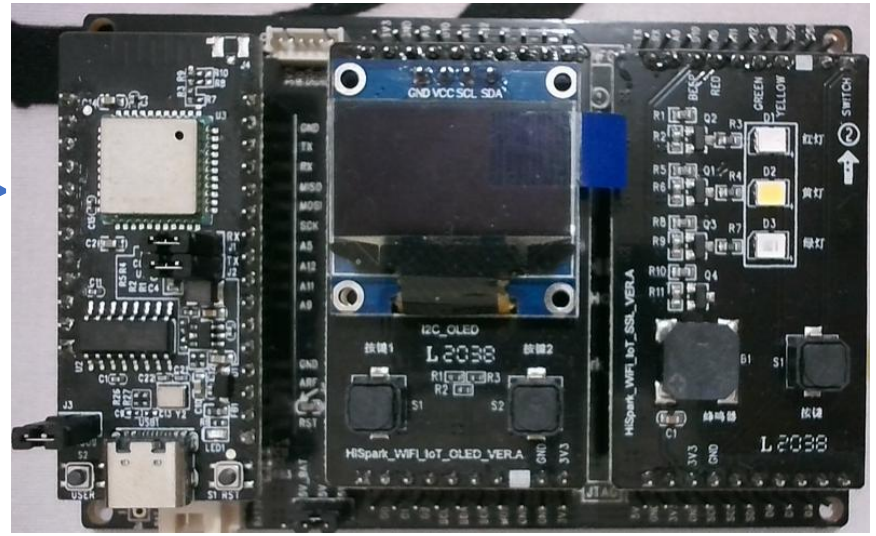
Harmony OS 智能硬件
入门系列课程 <快速上手>

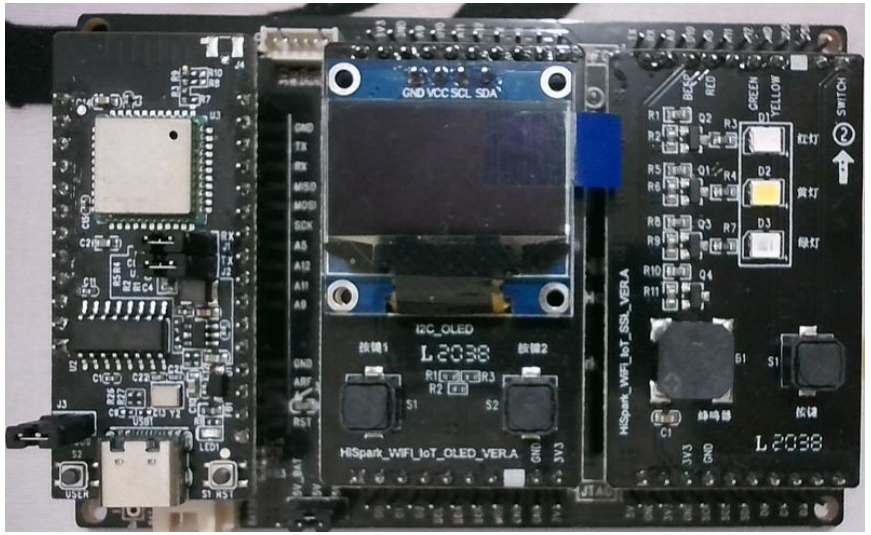


快速掌握Hi3861开发板基础开发技巧

第 3 讲：Hi3861外设——智能红绿灯板开发

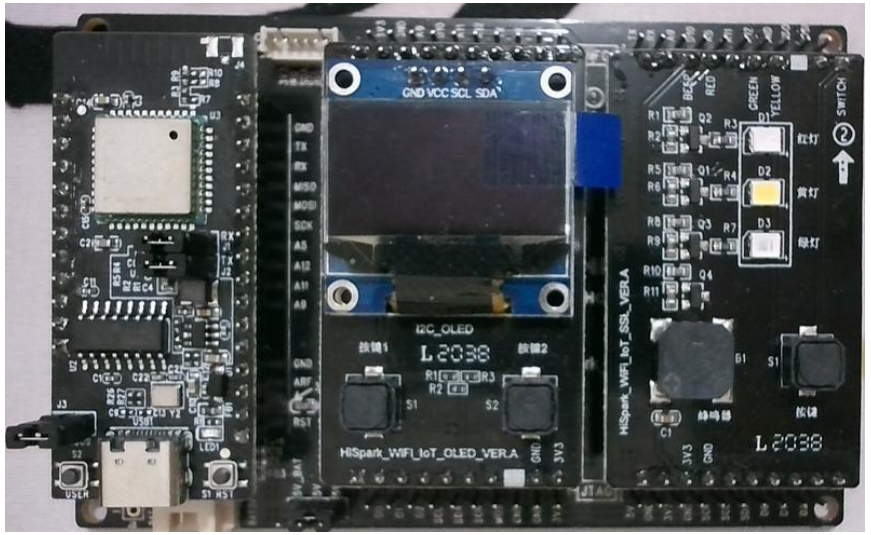
本讲内容

- 第1节：核心板上LED灯开发
- 第2节：交通灯功能实现
- 第3节：控制蜂鸣器发出声音
- 第4节：蜂鸣器实现播放音乐



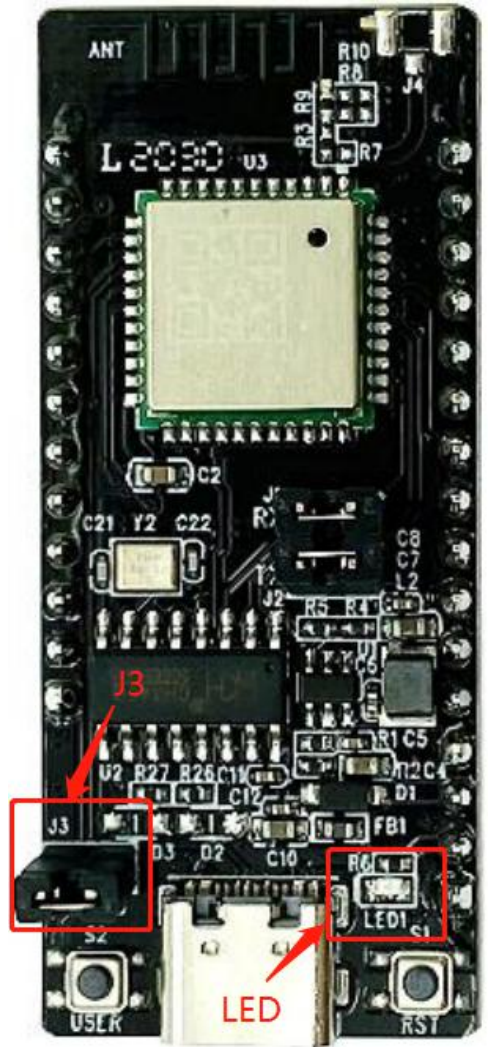
本讲目标

- 1、会分析电路图
- 2、掌握GPIO引脚开发步骤
- 3、掌握led开发方法
- 4、掌握线程开发

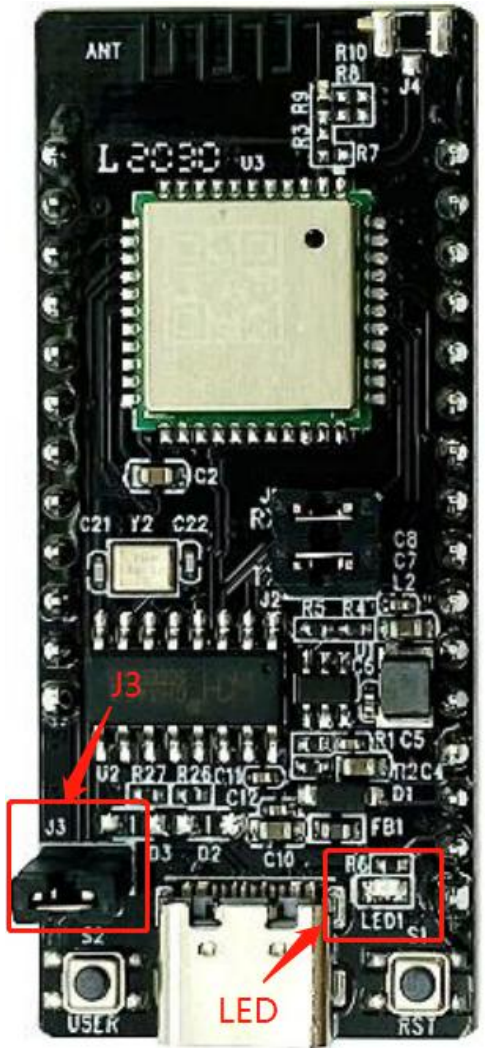


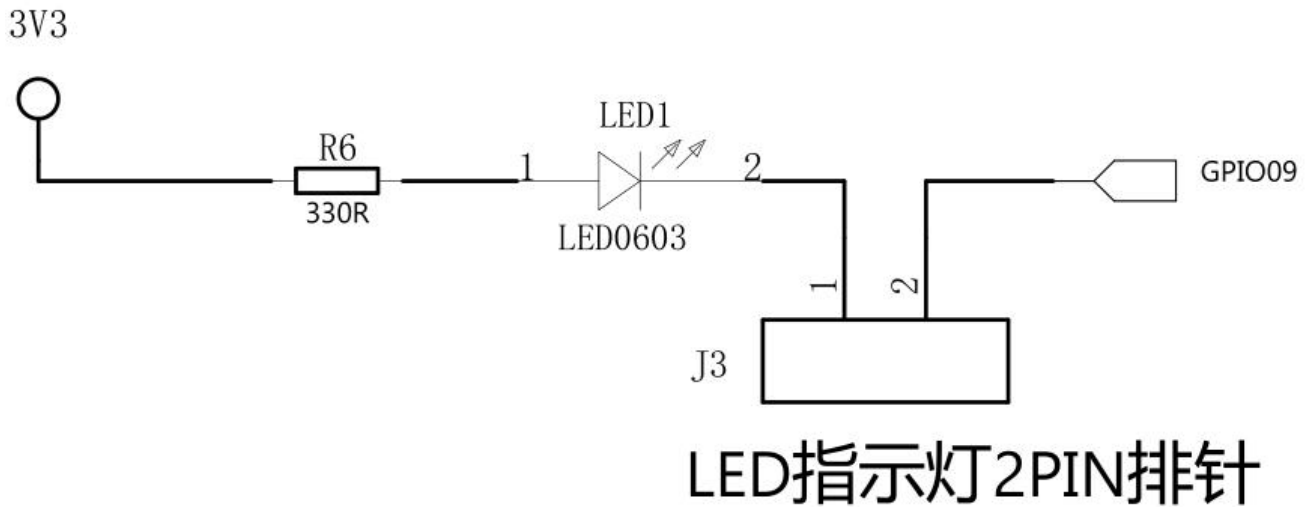
第1节：核心板上LED灯开发

- 知识点1：电路图分析
- 知识点2：代码实现LED灯亮灭



知识点1【电路图分析】





LED指示灯2PIN排针

- 用跳线帽连接 J3 的两个 pin
- 使用GPIO9
- 输出低电平点亮
- 输出高电平熄灭

知识点2【代码实现LED灯亮灭】

实现步骤：

1. 初始化GPIO设备`
2. 设置管脚WIFI_IOT_IO_NAME_GPIO_9功能
3. 设置引脚WIFI_IOT_IO_NAME_GPIO_9方向为输出
4. 给管脚WIFI_IOT_IO_NAME_GPIO_9输出低电平点亮LED灯
5. 给管脚WIFI_IOT_IO_NAME_GPIO_9输出高电平熄灭LED灯

知识点2【代码实现LED灯亮灭】

led_demo.c

<code/>

```
#include <stdio.h>
#include "ohos_init.h"
#include <unistd.h>
#include "wifiiot_gpio.h"
#include "wifiiot_gpio_ex.h"
#include "ohos_types.h"
void leddemo(void)
{
    GpioInit(); //初始化GPIO设备
    IoSetFunc(WIFI_IOT_IO_NAME_GPIO_9, WIFI_IOT_IO_FUNC_GPIO_9_GPIO);
    GpioSetDir(WIFI_IOT_IO_NAME_GPIO_9, WIFI_IOT_GPIO_DIR_OUT);
    GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_9, WIFI_IOT_GPIO_EDGE_FALL_LEVEL_LOW);
    printf("led 点亮\n");

    usleep(4000000);
    GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_9, WIFI_IOT_GPIO_EDGE_RISE_LEVEL_HIGH);
    printf("led 熄灭\n");
}
SYS_RUN(leddemo);
```

知识点2【代码实现LED灯亮灭】

./applications/sample/wifi-iot/app/chapter_03/BUILD.gn文件

<code/>

```
static_library("chapter_03_demo") {
    sources = [
        "led_demo.c",
    ]

    include_dirs = [
        "//utils/native/lite/include",
        "//base/iot_hardware/interfaces/kits/wifiiot_lite",
    ]
}
```

知识点2【代码实现LED灯亮灭】

./applications/sample/wifi-iot/app/BUILD.gn文件

<code/>

```
import("//build/lite/config/component/lite_component.gni")

lite_component("app") {
    features = [
        "chapter_03:chapter_03_demo",
    ]
}
```

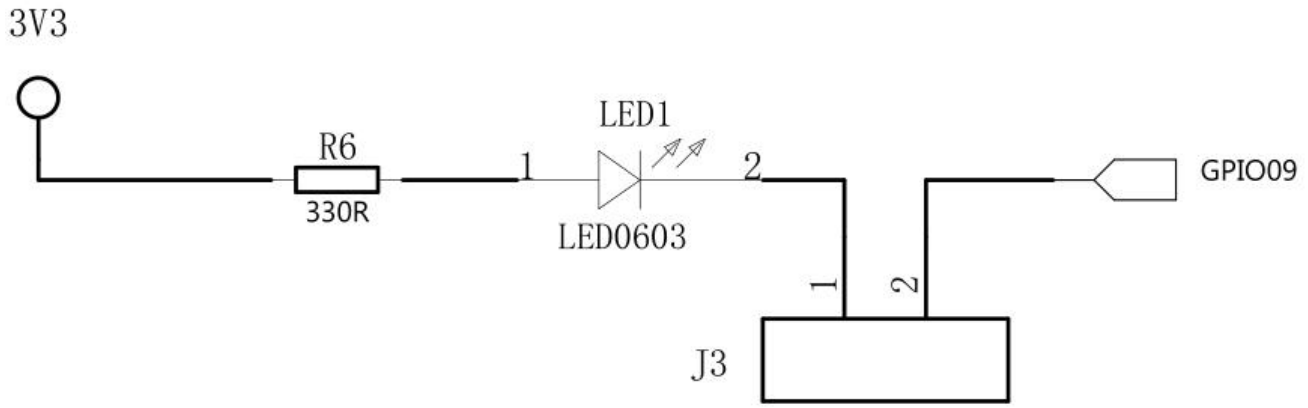
知识点2【代码实现LED灯亮灭】

操作演示

本节小结

本讲所学知识点有：

- 知识点1：电路图分析
- 知识点2：代码实现LED灯亮灭

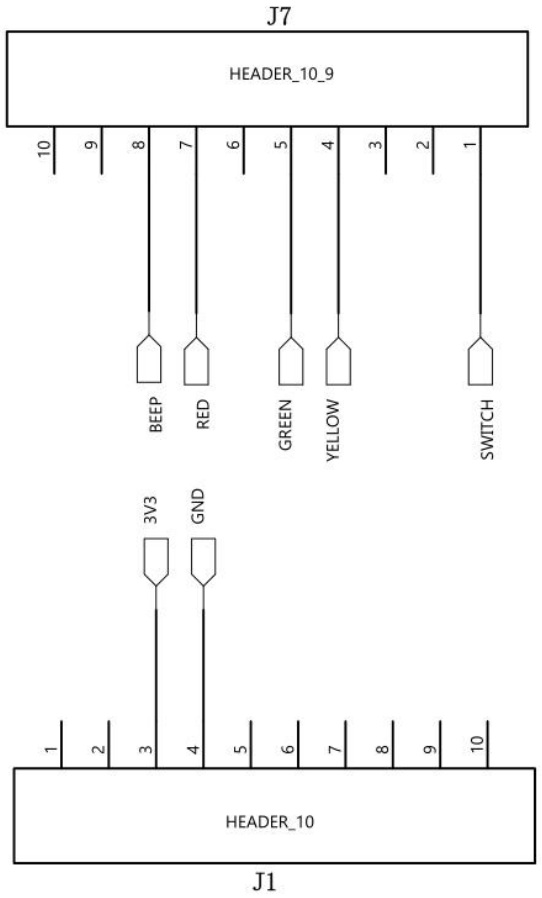


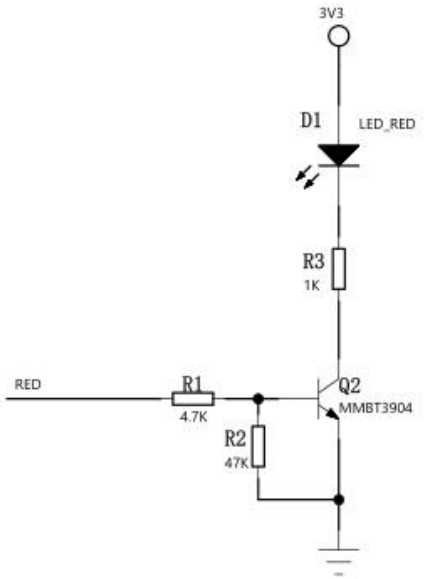
LED指示灯2PIN排针

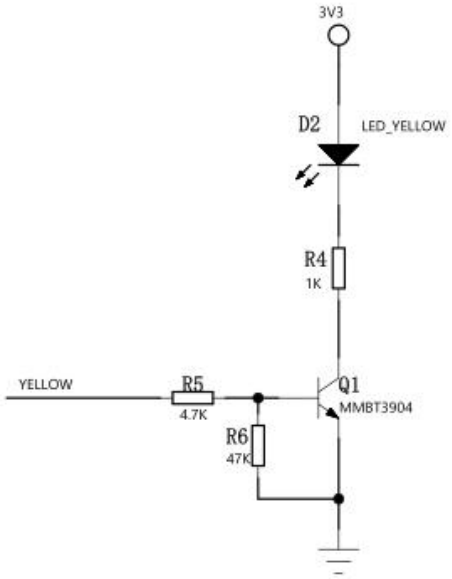
第2节：交通灯功能实现

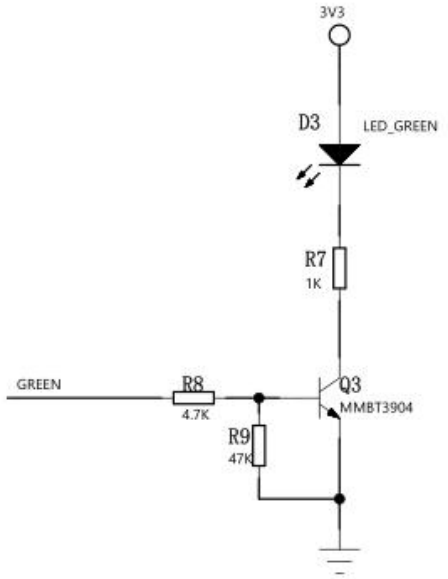
- 知识点1：电路图分析
- 知识点2：代码实现交通灯功能

知识点1【电路图分析】









- 输出高电平点亮红色LED灯
- 输出高电平点亮绿色LED灯
- 输出高电平点亮黄色LED灯

知识点1【电路图分析】

- GPIO_10 连接红色LED，输出高电平点亮红色LED灯
- GPIO_11 连接绿色LED，输出高电平点亮绿色LED灯
- GPIO_12 连接黄色LED，输出高电平点亮黄色LED灯

知识点2【代码实现交通灯功能】

实现步骤：

1. 创建traffic_light_demo.c文件，并初始化
2. 循环点亮绿、黄、红LED灯

知识点2【代码实现LED灯亮灭】

步骤1：创建traffic_light_demo.c文件，并初始化

<code/>

```
#include <stdio.h>
#include <unistd.h>
#include "ohos_init.h"
#include "cmsis_os2.h"
#include "wifiiot_gpio.h"
#include "wifiiot_gpio_ex.h"

void init(void)
{
    GpioInit();
    IoSetFunc(WIFI_IOT_IO_NAME_GPIO_10, WIFI_IOT_IO_FUNC_GPIO_10_GPIO);
    GpioSetDir(WIFI_IOT_IO_NAME_GPIO_10, WIFI_IOT_GPIO_DIR_OUT);

    IoSetFunc(WIFI_IOT_IO_NAME_GPIO_11, WIFI_IOT_IO_FUNC_GPIO_11_GPIO);
    GpioSetDir(WIFI_IOT_IO_NAME_GPIO_11, WIFI_IOT_GPIO_DIR_OUT);

    IoSetFunc(WIFI_IOT_IO_NAME_GPIO_12, WIFI_IOT_IO_FUNC_GPIO_12_GPIO);
    GpioSetDir(WIFI_IOT_IO_NAME_GPIO_12, WIFI_IOT_GPIO_DIR_OUT);
}
```

知识点2【代码实现LED灯亮灭】

步骤2：循环点亮绿、黄、红LED灯

<code/>

```
static void StartTrafficLightTask(void)
{
    init();
    while (1)
    {
        GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_11, WIFI_IOT_GPIO_VALUE1);
        usleep(3000 * 1000);
        GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_11, WIFI_IOT_GPIO_VALUE0);
        GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_12, WIFI_IOT_GPIO_VALUE1);
        usleep(1000 * 1000);
        GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_12, WIFI_IOT_GPIO_VALUE0);
        GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_10, WIFI_IOT_GPIO_VALUE1);
        usleep(3000 * 1000);
        GpioSetOutputVal(WIFI_IOT_IO_NAME_GPIO_10, WIFI_IOT_GPIO_VALUE0);
    }
}

APP_FEATURE_INIT(StartTrafficLightTask);
```

知识点2【代码实现LED灯亮灭】

./applications/sample/wifi-iot/app/chapter_03/BUILD.gn文件

<code/>

```
static_library("chapter_03_demo") {
    sources = [
        # "led_demo.c",
        "traffic_light_demo.c",
    ]

    include_dirs = [
        "//utils/native/lite/include",
        "//kernel/liteos_m/components/cmsis/2.0",
        "//base/iot_hardware/interfaces/kits/wifiiot_lite",
    ]
}
```

知识点2【代码实现LED灯亮灭】

./applications/sample/wifi-iot/app/BUILD.gn文件

<code/>

```
import("//build/lite/config/component/lite_component.gni")

lite_component("app") {
    features = [
        "chapter_03:chapter_03_demo",
    ]
}
```

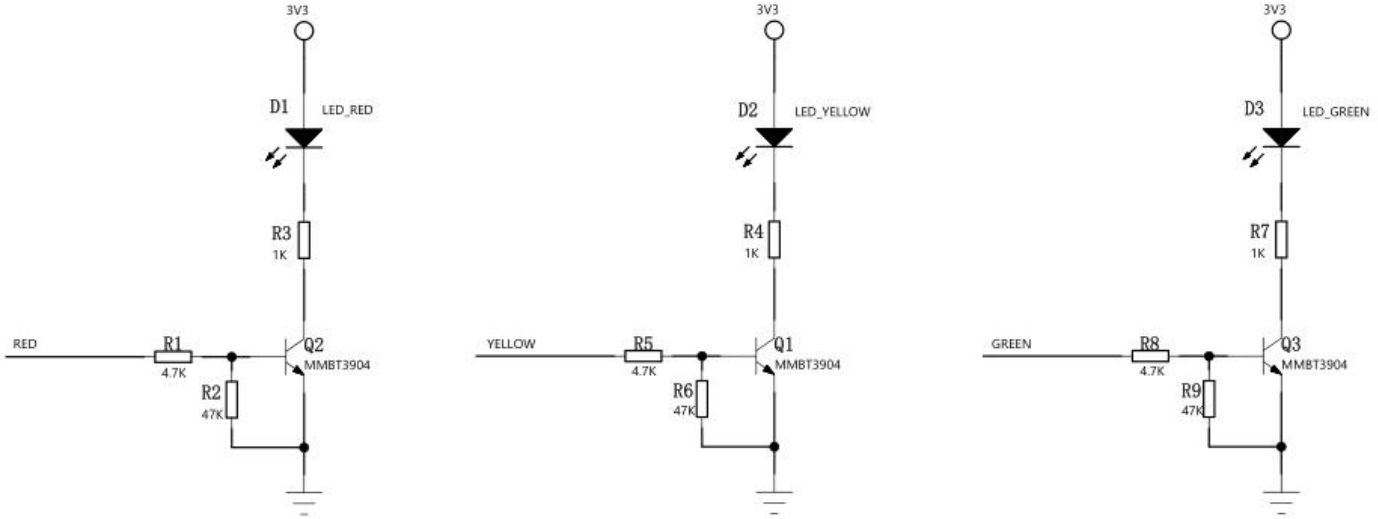
知识点2【代码实现LED灯亮灭】

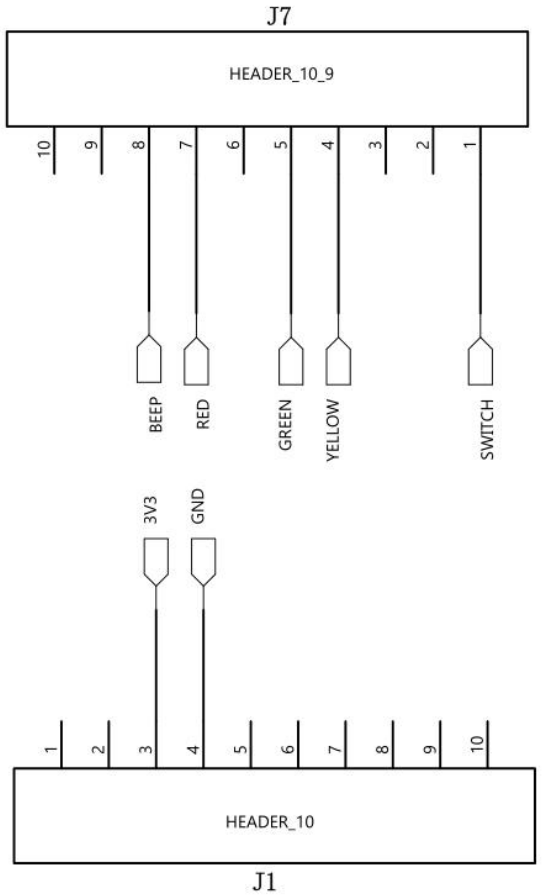
操作演示

本节小结

本讲所学知识点有：

- 知识点1：电路图分析
- 知识点2：代码实现交通灯功能

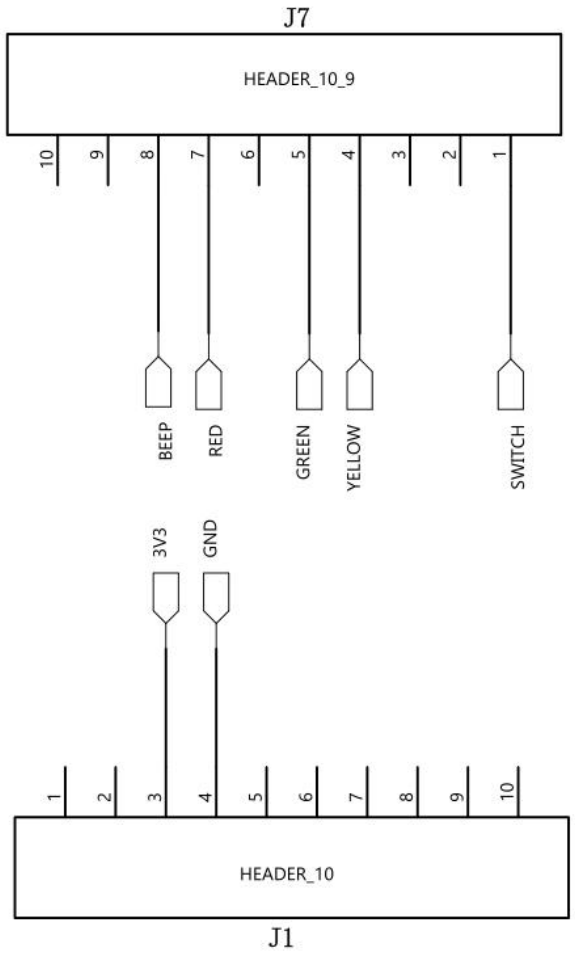


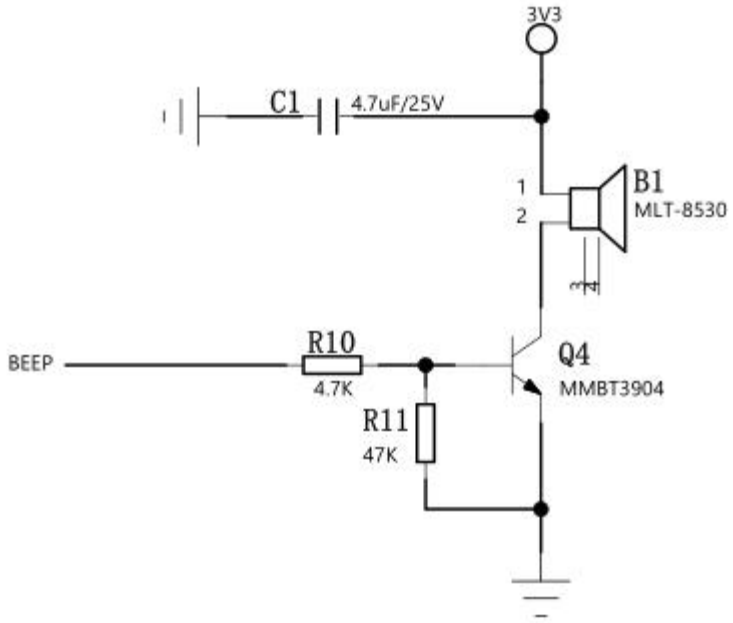


第3节：控制蜂鸣器发出声音

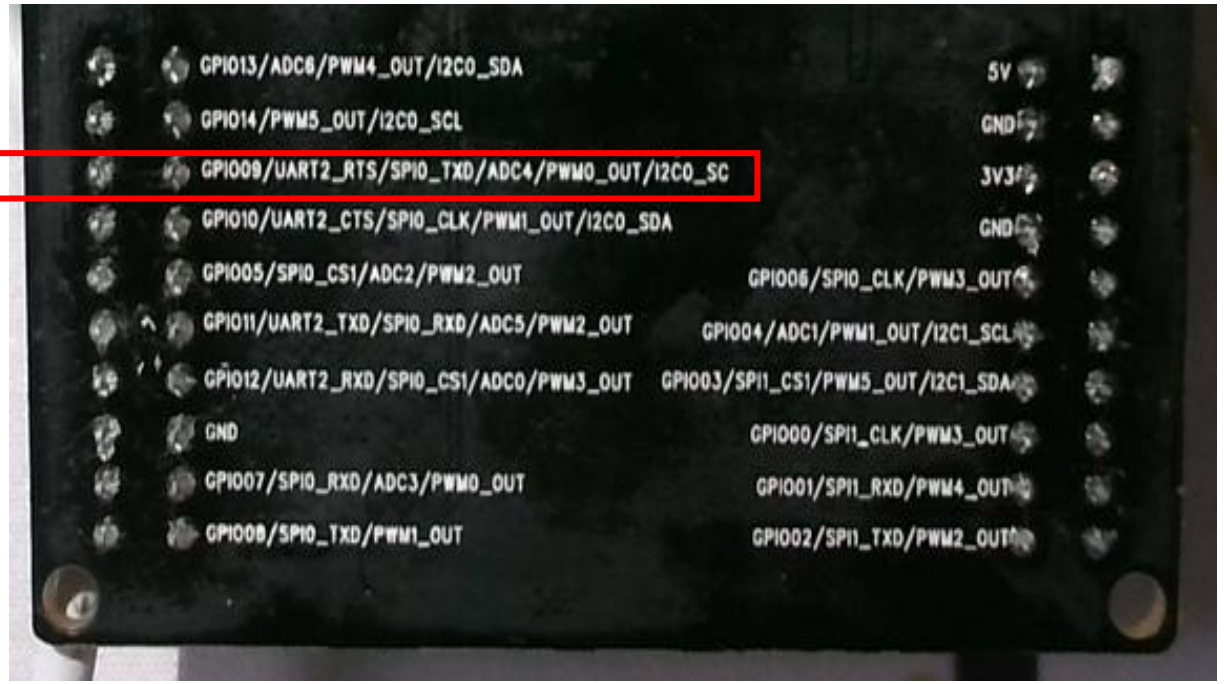
- 知识点1：电路图分析
- 知识点2：输出PWM波控制蜂鸣器发出声音

知识点1【电路图分析】


J7
HEADER_10_9
10
9
8
7
6
5
4
3
2
1
BEEP
RED
GREEN
YELLOW
SWITCH
3V3
GND
1
2
3
4
5
6
7
8
9
10
HEADER_10
J1


3V3
C1
4.7uF/25V
1
2
B1
MLT-8530
BEEP
R10
4.7K
Q4
MMBT3904
R11
47K

知识点1【电路图分析】



- GPIO_09 连接BEEP，

知识点1【电路图分析】

LED指示灯2PIN排针

- GPIO_09 连接BEEP，
- 核心版J3跳线断开连接，否则核心板LED1会有相应操作响应

知识点2【输出PWM波控制蜂鸣器发出声音】

实现步骤：

1. 创建beeper_demo.c文件，并初始化
2. 循环调用PwmStart、PwmStop方法

知识点2【输出PWM波控制蜂鸣器发出声音】

步骤1：创建beeper_demo.c文件，并初始化

<code/>

```
#include <stdio.h>
#include <unistd.h>
#include "ohos_init.h"
#include "cmsis_os2.h"
#include "wifiiot_gpio.h"
#include "wifiiot_gpio_ex.h"
#include "wifiiot_pwm.h"
#include "hi_pwm.h"

void init(void)
{
    GpioInit();

    // 蜂鸣器引脚 设置为 PWM功能
    IoSetFunc(WIFI_IOT_IO_NAME_GPIO_9, WIFI_IOT_IO_FUNC_GPIO_9_PWM0_OUT);
    PwmInit(WIFI_IOT_PWM_PORT_PWM0);
}
```

知识点2【输出PWM波控制蜂鸣器发出声音】

步骤2：循环点亮绿、黄、红LED灯

<code/>

```
void BeeperDemo(void)
{
    init();
    while (1)
    {

        uint16_t freqDivisor = 34052;
        //占空比，频率
        PwmStart(WIFI_IOT_PWM_PORT_PWM0, freqDivisor / 2, freqDivisor);
        usleep(2000000);
        PwmStop(WIFI_IOT_PWM_PORT_PWM0);
        usleep(1000000);
    }
}

SYS_RUN(BeeperDemo);
```

知识点2【输出PWM波控制蜂鸣器发出声音】

./applications/sample/wifi-iot/app/chapter_03/BUILD.gn文件

<code/>

```
static_library("chapter_03_demo") {
    sources = [
        # "led_demo.c",
 #       "beeper_music_demo.c",
        # "traffic_light_demo.c",
          "beeper_demo.c",
    ]

    include_dirs = [
        "//utils/native/lite/include",
        "//kernel/liteos_m/components/cmsis/2.0",
        "//base/iot_hardware/interfaces/kits/wifiiot_lite",
    ]
}
```

知识点2【输出PWM波控制蜂鸣器发出声音】

./applications/sample/wifi-iot/app/BUILD.gn文件

<code/>

```
import("//build/lite/config/component/lite_component.gni")

lite_component("app") {
    features = [
        "chapter_03:chapter_03_demo",
    ]
}
```

知识点2【输出PWM波控制蜂鸣器发出声音】

修改vendor\hisi\hi3861\hi3861\build\config\usr_config.mk文件中的CONFIG_PWM_SUPPORT行：

<code/>

```
# 默认情况下，hi3861_sdk中，PWM的CONFIG选项没有打开
#CONFIG_PWM_SUPPORT is not set
CONFIG_PWM_SUPPORT=y
```

知识点2【输出PWM波控制蜂鸣器发出声音】

操作演示

本节小结

本讲所学知识点有：

- 知识点1：电路图分析
- 知识点2：输出PWM波控制蜂鸣器发出声音

第4节：蜂鸣器实现播放音乐

- 知识点1：案例源码
- 知识点2：案例源码分析
- 知识点3：案例效果演示

知识点1【案例源码】

./applications/sample/wifi-iot/app/chapter_03/beeper_music_demo.c

<code/>

```
#include <stdio.h>
#include <unistd.h>
#include "ohos_init.h"
#include "cmsis_os2.h"
#include "wifiiot_gpio.h"
#include "wifiiot_gpio_ex.h"
#include "wifiiot_watchdog.h"
#include "wifiiot_pwm.h"
#include "hi_pwm.h"
static volatile int g_buttonPressed = 0;
```

知识点1【案例源码】

./applications/sample/wifi-iot/app/chapter_03/beeper_music_demo.c

<code/>

```
static const uint16_t g_tuneFreqs[] = {
    0,     // 40M Hz 对应的分频系数:
    38223, // 1046.5
    34052, // 1174.7
    30338, // 1318.5
    28635, // 1396.9
    25511, // 1568
    22728, // 1760
    20249, // 1975.5
    51021  // 5_ 783.99 // 第一个八度的 5
};
```

知识点1【案例源码】

./applications/sample/wifi-iot/app/chapter_03/beeper_music_demo.c

<code/>

```
// 曲谱音符
static const uint8_t g_scoreNotes[] = {
    // 《两只老虎》简谱：http://www.jianpu.cn/pu/33/33945.htm
    1, 2, 3, 1, 1, 2, 3, 1, 3, 4, 5, 3, 4, 5,
    5, 6, 5, 4, 3, 1, 5, 6, 5, 4, 3, 1, 1, 8, 1, 1, 8, 1, // 最后两个 5 应该是低八度的，链接图片中的曲谱不对，声音到最后听起来不太对劲
};

// 曲谱时值
static const uint8_t g_scoreDurations[] = {
    4,   4,   4, 4,  4,   4,   4, 4,  4, 4, 8,  4, 4, 8,
    3, 1, 3, 1, 4, 4,  3, 1, 3, 1, 4, 4,  4, 4, 8,  4, 4, 8,
};
```

知识点1【案例源码】

./applications/sample/wifi-iot/app/chapter_03/beeper_music_demo.c

<code/>

```
static void *BeeperMusicTask(const char *arg)
{
    (void)arg;
    printf("BeeperMusicTask start!\r\n");
    hi_pwm_set_clock(PWM_CLK_XTAL); // 设置时钟源为晶体时钟（40MHz，默认时钟源160MHz）
    while (1)
    {
        for (size_t i = 0; i < sizeof(g_scoreNotes) / sizeof(g_scoreNotes[0]); i++)
        {
            uint32_t tune = g_scoreNotes[i]; // 音符
            uint16_t freqDivisor = g_tuneFreqs[tune];
            uint32_t tuneInterval = g_scoreDurations[i] * (125 * 1000); // 音符时间
            printf("%d %d %d %d\r\n", tune, (40 * 1000 * 1000) / freqDivisor, freqDivisor, tuneInterval);
            PwmStart(WIFI_IOT_PWM_PORT_PWM0, freqDivisor / 2, freqDivisor);
            usleep(tuneInterval);
            PwmStop(WIFI_IOT_PWM_PORT_PWM0);
        }
    }
    return NULL;
}
```

知识点1【案例源码】

./applications/sample/wifi-iot/app/chapter_03/beeper_music_demo.c

<code/>

```
static void StartBeepMusicTask(void)
{
    osThreadAttr_t attr;
    GpioInit();
    // 蜂鸣器引脚 设置为 PWM功能
    IoSetFunc(WIFI_IOT_IO_NAME_GPIO_9, WIFI_IOT_IO_FUNC_GPIO_9_PWM0_OUT);
    PwmInit(WIFI_IOT_PWM_PORT_PWM0);
    WatchDogDisable();

    attr.name = "BeeperMusicTask";
    attr.attr_bits = 0U;
    attr.cb_mem = NULL;
    attr.cb_size = 0U;
    attr.stack_mem = NULL;
    attr.stack_size = 1024;
    attr.priority = osPriorityNormal;
    if (osThreadNew((osThreadFunc_t)BeeperMusicTask, NULL, &attr) == NULL)
    {
        printf("[LedExample] Falied to create BeeperMusicTask!\n");
    }
}
SYS_RUN(StartBeepMusicTask);
```

知识点1【案例源码】

./applications/sample/wifi-iot/app/chapter_03/BUILD.gn文件

<code/>

```
static_library("chapter_03_demo") {
    sources = [
        # "led_demo.c",
        # "traffic_light_demo.c",
        #   "beeper_demo.c",
       "beeper_music_demo.c",
    ]

    include_dirs = [
        "//utils/native/lite/include",
        "//kernel/liteos_m/components/cmsis/2.0",
        "//base/iot_hardware/interfaces/kits/wifiiot_lite",
    ]
}
```

知识点1【案例源码】

./applications/sample/wifi-iot/app/BUILD.gn文件

<code/>

```
import("//build/lite/config/component/lite_component.gni")

lite_component("app") {
    features = [
        "chapter_03:chapter_03_demo",
    ]
}
```

知识点2【案例源码分析】

知识点2【案例源码分析】

操作演示

知识点2【案例源码分析】

代码分析流程：

1. SYS_RUN(StartBeepMusicTask);
2. static void StartBeepMusicTask(void)
3. static void *BeeperMusicTask(const char *arg)

知识点2【测试验证】

操作演示

本节小结

本讲所学知识点有：

- 知识点1：案例源码
- 知识点2：案例源码分析
- 知识点3：案例效果演示

本章总结

本章所学内容有：

- 第1节：核心板上LED灯开发
- 第2节：交通灯功能实现
- 第3节：控制蜂鸣器发出声音
- 第4节：蜂鸣器实现播放音乐

任务挑战

挑战任务：在插入音乐的案例中添加核心板上的led灯操作，根据音乐节奏控件led的亮与灭

THANKS

更多学习视频，关注宅客学院……