

## 优秀不够，你是否无可替代

知识从未如此性感。烂程序员关心的是代码,好程序员关心的是数据结构和它们之间的关系 --QQ群: 607064330 --本人  
QQ:946029359 --淘宝 <https://shop411638453.taobao.com/>

随笔 - 772, 文章 - 0, 评论 - 322, 阅读 - 190万

### 导航

博客园  
首页  
新随笔  
联系  
订阅 8ML  
管理

### 公告

渡我不渡她 -  
Not available  
00:00 / 00:00

- 1 渡我不渡她
- 2 小镇姑娘
- 3 PDD洪荒之力

 加入QQ群

昵称：杨奉武  
园龄：5年11个月  
粉丝：662  
关注：1

### 搜索

 找找看  
 谷歌搜索

### 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

Air724UG学习开发(5)  
Android(22)  
Android 开发(8)  
C# 开发(4)  
CH395Q学习开发(17)  
CH579M物联网开发(7)  
CH579M学习开发(7)  
ESP32学习开发(20)  
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)

## 203-ESP32\_SDK开发-TCP服务器(模组AP热点模式,支持多个客户端连接通信)

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

### 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Git下载链接:<https://github.com/yangfengwu45/learn-esp32.git>

资料源码百度网

盘:<https://pan.baidu.com/s/10SBk0NsvLtJYHpDab9islg> 提取码：25oy

【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](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-新建工程补充-通过官方示例创建工程](#)
- [006-关于操作系统-任务,任务堆栈空间,任务的挂起,恢复,删除](#)
- [007-使用缓存管理传递数据](#)
- -----基本外设-----
- [101-ESP32管脚说明](#)
- [102-GPIO](#)
- [103-硬件定时器timer](#)
- [104-软件定时器esp\\_timer](#)
- [105-uart串口,485通信](#)
- [106-GPI](#)

ESP8266 LUA开发基础入门篇  
备份(22)  
ESP8266 SDK开发(33)  
ESP8266 SDK开发基础入门篇  
备份(30)  
GPRS Air202 LUA开发(11)  
HC32F460(华大单片机)学习开  
发(5)  
NB-IOT Air302 AT指令和LUA  
脚本语言开发(27)  
PLC(三菱PLC)基础入门篇(2)  
STM32+Air724UG(4G模组)  
物联网开发(43)  
STM32+BC26/260Y物联网开  
发(37)  
STM32+CH395Q(以太网)物  
联网开发(24)  
STM32+ESP8266(ZLESP8266/  
物联网开发(1)  
STM32+ESP8266+AIR202/30:  
远程升级方案(16)  
STM32+ESP8266+AIR202/30:  
终端管理方案(6)  
STM32+ESP8266+Air302物  
联网开发(64)  
STM32+W5500+AIR202/302  
基本控制方案(25)  
STM32+W5500+AIR202/302  
远程升级方案(6)  
UCOSii操作系统(1)  
W5500 学习开发(8)  
编程语言C#(11)  
编程语言Lua脚本语言基础入  
门篇(6)  
编程语言Python(1)  
单片机(LPC1778)LPC1778(2)  
单片机(MSP430)开发基础入门  
篇(4)  
单片机(STC89C51)单片机开发  
板学习入门篇(3)  
单片机(STM32)基础入门篇(3)  
单片机(STM32)综合应用系列  
(16)  
电路模块使用说明(12)  
感想(6)  
更多

#### 阅读排行榜

1. ESP8266使用详解(AT,LUA, SDK)(173361)
2. 1-安装MQTT服务器(Windo ws),并连接测试(101609)
3. ESP8266刷AT固件与node mcu固件(65694)
4. 用ESP8266+android,制作 自己的WIFI小车(ESP8266篇) (65627)
5. 有人WIFI模块使用详解(389 15)
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)(36338)
7. 关于TCP和MQTT之间的转 换(34231)
8. C#中public与private与stat ic(33912)
9. android 之TCP客户端编程 (32379)
10. android客服端+eps8266 +单片机+路由器之远程控制系 统(31472)

- [I06-SPI](#)
- [107-flash数据存储nvs](#)
- -----网络通信-----
- [201-softAP模式配置模组发出的热点](#)
- [202-station模式配置模组连接路由器热点](#)
- [203-TCP服务器\(模组AP热点模式,支持多个客户端连接通信\).](#)
- 

## 说明

这节测试一下模组在AP模式下作为TCP服务器, 手机或者电脑连接 模块的无线,然后使用TCP客户端连接通信.

## 官方给的例子路径

2021年10月10日 10:00:00 esp-if	
名称	修改
.git	2021
.github	2021
components	2021
docs	2021
examples	2021
make	2021
tools	2021
.editorconfig	2021
.flake8	2021
.gitignore	2021
.gitlab-ci.yml	2021
.gitmodules	2021
.readthedocs.yml	2021

## 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)

## 最新评论

1. Re:1-ESP8266 SDK开发基础入门篇--开发环境搭建杨老大的帖子精华  
--土疙瘩
2. Re:102-CH579M学习开发-基本外设-串口IN4148是不是接反了呀？  
--freemote

- bluetooth
- build\_system
- common\_components
- cxx
- ethernet
- get-started
- mesh
- peripherals
- protocols
- provisioning
- security
- storage
- system
- wifi
- README.md

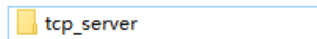
asio	2021/5/2 10
cbor	2021/5/2 10
coap_client	2021/5/2 10
coap_server	2021/5/2 10
esp_http_client	2021/5/2 10
esp_local_ctrl	2021/5/2 10
http_request	2021/5/2 10
http_server	2021/5/2 10
http2_request	2021/5/2 10
https_mbedtls	2021/5/2 10
https_request	2021/5/2 10
https_server	2021/5/2 10
https_x509_bundle	2021/5/2 10
icmp_echo	2021/5/2 10
mdns	2021/5/2 10
modbus	2021/5/2 10
mqtt	2021/5/2 10
openssl_client	2021/5/2 10
openssl_server	2021/5/2 10
pppos_client	2021/5/2 10
smtp_client	2021/5/2 10
sntp	2021/5/2 10
sockets	2021/5/2 10
websocket	2021/5/2 10
README.md	2021/5/2 10

1 2 3

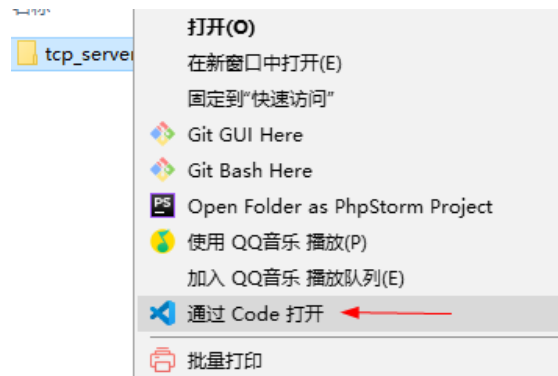
- tcp\_client
- tcp\_client\_multi\_net
- tcp\_server
- udp\_client
- udp\_multicast
- udp\_server
- README.md

# 工程文件测试

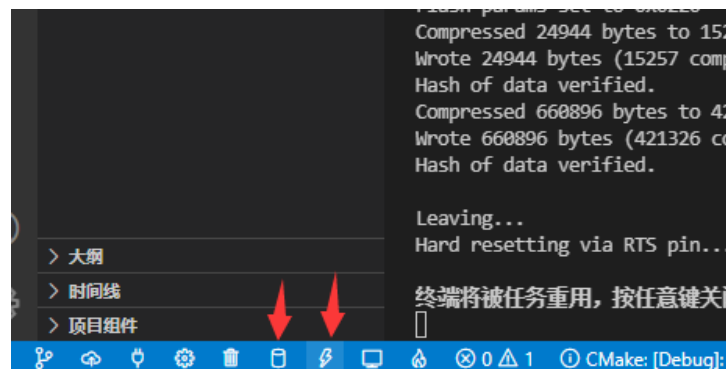
## 1.把这节的代码放到英文目录



## 2.鼠标右键选择使用VScode打开



## 3.编译下载到开发板(第一次编译时间有点长)



## 4.使用手机或者电脑连接名称为ESP32\_WIFI的热点



## 5.打开TCP调试助手,使用TCP客户端连接TCP服务器

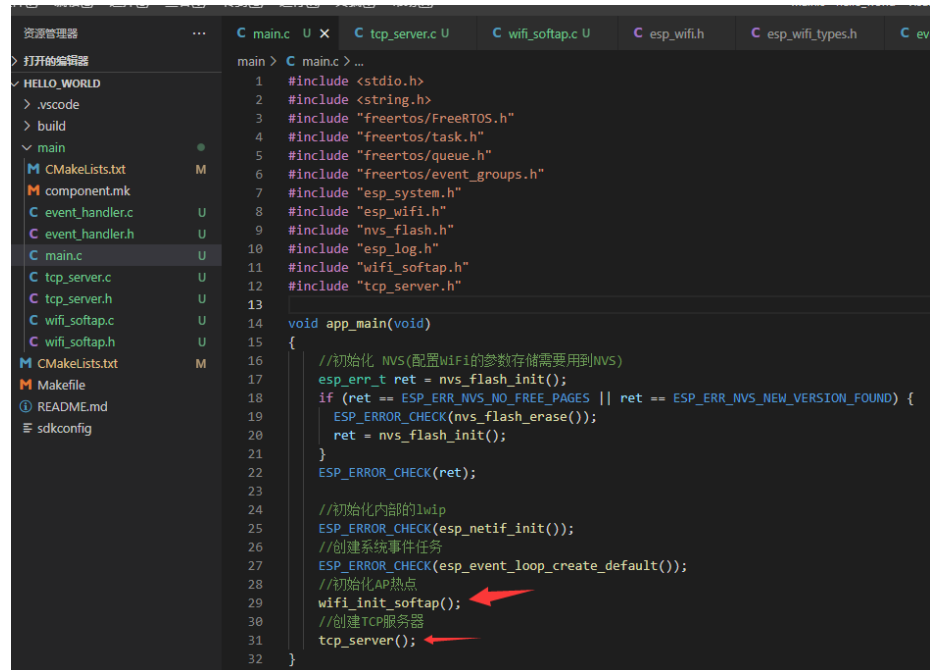
服务器信息为 IP地址:192.168.4.1 端口号:8080

然后发送数据给服务器,就收到服务器返回相同的数据



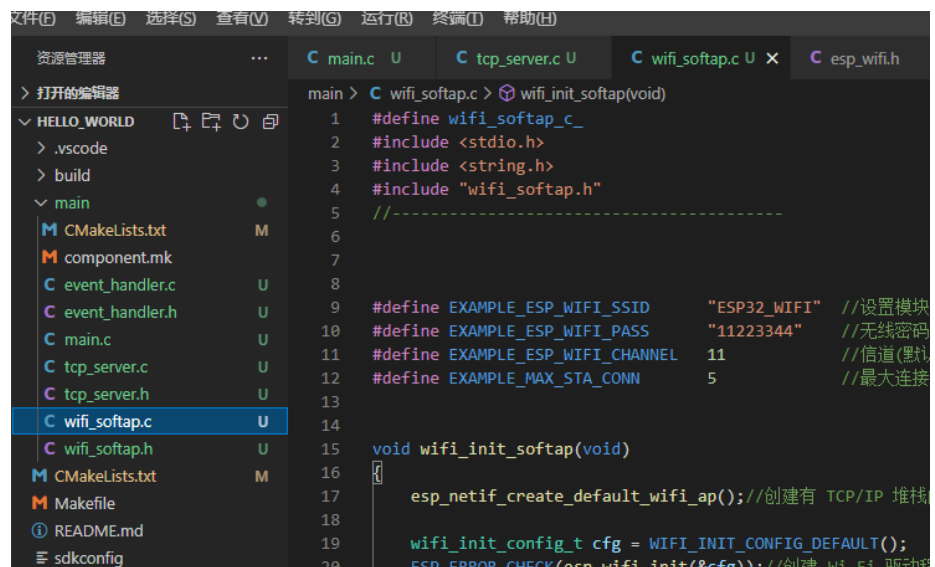
# 工程文件说明(以自己学过的51单片机或者STM32看待文件)

## 1.主函数调用配置热点和创建TCP服务器函数



```
main > C main.c > ...
1 #include <stdio.h>
2 #include <string.h>
3 #include "freertos/FreeRTOS.h"
4 #include "freertos/task.h"
5 #include "freertos/queue.h"
6 #include "freertos/event_groups.h"
7 #include "esp_system.h"
8 #include "esp_wifi.h"
9 #include "nvs_flash.h"
10 #include "esp_log.h"
11 #include "wifi_softap.h"
12 #include "tcp_server.h"
13
14 void app_main(void)
15 {
16     //初始化 NVS(配置WiFi的参数存储需要用到NVS)
17     esp_err_t ret = nvs_flash_init();
18     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
19         ESP_ERROR_CHECK(nvs_flash_erase());
20         ret = nvs_flash_init();
21     }
22     ESP_ERROR_CHECK(ret);
23
24     //初始化内部的lwip
25     ESP_ERROR_CHECK(esp_netif_init());
26     //创建系统事件任务
27     ESP_ERROR_CHECK(esp_event_loop_create_default());
28     //初始化AP热点
29     wifi_init_softap();
30     //创建TCP服务器
31     tcp_server();
32 }
```

## 2.配置热点单独弄了一个文件



```
main > C wifi_softap.c > wifi_init_softap(void)
1 #define wifi_softap_c_
2 #include <stdio.h>
3 #include <string.h>
4 #include "wifi_softap.h"
5 //-----
6
7
8
9 #define EXAMPLE_ESP_WIFI_SSID "ESP32_WIFI" //设置模块
10 #define EXAMPLE_ESP_WIFI_PASS "11223344" //无线密码
11 #define EXAMPLE_ESP_WIFI_CHANNEL 11 //信道(信道)
12 #define EXAMPLE_MAX_STA_CONN 5 //最大连接
13
14
15 void wifi_init_softap(void)
16 {
17     esp_netif_create_default_wifi_ap(); //创建有 TCP/IP 堆栈
18
19     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
20     ESP_ERROR_CHECK(esp_wifi_init(&cfg)); //创建 Wi-Fi 驱动程
```

## 3.可自行配置的热点名称和密码

```
main > C wifi_softap.c > wifi_init_softap(void)
1  #define wifi_softap_c_
2  #include <stdio.h>
3  #include <string.h>
4  #include "wifi_softap.h"
5  //-----
6
7
8
9  #define EXAMPLE_ESP_WIFI_SSID    "ESP32_WIFI"    //设置模块发出的无线名称
10 #define EXAMPLE_ESP_WIFI_PASS    "11223344"      //无线密码
11 #define EXAMPLE_ESP_WIFI_CHANNEL  11             //信道(默认1)
12 #define EXAMPLE_MAX_STA_CONN     5               //最大连接数(最大10个)
13
14
15 void wifi_init_softap(void)
16 {
17     esp_netif_create_default_wifi_ap(); //创建有 TCP/IP 堆栈的默认网络接口实例绑定
18
19     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
20     ESP_ERROR_CHECK(esp_wifi_init(&cfg)); //创建 Wi-Fi 驱动程序任务，并初始化 Wi-Fi
21
22 }
```

## 5.执行创建TCP服务器任务

```
main > C tcp_server.c > tcp_server_task(void *)
114     shutdown(sock, 0);
115     close(sock);
116 }
117
118 CLEAN_UP:
119     close(listen_sock);
120     vTaskDelete(NULL);
121 }
122
123 void tcp_server(void)
124 {
125     #ifdef CONFIG_EXAMPLE_IPV4
126     xTaskCreate(tcp_server_task, "tcp_server", 4096, (void*)AF_INET, 5, NULL);
127     #endif
128     #ifdef CONFIG_EXAMPLE_IPV6
129     xTaskCreate(tcp_server_task, "tcp_server", 4096, (void*)AF_INET6, 5, NULL);
130     #endif
131 }
132
```

## 6.配置TCP服务器参数

```
C tcp_server.c U X C esp_log.h C sockets.h C inet.h C wifi_softap.c U C esp_v...
main > C tcp_server.c > ...
226 static void tcp_server_task(void *pvParameters)
227 {
228     int i;
229     char addr_str[128];
230     int addr_family = (int)pvParameters;
231     int ip_protocol = 0;
232     struct sockaddr_in6 dest_addr;
233     if (addr_family == AF_INET) {
234         struct sockaddr_in *dest_addr_ip4 = (struct sockaddr_in *)&dest_addr;
235         dest_addr_ip4->sin_addr.s_addr = htonl(INADDR_ANY);
236         dest_addr_ip4->sin_family = AF_INET;
237         dest_addr_ip4->sin_port = htons(PORT);
238         ip_protocol = IPPROTO_IP;
239     } else if (addr_family == AF_INET6) {
240         bzero(&dest_addr.sin6_addr.un, sizeof(dest_addr.sin6_addr.un));
241         dest_addr.sin6_family = AF_INET6;
242         dest_addr.sin6_port = htons(PORT);
243         ip_protocol = IPPROTO_IPV6;
244     }
245     int listen_sock = socket(addr_family, SOCK_STREAM, ip_protocol);
246     if (listen_sock < 0) {
247         ESP_LOGE(TAG, "Unable to create socket: errno %d", errno);
248         vTaskDelete(NULL);
249         return;
250     }
251     #if defined(CONFIG_EXAMPLE_IPV4) && defined(CONFIG_EXAMPLE_IPV6)
252     // Note that by default IPV6 binds to both protocols, it is must be disabled
253     // if both protocols used at the same time (used in CI)
254     int opt = 1;
255     setsockopt(listen_sock, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
256     setsockopt(listen_sock, IPPROTO_IPV6, IPV6_V6ONLY, &opt, sizeof(opt));
257     #endif
258     ESP_LOGI(TAG, "Socket created");
259
260     int err = bind(listen_sock, (struct sockaddr *)&dest_addr, sizeof(dest_addr));
261     if (err != 0) {
262         ESP_LOGE(TAG, "Socket unable to bind: errno %d", errno);
263         ESP_LOGE(TAG, "IPPROTO: %d", addr_family);
264         goto CLEAN_UP;
265     }
266     ESP_LOGI(TAG, "Socket bound, port %d", PORT);
267
268     err = listen(listen_sock, 1);
269     if (err != 0) {
270         ESP_LOGE(TAG, "Error occurred during listen: errno %d", errno);
271         goto CLEAN_UP;
272     }
273 }
```

用户如果需要修改端口号,可在头文件修改

```
C tcp_server.h U X
main > C tcp_server.h > ...
17 #include "nvs_flash.h"
18 #include "esp_netif.h"
19
20 #include "lwip/err.h"
21 #include "lwip/sockets.h"
22 #include "lwip/sys.h"
23 #include <lwip/netdb.h>
24
25 #define CONFIG_EXAMPLE_IPV4
26 #define TCP_SERVER_PORT 8080 //服务器监听的端口号
27
28 #define TCP_SERVER_MAX_COUNT 6 //设置最大支持的TCP客户端连接个数
29 #define TCP_SERVER_MIN_PRIORITY 8 //客户端接收数据任务最低优先级(64)
30
```



## 7.初始化配置多客户端连接用到的参数, 有客户端连接之后配置下参数

①:信号量的个数和客户端的个数一样,每次创建一个客户端就取走一个信号量,每关闭一个客户端就回收一个,这样的话限制了客户端的连接个数.

②:事先把每个客户端的接收回调函数写好

③:查看并使用还没有使用的socket, 并执行相应的接收数据回调函数

```
C tcp_server.h U C tcp_server.c U X
main > C tcp_server.c > ...
273 }
274
275 //创建信号量
276 TaskToIrqSemaphoreCounting=xSemaphoreCreateCounting( TCP_SERVER_MAX_COUNT, TCP_SERVER_MAX_COUNT );
277 struct sockaddr_in6 source_addr; // Large enough for both IPv4 or IPv6
278 uint addr_len = sizeof(source_addr);
279
280 for(i=0;i<TCP_SERVER_MAX_COUNT;i++) tcp_server_struct_value[i].state=0;
281
282 /*如果修改了客户端个数,需要修改这个地方,设置回调函数*/
283 tcp_server_struct_value[0].tcp_client_rcv_cb = tcp_client0_rcv;
284 tcp_server_struct_value[1].tcp_client_rcv_cb = tcp_client1_rcv;
285 tcp_server_struct_value[2].tcp_client_rcv_cb = tcp_client2_rcv;
286 tcp_server_struct_value[3].tcp_client_rcv_cb = tcp_client3_rcv;
287 tcp_server_struct_value[4].tcp_client_rcv_cb = tcp_client4_rcv;
288 tcp_server_struct_value[5].tcp_client_rcv_cb = tcp_client5_rcv;
289
290 while (1) {
291
292     //获取一个信号量
293     if(xSemaphoreTake(TaskToIrqSemaphoreCounting,portMAX_DELAY))
294     {
295         ESP_LOGI(TAG, "Socket listening");
296         int sock = accept(listen_sock, (struct sockaddr *)&source_addr, &addr_len); //等待客户端连接
297         if (sock < 0) {
298             ESP_LOGE(TAG, "Unable to accept connection: errno %d", errno);
299             break;
300         }
301         // Convert ip address to string
302         if (source_addr.sin6_family == PF_INET) {
303             inet_ntoa_r(((struct sockaddr_in *)&source_addr)->sin_addr.s_addr, addr_str, sizeof(addr_str) - 1);
304         } else if (source_addr.sin6_family == PF_INET6) {
305             inet6_ntoa_r(source_addr.sin6_addr, addr_str, sizeof(addr_str) - 1);
306         }
307         ESP_LOGI(TAG, "Socket id %d accepted ip address: %s", sock,addr_str);
308
309         for(i=0;i<TCP_SERVER_MAX_COUNT;i++)
310         {
311             if(tcp_server_struct_value[i].state==0)
312             {
313                 tcp_server_struct_value[i].state=1;
314                 tcp_server_struct_value[i].sock_id = sock;
315                 xTaskCreate(tcp_server_struct_value[i].tcp_client_rcv_cb, "tcp_server_client", 4096, (void*)&tcp_server_struct_value[i], 1, &tcp_server_struct_value[i].task_id);
316                 break;
317             }
318         }
319     }
320 }
321 }
```

## 8.接收回调函数除了名字不一样,其余都是一样的,就是接收到什么数据就返回什么数据

```
9
10 //信号量
11 SemaphoreHandle_t TaskToTrqSemaphoreCounting;
12
13
14 void tcp_client0_recv(void *pvParameters)
15 {
16     int len;
17     char rx_buffer[128];
18     int sock=tcp_server_struct_value[0].sock_id;
19     do {
20         len = recv(sock, rx_buffer, sizeof(rx_buffer)-1, 0);
21         if (len < 0) {
22             ESP_LOGE(TAG, "Error occurred during receiving: errno %d", errno);
23         } else if (len == 0) {
24             ESP_LOGW(TAG, "Connection closed");
25         } else {
26             rx_buffer[len] = 0; // Null-terminate whatever is received and treat it like a string
27             ESP_LOGI(TAG, "Received %d bytes: %s", len, rx_buffer);
28
29             // send() can return less bytes than supplied length.
30             // Walk-around for robust implementation.
31             int to_write = len;
32             while (to_write > 0) {
33                 int written = send(sock, rx_buffer + (len - to_write), to_write, 0);
34                 if (written < 0) {
35                     ESP_LOGE(TAG, "Error occurred during sending: errno %d", errno);
36                 }
37                 to_write -= written;
38             }
39         }
40     } while (len > 0);
41
42     tcp_server_struct_value[0].state=0;
43     xSemaphoreGive(TaskToTrqSemaphoreCounting);
44     shutdown(sock, 0);
45     close(sock);
46     vTaskDelete(NULL);
47 }
48
49 void tcp_client1_recv(void *pvParameters)
50 {
51     int len;
52     char rx_buffer[128];
53     int sock=tcp_server_struct_value[1].sock_id;
54     do {
55         len = recv(sock, rx_buffer, sizeof(rx_buffer)-1, 0);
```

## 9.如果想单独处理发送数据,可以参考下面的例子

```
6
7
8 static const char *TAG = "example";
9
10 //信号量
11 SemaphoreHandle_t TaskToTrqSemaphoreCounting;
12
13
14
15 void tcp_server_send(char *data,int len)
16 {
17     int i;
18     for(i=0;i<TCP_SERVER_MAX_COUNT;i++)
19     {
20         if(tcp_server_struct_value[i].state == 1)//该客户端是连接的
21         {
22             /*给对应客户端的socket id 发送数据*/
23             send(tcp_server_struct_value[i].sock_id, data, len, 0);
24         }
25     }
26 }
27
```

[好文要顶](#)[关注我](#)[收藏该文](#)[杨奉武](#)[关注 - 1](#)[粉丝 - 662](#)

1

0

[« 上一篇： 中国移动M5311模块使用手册\(TCP, MQTT\)](#)posted on 2021-10-01 12:11 杨奉武 阅读(19) 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](#)[B](#)

支持 Markdown

自动补全

[提交评论](#)[退出](#)[\[Ctrl+Enter快捷键提交\]](#)**【推荐】** 跨平台组态\工控\仿真\CAD 50万行C++源码全开放免费下载！**【推荐】** 和开发者在一起：华为开发者社区，入驻博客园科技品牌专区**【注册】** 10W+ APP开发者成长平台：流量变现、用户增长、LTV提升！**编辑推荐：**

- .Net 微服务实战之可观测性
- 使用 three.js 实现炫酷的酸性风格 3D 页面
- 一个故事看懂 CPU 的 TLB
- CSS 奇技淫巧 | 妙用混合模式实现文字镂空波浪效果
- 记一次 .NET 某上市工业智造 CPU+内存+挂死 三高分析

**最新新闻：**

- Google涂鸦纪念奇卡诺拳击手、活动家Rodolfo 'Corky' Gonzales ( 2021-10-01 21:57 )
  - 恒大汽车被曝员工半薪轮休：都在刷简历 ( 2021-10-01 21:50 )
  - Google修补两个Chrome 0day危急漏洞 用户应尽快更新 ( 2021-10-01 21:45 )
  - 一颗巨大的彗星最初被误认为是一颗矮行星 ( 2021-10-01 21:38 )
  - 全球最大AI巨量模型，参数2457亿炼丹16天专注中文 ( 2021-10-01 21:30 )
- [» 更多新闻...](#)

历史上的今天：

2020-10-01 硬件基础知识和典型应用-关于485方式通信

Powered by:

博客园

Copyright © 2021 杨奉武

Powered by .NET 6 on Kubernetes



单片机,物联网,上位机,...

扫一扫二维码，入群聊。