3.2 点亮 RGB 灯

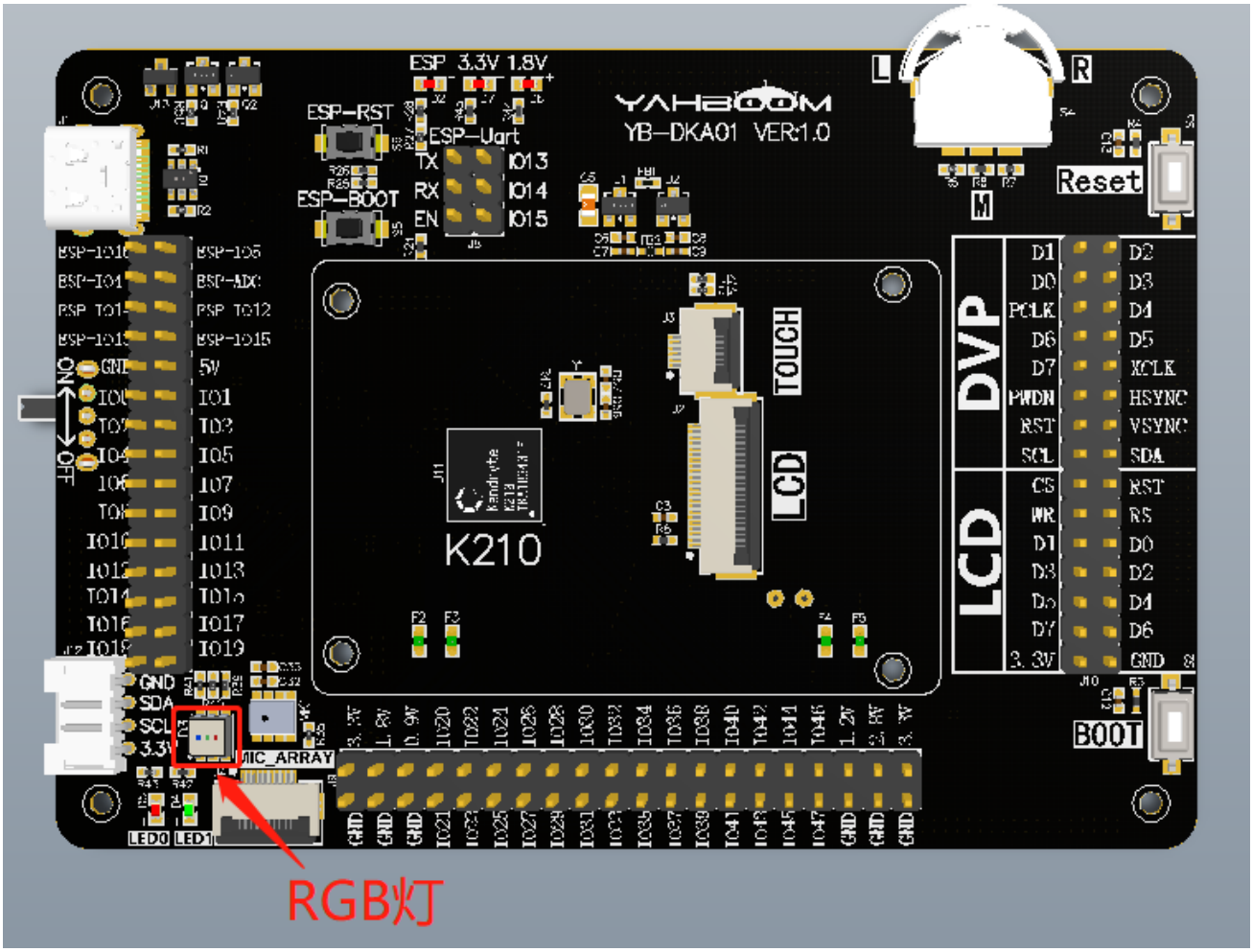
一、实验目的

本节课主要学习 K210 的高速 GPIOHS，点亮 RGB 灯。

二、实验准备

1.实验元件

RGB 灯

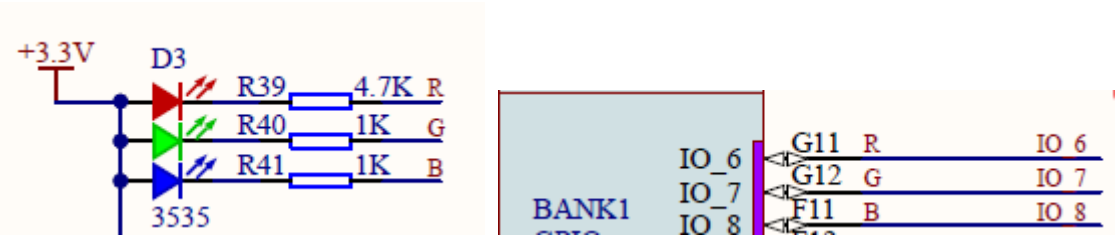


2.元件特性

RGB 灯可以点亮红色、绿色、蓝灯等颜色，再根据红色绿色蓝色的不同亮度组合成其他颜色，例如黄色、紫色等。

3.硬件连接

K210 开发板出厂默认已经焊接好 RGB 灯。RGB 灯 R 连接的是 IO6，G 连接的是 IO7，B 连接的是 IO8。



4.SDK 中对应 API 功能

对应的头文件 gpiohs.h

高速 goiohs 总共有 32 个，可配置输入输出模式，可配置上拉下拉或高阻，每个 IO 都有单独的中断源，中断支持边沿和电平触发，每个 IO 都可以自由分配到 FPIOA 上的 48 个引脚之一。

为用户提供以下接口：

- gpiohs_set_drive_mode：设置输入输出模式
- gpiohs_set_pin：设置引脚电平
- gpiohs_get_pin：读取引脚电平
- gpiohs_set_pin_edge：设置中断触发电平
- gpiohs_set_irq (0.6.0 后不再支持，请使用gpiohs_irq_register)
- gpiohs_irq_register：注册引脚中断服务
- gpiohs_irq_unregister：注销引脚中断

三、实验原理

RGB 灯内部是由三颗颜色分别为红色、绿灯和蓝色的 LED 组成，所以发光原理与 LED 是一样的。不同的是三颗 LED 灯靠得很近，这样就可以设置不同的颜色，来达到展示不同颜色的功能。

四、实验过程

1. 首先根据上面的硬件连接引脚图，K210 的硬件引脚和软件功能使用的是 FPIOA 映射关系。

这里要注意的是程序里操作的都是软件引脚，所以需要先把硬件引脚映射成软件 GPIO 功能，操作的时候直接操作软件 GPIO 即可。

```
/*****************************HARDWARE-PIN*********************
// 硬件IO口，与原理图对应
#define PIN_RGB_R             (6)
#define PIN_RGB_G             (7)
#define PIN_RGB_B             (8)

/*****************************SOFTWARE-GPIO*********************
// 软件GPIO口，与程序对应
#define RGB_R_GPIONUM          (0)
#define RGB_G_GPIONUM          (1)
#define RGB_B_GPIONUM          (2)

/*****************************FUNC-GPIO*************************
// GPIO口的功能，绑定到硬件IO口
#define FUNC_RGB_R             (FUNC_GPIOHS0 + RGB_R_GPIONUM)
#define FUNC_RGB_G             (FUNC_GPIOHS0 + RGB_G_GPIONUM)
#define FUNC_RGB_B             (FUNC_GPIOHS0 + RGB_B_GPIONUM)
```

```
void hardware_init(void)
{
    // fpioa映射
    fpioa_set_function(PIN_RGB_R, FUNC_RGB_R);
    fpioa_set_function(PIN_RGB_G, FUNC_RGB_G);
    fpioa_set_function(PIN_RGB_B, FUNC_RGB_B);

}
```

2. 在使用 RGB 灯前需要初始化，把 RGB 灯的软件 GPIO 设置为输出模式。

```
void init_rgb(void)
{
    // 设置RGB灯的GPIO模式为输出
    gpiohs_set_drive_mode(RGB_R_GPIONUM, GPIO_DM_OUTPUT);
    gpiohs_set_drive_mode(RGB_G_GPIONUM, GPIO_DM_OUTPUT);
    gpiohs_set_drive_mode(RGB_B_GPIONUM, GPIO_DM_OUTPUT);

    // 关闭RGB灯
    rgb_all_off();
}
```

3.然后关闭 RGB 灯，同样是设置 RGB 灯的 GPIO 为高电平则可以让 RGB 灯熄灭。

```
void rgb_all_off(void)
{
    gpiohs_set_pin(RGB_R_GPIONUM, GPIO_PV_HIGH);
    gpiohs_set_pin(RGB_G_GPIONUM, GPIO_PV_HIGH);
    gpiohs_set_pin(RGB_B_GPIONUM, GPIO_PV_HIGH);
}
```

4.最后在 while 循环中每 0.5 秒修改 state 的值，从而改变 RGB 灯的颜色，由于不清楚具体上一次亮的是哪个灯，所以在每次设置灯的颜色前都把 RGB 灯关闭。state = state % 3;语句的功能是把 state 的值限制为 0,1,2,刚好与软件 GPIO 对应。

```
int main(void)
{
    // RGB灯状态，0=红灯亮，1=绿灯亮，2=蓝灯亮
    int state = 0;

    // 硬件引脚初始化
    hardware_init();
    // 初始化RGB灯
    init_rgb();

    while (1)
    {
        rgb_all_off();    // 先关闭RGB灯
        // 根据state的值点亮不同颜色的灯
        gpiohs_set_pin(state, GPIO_PV_LOW);
        msleep(500);
        state++;
        state = state % 3;
    }
    return 0;
}
```

5.编译调试，烧录运行

把本课程资料中的 gpiohs_rgb 复制到 SDK 中的 src 目录下，

然后进入 build 目录，运行以下命令编译。

```
cmake .. -DPROJ=gpiohs_rgb -G "MinGW Makefiles"
make
```

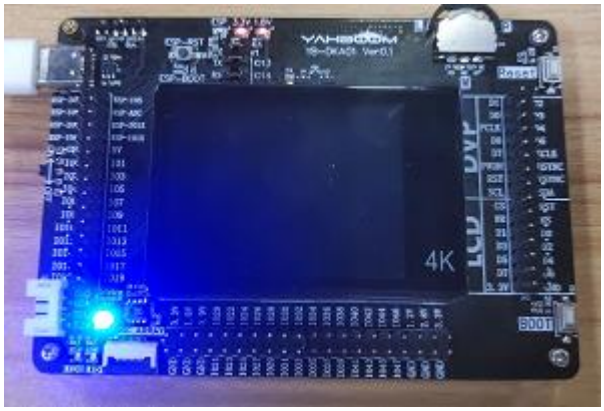
```
[ 97%] Building C object CMakeFiles/gpiohs_rgb.dir/src/gpiohs_rgb/main.c.obj
[100%] Linking C executable gpiohs_rgb
Generating .bin file ...
[100%] Built target gpiohs_rgb
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build>
```

编译完成后，在 build 文件夹下会生成 gpiohs_rgb.bin 文件。

使用 type-C 数据线连接电脑与 K210 开发板，打开 kflash，选择对应的设备，再将程序固件烧录到 K210 开发板上。

五、实验现象

RGB 灯会亮红灯，0.5 秒后亮绿灯，再 0.5 秒后亮蓝灯，接着 0.5 秒后再亮红灯，以此循环，每 0.5 秒切换一次颜色。



六、实验总结

1.RGB 灯与 LED 灯的点亮原理是一样的。

2.高速 gpiohs 每个 IO 都有单独的中断源,而通用 gpio 是共享一个总中断源的。

附：API

对应的头文件 gpiohs.h

gpiohs_set_drive_mode

描述

设置 GPIO 驱动模式。

函数原型

```
void gpiohs_set_drive_mode(uint8_t pin, gpio_drive_mode_t mode)
```

参数

参数名称	描述	输入输出
pin	GPIO 管脚	输入
mode	GPIO 驱动模式	输入

返回值

无。

gpio_set_pin

描述

设置 GPIO 管脚值。

函数原型

```
void gpiohs_set_pin(uint8_t pin, gpio_pin_value_t value)
```

参数

参数名称	描述	输入输出
pin	GPIO 管脚	输入
value	GPIO 值	输入

返回值

无。

gpio_get_pin

描述

获取 GPIO 管脚值。

函数原型

```
gpio_pin_value_t gpiohs_get_pin(uint8_t pin)
```

参数

参数名称	描述	输入输出
pin	GPIO 管脚	输入

返回值

获取的 GPIO 管脚值。

gpiohs_set_pin_edge

描述

设置高速 GPIO 中断触发模式。

函数原型

```
void gpiohs_set_pin_edge(uint8_t pin, gpio_pin_edge_t edge)
```

参数

参数名称	描述	输入输出
pin	GPIO 管脚	输入
edge	中断触发方式	输入

返回值

无。

gpiohs_set_irq

描述

设置高速 GPIO 的中断回调函数。

函数原型

```
void gpiohs_set_irq(uint8_t pin, uint32_t priority, void(*func)());
```

参数

参数名称	描述	输入输出
pin	GPIO 管脚	输入
priority	中断优先级	输入
func	中断回调函数	输入

返回值

无。

gpiohs_irq_register

描述

设置高速 GPIO 的中断回调函数。

函数原型

```
void gpiohs_irq_register(uint8_t pin, uint32_t priority, plic_irq_callback_t
callback, void *ctx)
```

参数

参数名称	描述	输入输出
pin	GPIO 管脚	输入

参数名称	描述	输入输出
priority	中断优先级	输入
plic_irq_callback_t	中断回调函数	输入
ctx	回调函数参数	输入

返回值

无。

gpiohs_irq_unregister

描述

注销 GPIOHS 中断。

函数原型

```
void gpiohs_irq_unregister(uint8_t pin)
```

参数

参数名称	描述	输入输出
pin	GPIO 管脚	输入

返回值

无。

数据类型

相关数据类型、数据结构定义如下：

- gpio_drive_mode_t：GPIO 驱动模式。
- gpio_pin_value_t：GPIO 值。
- gpio_pin_edge_t：GPIO 边沿触发模式。

gpio_drive_mode_t

描述

GPIO 驱动模式。

定义

```
typedef enum _gpio_drive_mode
{
    GPIO_DM_INPUT,
    GPIO_DM_INPUT_PULL_DOWN,
    GPIO_DM_INPUT_PULL_UP,
    GPIO_DM_OUTPUT,
} gpio_drive_mode_t;
```

成员

成员名称	描述
GPIO_DM_INPUT	输入
GPIO_DM_INPUT_PULL_DOWN	输入下拉
GPIO_DM_INPUT_PULL_UP	输入上拉
GPIO_DM_OUTPUT	输出

gpio_pin_value_t

描述

GPIO 值。

定义

```
typedef enum _gpio_pin_value
{
    GPIO_PV_LOW,
    GPIO_PV_HIGH
} gpio_pin_value_t;
```

成员

成员名称	描述
GPIO_PV_LOW	低
GPIO_PV_HIGH	高

gpio_pin_edge_t

描述

高速 GPIO 边沿触发模式。

定义

```
typedef enum _gpio_pin_edge
{
    GPIO_PE_NONE,
    GPIO_PE_FALLING,
    GPIO_PE_RISING,
    GPIO_PE_BOTH,
    GPIO_PE_LOW,
    GPIO_PE_HIGH = 8,
} gpio_pin_edge_t;
```

成员

成员名称	描述
GPIO_PE_NONE	不触发
GPIO_PE_FALLING	下降沿触发
GPIO_PE_RISING	上升沿触发
GPIO_PE_BOTH	双沿触发
GPIO_PE_LOW	低电平触发
GPIO_PE_HIGH	高电平触发