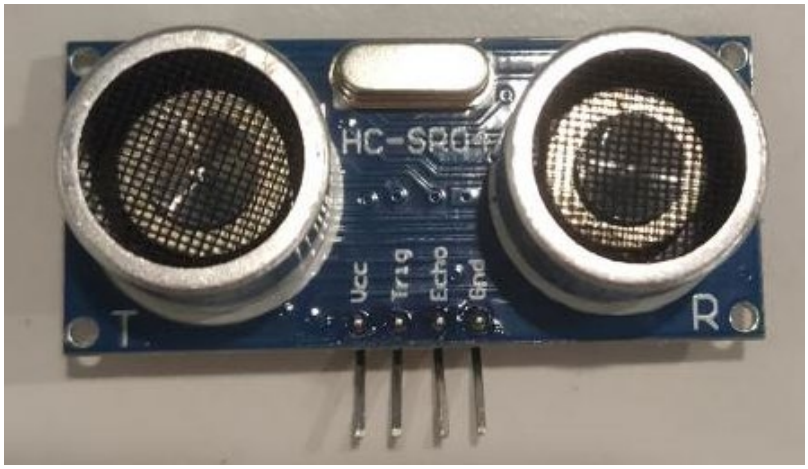
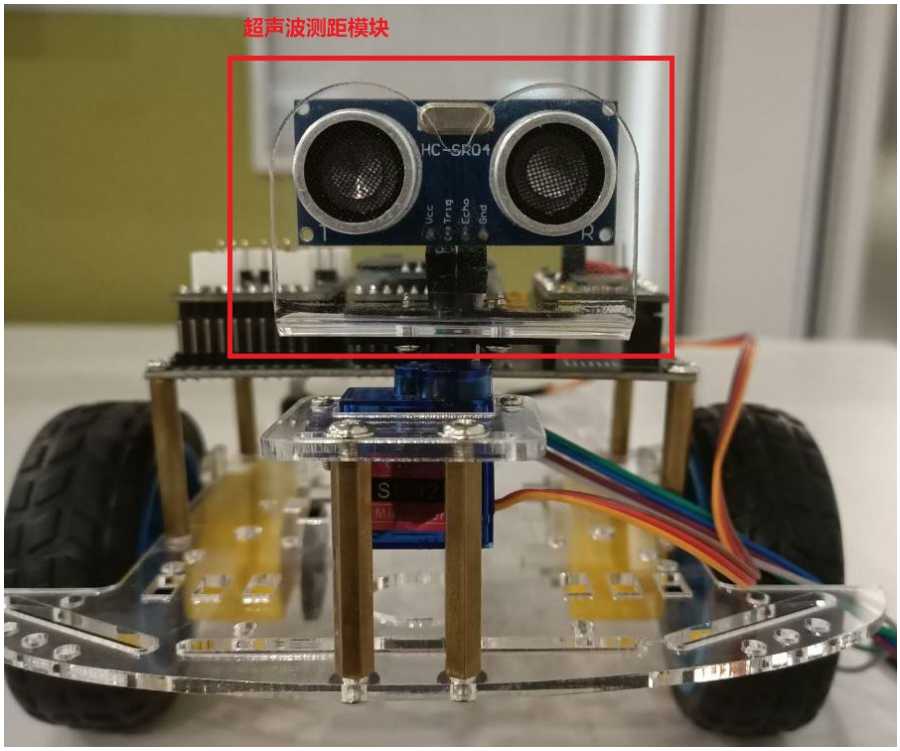


图 1.1-2 超声波测距模块在智能小车上的安装



1.2 超声波测距工作原理介绍

1. 所采用模块为 HC-SR04 超声波测距模块，可提供 2cm-400cm 的非接触式距离感测功能。测距精度可达到 3mm ，基本工作原理为：

（1）采用 IO 口 TRIG 触发测距，给至少 10us 的高电平信号。

（2）模块自动发送 8 个 40hz 的方波检测是否有信号返回。

（3）有信号返回，通过 IO 口 ECHO 输入一个高电平，高电平持续时间就是超声波从发射到返回的时间。

（4）测试距离 （m）= （（高电平时间 （s））*声速（340m/S））/2。

或 测试距离（cm） = 高电平时间 （us）* (1 0 00/ 17 us/cm) = 高电平时间/58.

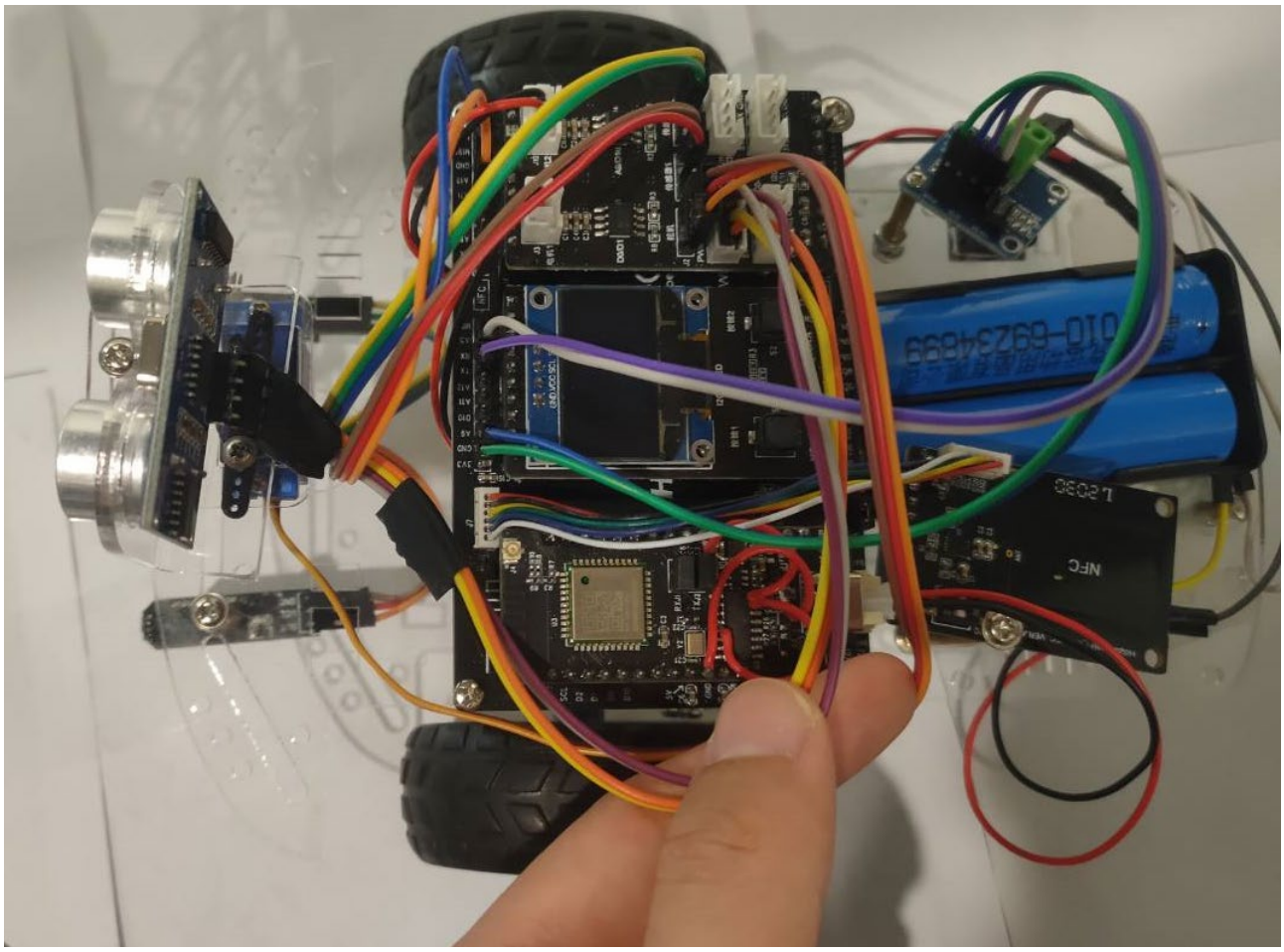
2. 超声波引脚连接如下：

TRIG ------ GPIO_7

ECHO ------ GPIO_8

超声波模块在小车上的连线如下所示：

图 1.2-1 超声波模块接线图



1.3 超声波测距功能软件实现

1． 首先进行 IO 引脚的初始化工作，主要为以下三个函数：

hi_u32 hi_io_set_func(hi_io_name id, hi_u8 val); //配置引脚复用

hi_u32 hi_gpio_set_dir(hi_gpio_idx id, hi_gpio_dir dir); //设置 IO 方向（输入/输出）

hi_u32 hi_gpio_set_ouput_val(hi_gpio_idx id, hi_gpio_value val);//设置电平状态

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_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，取值范围如下：

```
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_gpio_value gpio_val ：gpio 输出状态，取值范围如下：

```
typedef enum {
    HI_GPIO_VALUE0 = 0, /**< Low level.CNcomment:低电平 CNend*/
    HI_GPIO_VALUE1 /**< High level.CNcomment:高电平 CNend*/
} hi_gpio_value;
```

在 demo 中示例：

（1）超声波测距初始化：

```
/*超声波trig口，设置为输出模式*/
hi_io_set_func(HI_IO_NAME_GPIO_7, HI_IO_FUNC_GPIO_7_GPIO);
hi_gpio_set_dir(HI_GPIO_IDX_7, HI_GPIO_DIR_OUT);
hi_gpio_set_ouput_val(HI_GPIO_IDX_7, HI_GPIO_VALUE0);

/*超声波echo口，先设置为输出模式，后续设置为输入模式*/
hi_io_set_func(HI_IO_NAME_GPIO_8, HI_IO_FUNC_GPIO_8_GPIO);
hi_gpio_set_dir(HI_GPIO_IDX_8, HI_GPIO_DIR_OUT);
hi_gpio_set_ouput_val(HI_GPIO_IDX_8, HI_GPIO_VALUE0);
```

（2）后续超声波测距实现如下：

```
hi_float car_get_distance(hi_void)
{
    static hi_u64 start_time = 0;
    hi_u64 end_time = 0;
    hi_float distance = 0.0;
    hi_gpio_value val = HI_GPIO_VALUE0;
    int gpio_8_level = GPIO_8_IS_LOW_LEVEL;

    /*超声波echo口，设置为输入模式*/
    hi_io_set_func(HI_IO_NAME_GPIO_8, HI_IO_FUNC_GPIO_8_GPIO);
    hi_gpio_set_dir(HI_GPIO_IDX_8, HI_GPIO_DIR_IN);

    /* 给trig发送至少10us的高电平脉冲，以触发传感器测距 */
    hi_gpio_set_ouput_val(HI_GPIO_IDX_7,HI_GPIO_VALUE1);
    hi_udelay(20);
    hi_gpio_set_ouput_val(HI_GPIO_IDX_7,HI_GPIO_VALUE0);

    /*计算与障碍物之间的距离*/
    while (1) {
        hi_gpio_get_input_val(HI_GPIO_IDX_8,&val);

        /*echo为高电平时开始计时*/
        if ((val == HI_GPIO_VALUE1) && (gpio_8_level == GPIO_8_IS_LOW_LEVEL)) {
            start_time = hi_get_us();
            gpio_8_level = GPIO_8_IS_HIGH_LEVEL;
        }

        /*echo为低电平时纪录结束时间*/
        if ((val == HI_GPIO_VALUE0) && (gpio_8_level == GPIO_8_IS_HIGH_LEVEL)) {
            end_time = hi_get_us() - start_time;
            start_time = 0;
            break;
        }
    }
    distance = end_time/DISTANCE_FORMULA;

    return distance;
}
```

计算超声波往返时间

计算距离