HiSpark-WiFi-IoT 光敏传感器编程指南

文档版本　00B01

发布日期　2020/8/6

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
2020-08-6	00B01	第一次临时版本发布。

目 录

1 WIFI-IoT 开发板光敏电阻功能实现

1.1 WIFI-IOT 开发板套件——光敏电阻工作原理

1. 光敏电阻结构与工作原理：

GL5537-1 光敏电阻本身没有极性，是一个纯粹的电阻器件，使用时可加直流电压，也可加交流电压。

无光照时，电阻阻值很大（这时通过 ADC 采集的电压最大），电流中电流很小。当光敏电阻受到一定波长范围的光照时，阻值急剧减少，电路中电流迅速增大，电阻越大，灵敏度越高。

图 1.1-1 工作原理

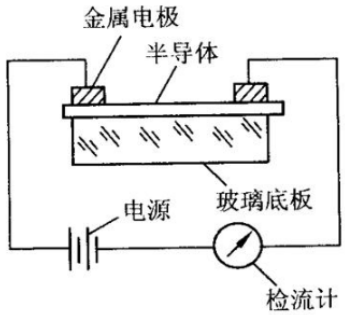


图 1.1-2 结构

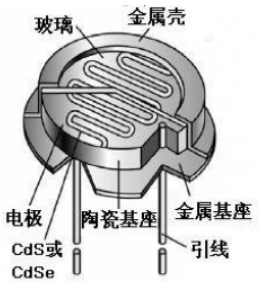
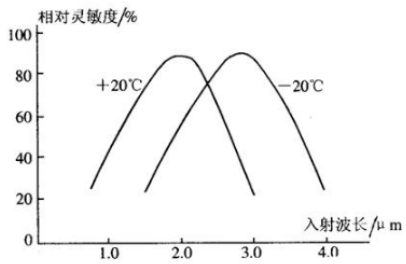


图 1.1-3 相对灵敏度



2. 光敏电阻主要参数：

1）亮电阻（kΩ）：指光敏电阻器受到光照射时的电阻值；

2）暗电阻(MΩ)：指光敏电阻器在无光照射（黑暗环境）时的电阻值；

3）最高工作电压(V)：指光敏电阻器在额定功率下所允许承受的最高电压；

4）亮电流：指光敏电阻器在规定的外加电压下受到光照射时所通过的电流；

5）暗电流(mA)：指在无光照射时，光敏电阻器在规定的外加电压下通过的电流；

6）时间常数（s）：指光敏电阻器从光照跃变开始到稳定亮电流的 63%时所需的时间；

7）电阻温度系数：指光敏电阻器在环境温度改变 1℃时，其电阻值的相对变化；

8）灵敏度：指光敏电阻器在有光照射和无光照射时电阻值的相对变化。

1.2 光敏传感器功能实现逻辑

1. HiSpark 开发板中，代码中采用 ADC 采集光敏电阻电压变化，判断光照强度，根据光照强度进行业务逻辑运算。

2. HiSpark 开发板光敏电阻接在 GPIO9 端口，端口 ADC 复用通道为 ADC_Channal_4,由此可以设置 ADC 初始化和端口复用功能。

Digital	UART0	UART1/2	SPI0	SPI1	ADC	I2C
GPIO_00		UART1_TXD		SPI1_CLK		I2C1_SDA
GPIO_01		UART1_RXD		SPI1_RXD		I2C1_SCL
GPIO_02		UART1_RTS		SPI1_TXD		
GPIO_03	UART0_LOG_TXD	UART1_CTS		SPI1_CS1		I2C1_SDA
GPIO_04	UART0_LOG_RXD				ADC1	I2C1_SCL
GPIO_05		UART1_RXD	SPI0_CS1		ADC2	
GPIO_06		UART1_TXD	SPI0_CLK			
GPIO_07		UART1_CTS	SPI0_RXD		ADC3	
GPIO_08		UART1_RTS	SPI0_TXD			
GPIO_09		UART2_RTS	SPI0_TXD		ADC4	I2C0_SCL

1.3 光敏电阻功能软件实现

1. LiteOS 打开 WIFI-IOT 的 SDK 后，首先初始化 GPIO 配置

hi_u32 hi_adc_init(hi_void)：ADC 初始化接口，使能 ADC 模块。

2. hi_u32 hi_adc_read(hi_adc_channel_index channel, hi_u16 *data, hi_adc_equ_model_sel equ_model, hi_adc_cur_bais cur_bais, hi_u16 delay_cnt): 读取 ADC 数值。

其中 hi_adc_read 接口参数含义如下：

hi_adc_channel_index channel：选择 ADC 通道，枚举类型如下：

```
typedef enum {
    HI_ADC_CHANNEL_0,
    HI_ADC_CHANNEL_1,
    HI_ADC_CHANNEL_2,
    HI_ADC_CHANNEL_3,
    HI_ADC_CHANNEL_4,
    HI_ADC_CHANNEL_5,
    HI_ADC_CHANNEL_6,
    HI_ADC_CHANNEL_7,
    HI_ADC_CHANNEL_BUTT,
} hi_adc_channel_index;
```

hi_u16 *data：将读取回来的 ADC 数值保存在 data 数组中。

hi_adc_equ_model_sel equ_model：选择平均算法模式，枚举类型如下：

```
typedef enum {
    HI_ADC_EQU_MODEL_1,     /**< 0：The average value is not used.CNcomment:1 次平均，即不进行平均 CNend */
    HI_ADC_EQU_MODEL_2,     /**< 1：2-time average algorithm mode.
                            CNcomment:2 次平均算法模式 CNend */
    HI_ADC_EQU_MODEL_4,     /**< 2：4-time average algorithm mode.
                            CNcomment:4 次平均算法模式 CNend */
    HI_ADC_EQU_MODEL_8,     /**< 3：8-time average algorithm mode.
                            CNcomment:8 次平均算法模式 CNend */
    HI_ADC_EQU_MODEL_BUTT,
} hi_adc_equ_model_sel;
```

hi_adc_cur_bais cur_bais: 模拟电源控制，枚举类型如下：

```
typedef enum {
    HI_ADC_CUR_BAIS_DEFAULT,        /**< 0：Auto control.
                                    CNcomment:自动识别模式 */
    HI_ADC_CUR_BAIS_AUTO,           /**< 1：Auto control.
                                    CNcomment:自动识别模式 */
    HI_ADC_CUR_BAIS_1P8V,           /**< 2：Manual control, AVDD=1.8V.
                                    CNcomment:手动控制，AVDD=1.8V */
    HI_ADC_CUR_BAIS_3P3V,           /**< 3：Manual control, AVDD=3.3V.
                                    CNcomment:手动控制，AVDD=3.3V */
    HI_ADC_CUR_BAIS_BUTT,
} hi_adc_cur_bais;
```

hi_u16 delay_cnt: 从配置采样到启动采样的延时时间计数，一次计数 334ns,范围 0~0xFF0 之间。

3. HiSpark 开发板上 ADC 初始化配置如下：

图 1.3-1

```
ret = hi_cipher_init();
if (ret != HI_ERR_SUCCESS) {
    err_info |= PERIPHERAL_INIT_ERR_CIPHER;
}
hi_adc_init();
```

读取 ADC 数值并进行电压转化：

图 1.3-2

```
hi_u8 get_light_status(hi_void)
{
    int i;
    hi_u16 data;
    hi_u32 ret;
    hi_u16 vlt;
    float voltage;
    float vlt_max = 0;
    float vlt_min = VLT_MIN;
    memset_s(g_adc_buf, sizeof(g_adc_buf), 0x0, sizeof(g_adc_buf));
    for (i = 0; i < ADC_TEST_LENGTH; i++) {
        ret = hi_adc_read(HI_ADC_CHANNEL_4, &data, HI_ADC_EQU_MODEL_4, HI_ADC_CUR_BAIS_DEFAULT, 0xF0); //ADC_Channal_6  自动识别模式  CNcomment:4次平均算法模式 CNend */
        if (ret != HI_ERR_SUCCESS) {
            printf("ADC Read Fail\n");
            return  HI_NULL;
        }
        g_adc_buf[i] = data;
    }

    for (i = 0; i < ADC_TEST_LENGTH; i++) {
        vlt = g_adc_buf[i];
        voltage = (float)vlt * 1.8 * 4 / 4096.0;  /* vlt * 1.8 * 4 / 4096.0为将码字转换为电压 */
        vlt_max = (voltage > vlt_max) ? voltage : vlt_max;
        vlt_min = (voltage < vlt_min) ? voltage : vlt_min;
    }

    if (vlt_max >3.0) {
        return OLED_FALG_ON;
    } else {
        return OLED_FALG_OFF;
    }
}
```

ADC采集数值进行电压值转化

根据电压阈值做业务处理

4. HiSpark 开发板将 ADC 采集电压阈值判断，点亮/点灭三色灯，如图所示：

图 1.3-3

```
/*
mode:6.Light detect
模式6: 光敏检测模式
    无光灯亮，有光灯灭
*/
hi_void light_detect_sample(hi_void)
{
    hi_u8 current_mode = g_current_mode;
    while (1) {
        g_with_light_flag = get_light_status();

        if (g_with_light_flag == OLED_FALG_ON) {

            hi_pwm_start(HI_PWM_PORT_PWM1, PWM_SMALL_DUTY, PWM_MIDDLE_DUTY);
            hi_pwm_start(HI_PWM_PORT_PWM2, PWM_SMALL_DUTY, PWM_MIDDLE_DUTY);
            hi_pwm_start(HI_PWM_PORT_PWM3, PWM_SMALL_DUTY, PWM_MIDDLE_DUTY);
        } else if (g_with_light_flag == OLED_FALG_OFF) {

            hi_pwm_start(HI_PWM_PORT_PWM1, PWM_LOW_DUTY, PWM_MIDDLE_DUTY);
            hi_pwm_start(HI_PWM_PORT_PWM2, PWM_LOW_DUTY, PWM_MIDDLE_DUTY);
            hi_pwm_start(HI_PWM_PORT_PWM3, PWM_LOW_DUTY, PWM_MIDDLE_DUTY);
        }

        if (current_mode != g_current_mode) {
            hi_pwm_start(HI_PWM_PORT_PWM1, PWM_LOW_DUTY, PWM_MIDDLE_DUTY);
            hi_pwm_start(HI_PWM_PORT_PWM2, PWM_LOW_DUTY, PWM_MIDDLE_DUTY);
            hi_pwm_start(HI_PWM_PORT_PWM3, PWM_LOW_DUTY, PWM_MIDDLE_DUTY);
            break;
        }
        hi_sleep(SLEEP_1_MS);
    }
}
```