优秀不够，你是否无可替代

导航

博客园
首页
新随笔
联系
订阅 XML
管理

我的标签

8266(88)
MQTT(50)
GPRS(33)
SDK(29)
Air202(28)
云服务器(21)
ESP8266(21)
Lua(18)
小程序(17)
STM32(16)
更多

随笔分类

Android(22)
Android 开发(8)
C# 开发(4)
CH395Q学习开发(17)
CH579M学习开发(7)
ESP32学习开发(10)
ESP8266 AT指令开发(基于STC89C52单片机)(3)
ESP8266 AT指令开发(基于STM32)(1)
ESP8266 AT指令开发基础入门篇备份(12)
ESP8266 LUA脚本语言开发(13)
ESP8266 LUA开发基础入门篇备份(22)

102-ESP32学习开发(SDK)-GPIO

<iframe name="ifd" src="https://mnifdv.cn/resource/cnblogs/LearnESP32" frameborder="0" scrolling="auto" width="100%" height="1500"></iframe>

开源ESP32开发(源码见资料源码)

测试板链接:ESP32测试板链接

资料源码:https://github.com/yangfengwu45/learn-esp32.git

【点击加入乐鑫WiFi模组开发交流群】(群号822685419)https://jq.qq.com/?_wv=1027&k=fXgd3UOo

python虚拟机: python-3.8.4-amd64.exe

ESP-IDF工具安装器: esp-idf-tools-setup-2.3.exe

- 基础开源教程:ESP32开发(arduino)
- 基础开源教程:ESP8266:LUA脚本开发
- 基础开源教程:ESP8266 AT指令开发(基于51单片机)
- 基础开源教程:Android学习开发
- 基础开源教程:C#学习开发
- 基础开源教程:微信小程序开发入门篇
  需要搭配的Android，C#等基础教程如上，各个教程正在整理。

- 000-ESP32开发板使用说明
- ESP32_SDK开发
- 001-开发环境搭建(Windows+VSCode)
- 002-测试网络摄像头(OV2640),实现远程视频监控(花生壳http映射)
- 003-学习ESP32资料说明
- 004-新建工程模板和创建新的文件
- 005-ESP32学习开发(SDK)-新建工程补充-通过官方示例创建工程
- ----------基本外设----------
- 101-ESP32学习开发(SDK)-ESP32管脚说明
-
-
-
-
-
-

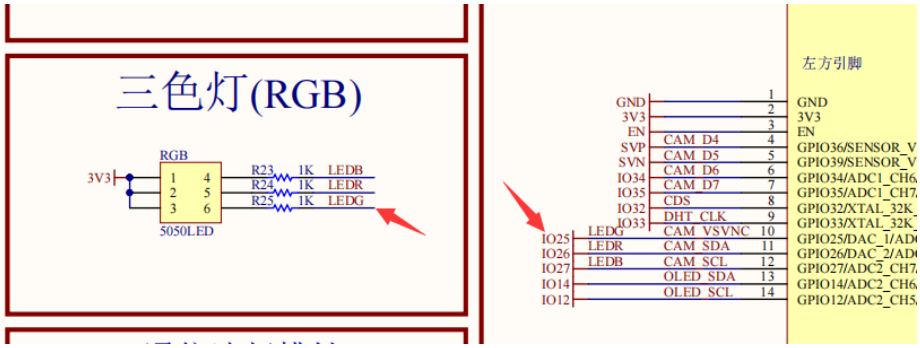
最新评论

1. Re:单片机模块化程序: 看看是不是你想要的按键处理
视频不见了
--伊森亨特
2. Re:C#开发: 通信篇-TCP客户端
感谢分享，直接就用上了
--Zfen

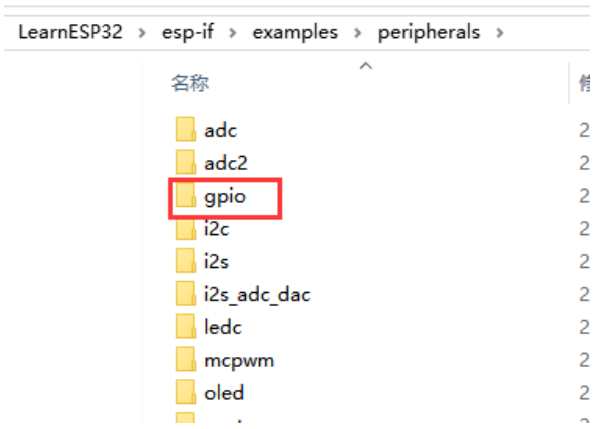
阅读排行榜

控制GPIO25输出高低电平

1.原理图



2.参考官方例程



6. (一)基于阿里云的MQTT远程控制(Android 连接MQTT服务器,ESP8266连接MQTT服务器实现远程通信控制----简单的连接通信)(35888)
7. 关于TCP和MQTT之间的转换(33124)
8. C#中public与private与static(32280)
9. android 之TCP客户端编程(31854)
10. android客服端+eps8266+单片机+路由器之远程控制系统(31298)

推荐排行榜

1. C#委托+回调详解(9)
2. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(8)
3. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
4. ESP8266使用详解(AT,LUA,SDK)(6)
5. 关于TCP和MQTT之间的转换(5)

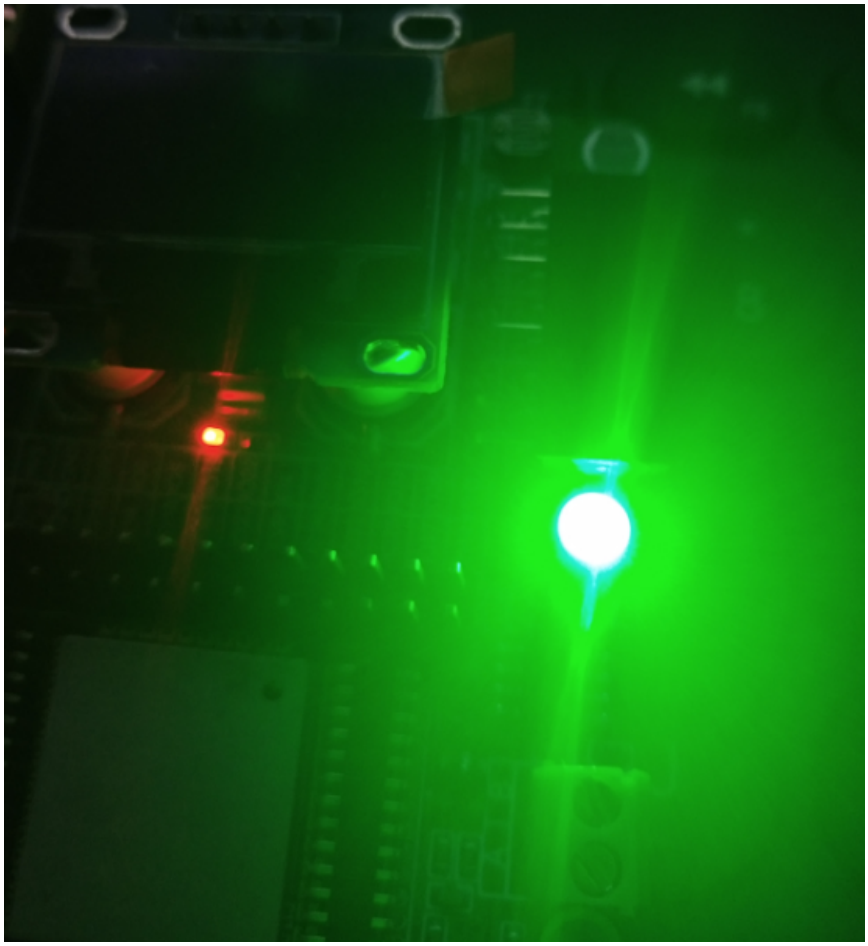
3.程序

main > C gpio_example_main.c > ...

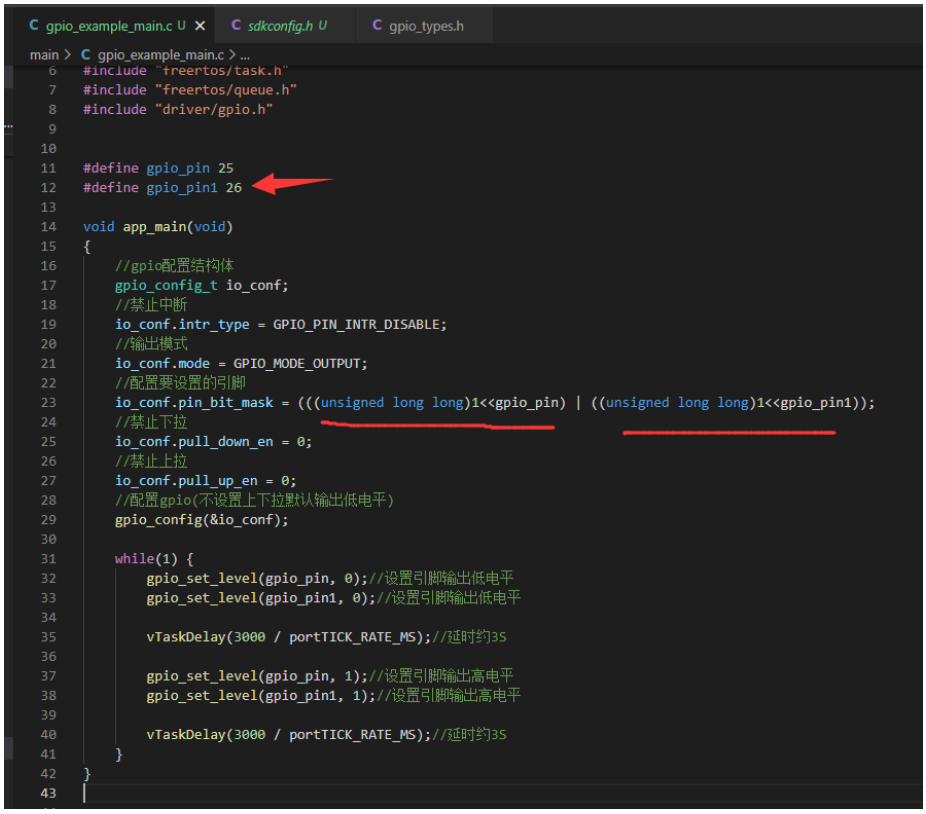
```c

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"


#define gpio_pin 25

void app_main(void)
{
    //gpio配置结构体
    gpio_config_t io_conf;
    //禁止中断
    io_conf.intr_type = GPIO_PIN_INTR_DISABLE;
    //输出模式
    io_conf.mode = GPIO_MODE_OUTPUT;
    //配置要设置的引脚
    io_conf.pin_bit_mask = (unsigned long long)1<<gpio_pin;
    //禁止下拉
    io_conf.pull_down_en = 0;
    //禁止上拉
    io_conf.pull_up_en = 0;
    //配置gpio(不设置上下拉默认输出低电平)
    gpio_config(&io_conf);

    while(1) {
        gpio_set_level(gpio_pin, 0);//设置引脚输出低电平
        vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S
        gpio_set_level(gpio_pin, 1);//设置引脚输出高电平
        vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S
    }
}

```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"


#define gpio_pin 25

void app_main(void)
{
    //gpio配置结构体
    gpio_config_t io_conf;
    //禁止中断
    io_conf.intr_type = GPIO_PIN_INTR_DISABLE;
    //输出模式
    io_conf.mode = GPIO_MODE_OUTPUT;
    //配置要设置的引脚
    io_conf.pin_bit_mask = (unsigned long long)1<<gpio_pin;
    //禁止下拉
    io_conf.pull_down_en = 0;
    //禁止上拉
    io_conf.pull_up_en = 0;
```

```
        //配置gpio(不设置上下拉默认输出低电平)
        gpio_config(&io_conf);

        while(1) {
            gpio_set_level(gpio_pin, 0);//设置引脚输出低电平
            vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S
            gpio_set_level(gpio_pin, 1);//设置引脚输出高电平
            vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S
        }
    }
```

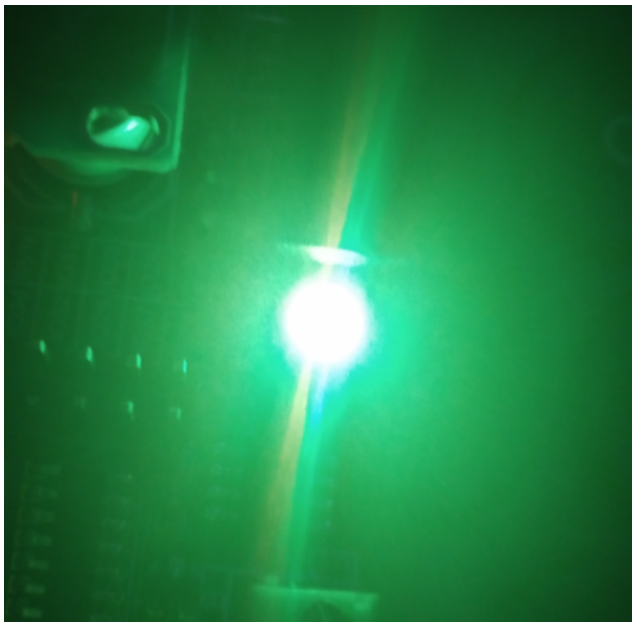


控制GPIO25 和 GPIO26 输出高低电平




```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"


#define gpio_pin 25
#define gpio_pin1 26

void app_main(void)
{
    //gpio配置结构体
    gpio_config_t io_conf;
    //禁止中断
    io_conf.intr_type = GPIO_PIN_INTR_DISABLE;
    //输出模式
    io_conf.mode = GPIO_MODE_OUTPUT;
    //配置要设置的引脚
    io_conf.pin_bit_mask = (((unsigned long long)1<<gpio_pin) | ((unsigned lo
    //禁止下拉
    io_conf.pull_down_en = 0;
    //禁止上拉
    io_conf.pull_up_en = 0;
    //配置gpio(不设置上下拉默认输出低电平)
    gpio_config(&io_conf);

    while(1) {
        gpio_set_level(gpio_pin, 0);//设置引脚输出低电平
        gpio_set_level(gpio_pin1, 0);//设置引脚输出低电平
```

```
            vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S

            gpio_set_level(gpio_pin, 1);//设置引脚输出高电平
            gpio_set_level(gpio_pin1, 1);//设置引脚输出高电平

            vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S
        }
    }
```



补充:

配置gpio还有一个参数 driver

GPIO_DRIVE_CAP_0 弱 weak
GPIO_DRIVE_CAP_1 强
GPIO_DRIVE_CAP_2 默认值
GPIO_DRIVE_CAP_DEFAULT 默认值
GPIO_DRIVE_CAP_3 最强

io_conf.driver = GPIO_DRIVE_CAP_3;

提示

GPIO的模式

GPIO_MODE_INPUT 输入
GPIO_MODE_OUTPUT 输出
GPIO_MODE_OUTPUT_OD 开漏输出

GPIO_MODE_INPUT_OUTPUT_OD 开漏输入输出
GPIO_MODE_INPUT_OUTPUT 输入输出(如果想让模块即做输入检测又做输出控制,需要设置这个模式)

```
    GPIO_INTR_MAX,
} gpio_int_type_t;

typedef enum {
    GPIO_MODE_DISABLE = GPIO_MODE_DEF_DISABLE,
    GPIO_MODE_INPUT = GPIO_MODE_DEF_INPUT,
    GPIO_MODE_OUTPUT = GPIO_MODE_DEF_OUTPUT,
    GPIO_MODE_OUTPUT_OD = ((GPIO_MODE_DEF_OUTPUT) | (GPIO_MODE_DEF_OD)),
    GPIO_MODE_INPUT_OUTPUT_OD = ((GPIO_MODE_DEF_INPUT) | (GPIO_MODE_DEF_OUTPUT) | (GPIO_MODE_DEF_OD)),
    GPIO_MODE_INPUT_OUTPUT = ((GPIO_MODE_DEF_INPUT) | (GPIO_MODE_DEF_OUTPUT)),
} gpio_mode_t;

typedef enum {
    GPIO_PULLUP_DISABLE = 0x0     /*!< Disable GPIO pull-up resistor */
```

配置GPIO0作为输入输出模式,检测引脚输出状态

gpio_example_main.c U | gpio.c | gpio_types.h

main > gpio_example_main.c > app_main(void)

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"


#define gpio_pin 0

void app_main(void)
{
    //gpio配置结构体
    gpio_config_t io_conf;
    //禁止中断
    io_conf.intr_type = GPIO_PIN_INTR_DISABLE;
    //输入输出模式
    io_conf.mode = GPIO_MODE_INPUT_OUTPUT;
    //配置要设置的引脚
    io_conf.pin_bit_mask = (unsigned long long)1<<gpio_pin;
    //禁止下拉
    io_conf.pull_down_en = 0;
    //禁止上拉
    io_conf.pull_up_en = 0;
    //配置gpio(不设置上下拉默认输出低电平)
    gpio_config(&io_conf);

    while(1) {
        gpio_set_level(gpio_pin, 0);//设置引脚输出低电平

        printf("获取引脚状态=%d\r\n",gpio_get_level(gpio_pin));


        vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S

        gpio_set_level(gpio_pin, 1);//设置引脚输出高电平
        printf("获取引脚状态=%d\r\n",gpio_get_level(gpio_pin));

        vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S
    }
}
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"


#define gpio_pin 0

void app_main(void)
{
    //gpio配置结构体
    gpio_config_t io_conf;
    //禁止中断
    io_conf.intr_type = GPIO_PIN_INTR_DISABLE;
    //输入输出模式
    io_conf.mode = GPIO_MODE_INPUT_OUTPUT;
    //配置要设置的引脚
    io_conf.pin_bit_mask = (unsigned long long)1<<gpio_pin;
    //禁止下拉
    io_conf.pull_down_en = 0;
    //禁止上拉
    io_conf.pull_up_en = 0;
    //配置gpio(不设置上下拉默认输出低电平)
    gpio_config(&io_conf);

    while(1) {
        gpio_set_level(gpio_pin, 0);//设置引脚输出低电平

        printf("获取引脚状态=%d\r\n",gpio_get_level(gpio_pin));


        vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S

        gpio_set_level(gpio_pin, 1);//设置引脚输出高电平
        printf("获取引脚状态=%d\r\n",gpio_get_level(gpio_pin));

        vTaskDelay(3000 / portTICK_RATE_MS);//延时约3S
    }
}
```

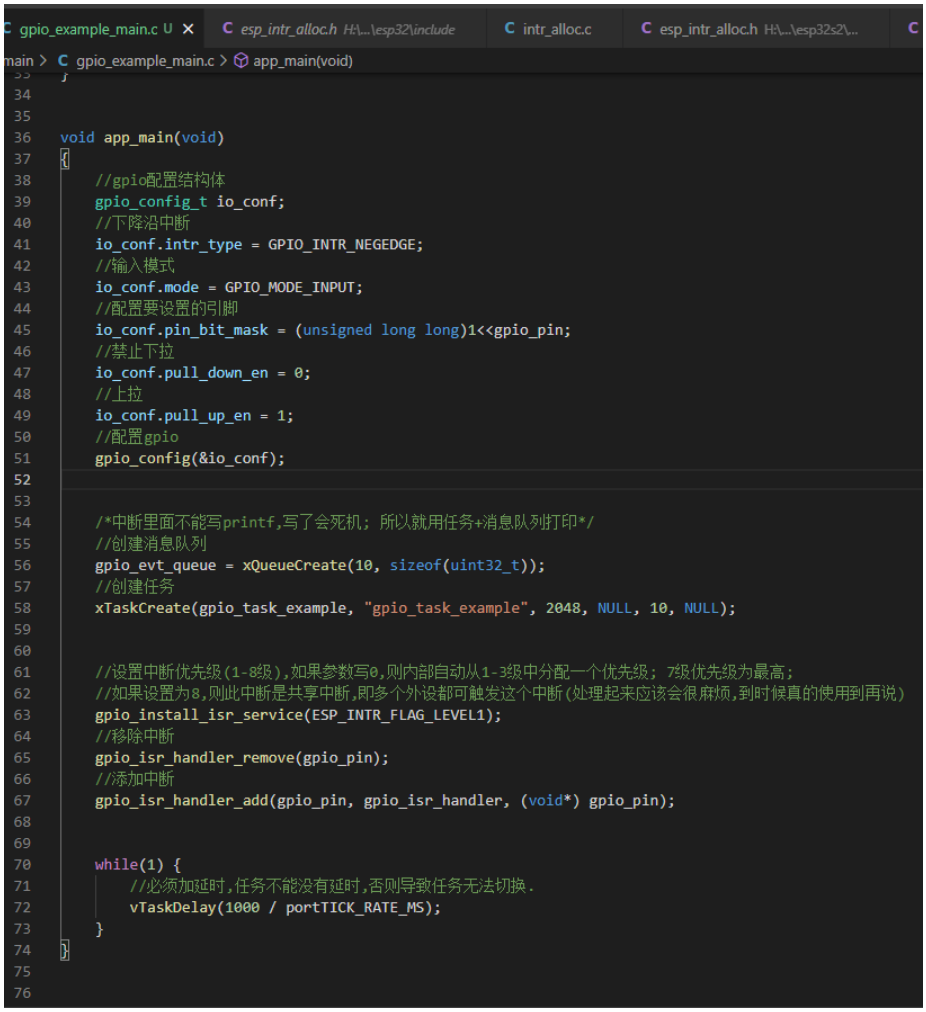
配置GPIO0下降沿中断

1,中断类型

GPIO_INTR_DISABLE 禁用GPIO中断
GPIO_INTR_POSEDGE GPIO中断类型：上升沿
GPIO_INTR_NEGEDGE 下降沿
GPIO_INTR_ANYEDGE 上升沿和下降沿
GPIO_INTR_LOW_LEVEL 输入低电平触发
GPIO_INTR_HIGH_LEVEL 输入高电平触发

2,程序


gpio_example_main.c U
esp_intr_alloc.h H:\...\esp32\include
intr_alloc.c
esp_intr_alloc.h
main > gpio_example_main.c > app_main(void)
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"
#define gpio_pin 0
static xQueueHandle gpio_evt_queue = NULL;
/*gpio中断回调函数*/
static void IRAM_ATTR gpio_isr_handler(void* arg)
{
uint32_t gpio_num = (uint32_t) arg;
//把消息存储到消息队列
xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);
}
/*任务函数*/
static void gpio_task_example(void* arg)
{
uint32_t io_num;
for(;;) {
/*消息队列里面有消息*/
if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY)) {
printf("GPIO[%d] intr, val: %d\n", io_num, gpio_get_level(io_num));
}
}
}
void app_main(void)
{
//gpio配置结构体
gpio_config_t io_conf;
//下降沿中断
io_conf.intr_type = GPIO_INTR_NEGEDGE;
//输入模式
io_conf.mode = GPIO_MODE_INPUT;
//配置要设置的引脚




```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"


#define gpio_pin 0

static xQueueHandle gpio_evt_queue = NULL;

/*gpio中断回调函数*/
static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    uint32_t gpio_num = (uint32_t) arg;
    //把消息存储到消息队列
    xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);
}

/*任务函数*/
static void gpio_task_example(void* arg)
{
    uint32_t io_num;
    for(;;) {
        /*消息队列里面有消息*/
```

```
            if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY)) {
                printf("GPIO[%d] intr, val: %d\n", io_num, gpio_get_level(io_num)
            }
        }
    }


    void app_main(void)
    {
        //gpio配置结构体
        gpio_config_t io_conf;
        //下降沿中断
        io_conf.intr_type = GPIO_INTR_NEGEDGE;
        //输入模式
        io_conf.mode = GPIO_MODE_INPUT;
        //配置要设置的引脚
        io_conf.pin_bit_mask = (unsigned long long)1<<gpio_pin;
        //禁止下拉
        io_conf.pull_down_en = 0;
        //上拉
        io_conf.pull_up_en = 1;
        //配置gpio
        gpio_config(&io_conf);


        /*中断里面不能写printf,写了会死机; 所以就用任务+消息队列打印*/
        //创建消息队列
        gpio_evt_queue = xQueueCreate(10, sizeof(uint32_t));
        //创建任务
        xTaskCreate(gpio_task_example, "gpio_task_example", 2048, NULL, 10, NULL)


        //设置中断优先级(1-8级),如果参数写0,则内部自动从1-3级中分配一个优先级; 7级优先级为最
        //如果设置为8,则此中断是共享中断,即多个外设都可触发这个中断(处理起来应该会很麻烦,到时候
        gpio_install_isr_service(ESP_INTR_FLAG_LEVEL1);
        //移除中断
        gpio_isr_handler_remove(gpio_pin);
        //添加中断
        gpio_isr_handler_add(gpio_pin, gpio_isr_handler, (void*) gpio_pin);


        while(1) {
            //必须加延时,任务不能没有延时,否则导致任务无法切换.
            vTaskDelay(1000 / portTICK_RATE_MS);
        }
    }
```

3,其实主要的就几句话

```c
void app_main(void)
{
    //gpio配置结构体
    gpio_config_t io_conf;
    //下降沿中断
    io_conf.intr_type = GPIO_INTR_NEGEDGE;
    //输入模式
    io_conf.mode = GPIO_MODE_INPUT;
    //配置要设置的引脚
    io_conf.pin_bit_mask = (unsigned long long)1<<gpio_pin;
    //禁止下拉
    io_conf.pull_down_en = 0;
    //上拉
    io_conf.pull_up_en = 1;
    //配置gpio
    gpio_config(&io_conf);


    /*中断里面不能写printf,写了会死机; 所以就用任务+消息队列打印*/
    //创建消息队列
    gpio_evt_queue = xQueueCreate(10, sizeof(uint32_t));
    //创建任务
    xTaskCreate(gpio_task_example, "gpio_task_example", 2048, NULL, 10, NULL);


    //设置中断优先级(1-8级),如果参数写0,则内部自动从1-3级中分配一个优先级; 7级优先级为最高;
    //如果设置为8,则此中断是共享中断,即多个外设都可触发这个中断(处理起来应该会很麻烦,到时候真的使用到再说)
    gpio_install_isr_service(ESP_INTR_FLAG_LEVEL1);
    //移除中断
    gpio_isr_handler_remove(gpio_pin);
    //添加中断
    gpio_isr_handler_add(gpio_pin, gpio_isr_handler, (void*) gpio_pin);
```

```c
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"
#include "driver/gpio.h"


#define gpio_pin 0

static xQueueHandle gpio_evt_queue = NULL;

/*gpio中断回调函数*/
static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    uint32_t gpio_num = (uint32_t) arg;
    //把消息存储到消息队列
    xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);
}
```

4,动作 按键 (GPIO0)


```
I (307) gpio: GPIO[0]| InputEn: 1| OutputEn: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 1
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 0
GPIO[0] intr, val: 1
GPIO[0] intr, val: 0
GPIO[0] intr, val: 1
GPIO[0] intr, val: 0
```

5,提示

按理说下降沿中断,读取到的引脚电平只有0才对,但是呢之所有1,是因为

```c
#define gpio_pin 0

static xQueueHandle gpio_evt_queue = NULL;

/*gpio中断回调函数*/
static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    uint32_t gpio_num = (uint32_t) arg;
    //把消息存储到消息队列
    xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);
}

/*任务函数*/
static void gpio_task_example(void* arg)
{
    uint32_t io_num;
    for(;;) {
        /*消息队列里面有消息*/
        if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY)) {
            printf("GPIO[%d] intr, val: %d\n", io_num, gpio_get_level(io_num));
        }
    }
}
```

进中断的时候确实是下降沿进来的
但是呢,读取引脚电平是在一个任务中
读取的,这时候读取的时候可能按键已经抖动
成高电平了,所以也可能读取1

分类: ESP32学习开发

好文要顶 关注我 收藏该文

杨奉武
关注 - 1
粉丝 - 629

0 0

« 上一篇： 005-ESP32学习开发(SDK)-新建工程补充-通过官方示例创建工程

posted on 2021-08-02 04:10 杨奉武 阅读(0) 评论(0) 编辑 收藏 举报

刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

支持 Markdown

自动补全

提交评论 退出

[Ctrl+Enter快捷键提交]