

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 - 808, 文章 - 0, 评论 - 327, 阅读 - 201万

导航

[博客园](#)
[首页](#)
[新随笔](#)
[联系](#)
[订阅](#)
[管理](#)

公告

渡我不渡她 -
Not available

00:00 / 03:41

- 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：6年2个月
 粉丝：693
 关注：1

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\(17\)](#)
[CH573F学习开发\(1\)](#)
[CH579M物联网开发\(12\)](#)
[CH579M学习开发\(8\)](#)
[ESP32学习开发\(22\)](#)
[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\(13\)](#)

205-ESP32_SDK开发-TCP服务器(select方式,支持多连接,高速高并发传输)

<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-esp>

资料源码百度网盘:<https://pan.baidu.com/s/10SBk0NsvLtJYHpDab>

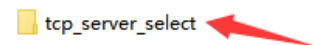
【点击加入乐鑫WiFi模组开发交流群】(群号822685419)<https://jq.qq.com/>

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-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中

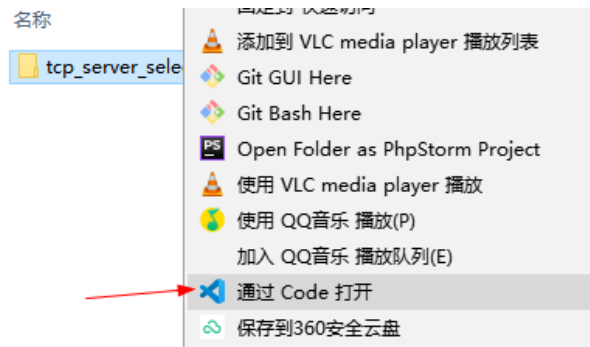


1. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(9)
2. C#委托+回调详解(9)
3. 用ESP8266+android,制作自己的WIFI小车(Android 软件)(6)
4. 我的大学四年(6)
5. ESP8266使用详解(AT,LUA, SDK)(6)

最新评论

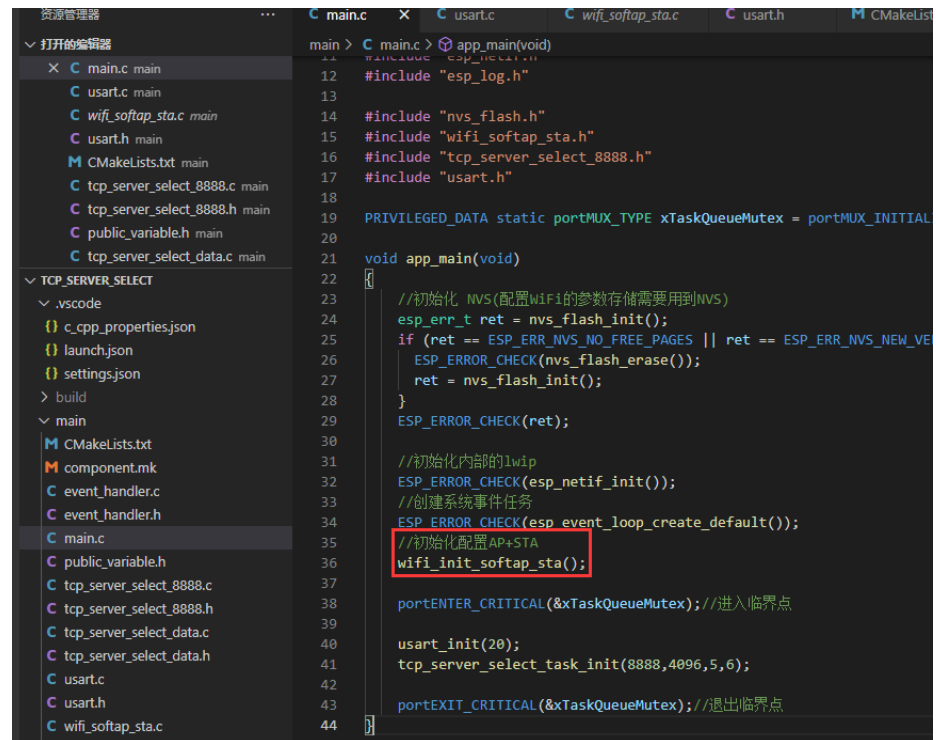
1. Re:2-6-1-视频传输,监控,直播方案-手机连接ESP32的热点,使用微信小程序查看摄像头图像(WiFi视频小车,局域网视频监控)
赞赞赞,感谢大佬无私奉献
--SJA2C2A
2. Re:中移动M5311模块使用手册(TCP,MQTT)
请问你用的usb转ttl是哪一种呢,我用的ch340可是开机串口助手没有SIM识别显示
--夏洛的网娅

2.鼠标右键选择使用VScode打开

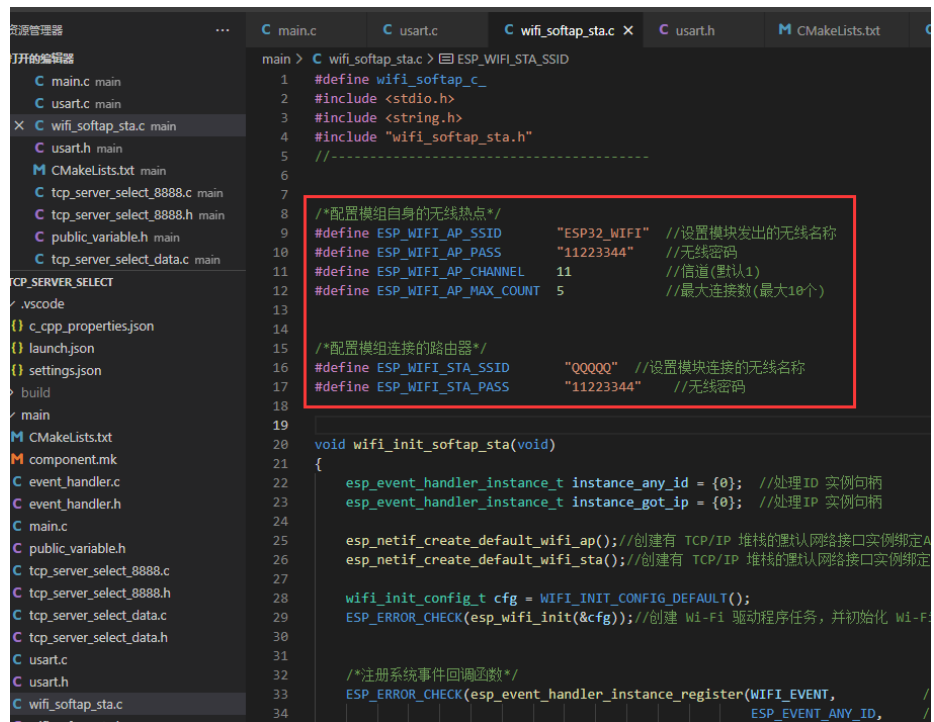


3.关于部分配置

用户进到此函数文件里面可以配置模块热点名称和模块连接的路由器信息

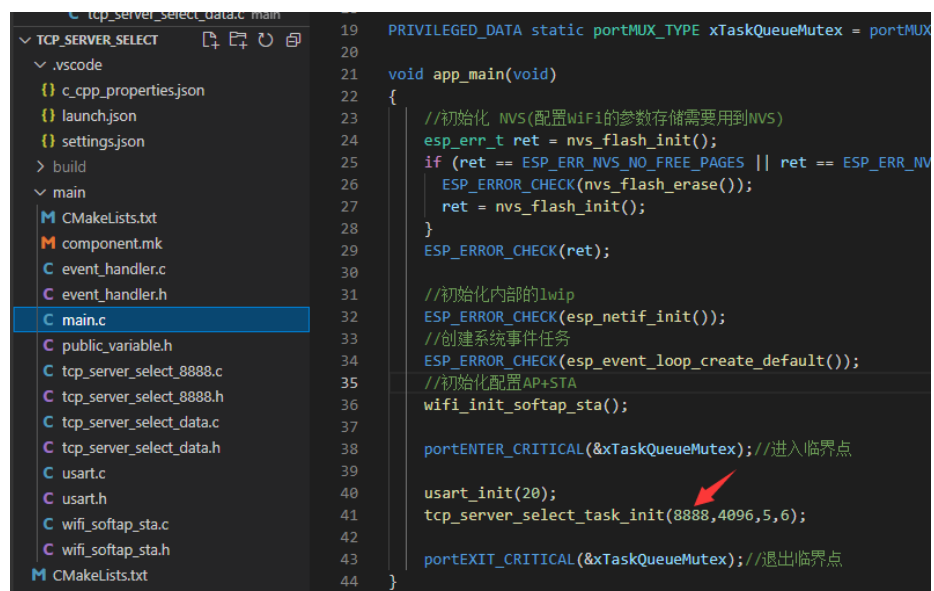


如果不需要连接路由器也不需要修改,顶多是内部连接不上而已.



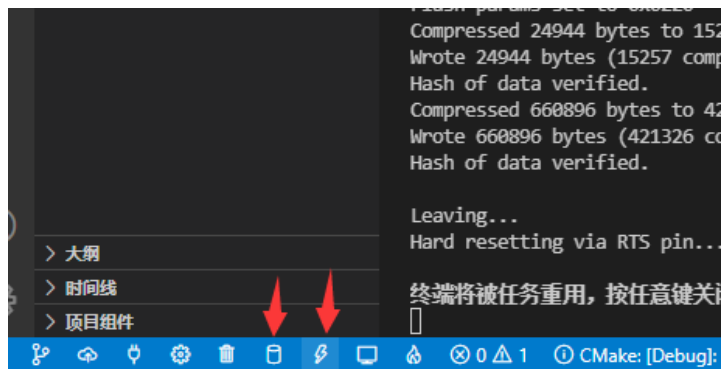
```
main > C wifi_softap_sta.c > ESP_WIFI_STA_SSID
1 #define wifi_softap_c_
2 #include <stdio.h>
3 #include <string.h>
4 #include "wifi_softap_sta.h"
5 //-----
6
7
8 /*配置模组自身的无线热点*/
9 #define ESP_WIFI_AP_SSID "ESP32_WIFI" //设置模块发出的无线名称
10 #define ESP_WIFI_AP_PASS "11223344" //无线密码
11 #define ESP_WIFI_AP_CHANNEL 11 //信道(默认1)
12 #define ESP_WIFI_AP_MAX_COUNT 5 //最大连接数(最大10个)
13
14
15 /*配置模组连接的路由器*/
16 #define ESP_WIFI_STA_SSID "000000" //设置模块连接的无线名称
17 #define ESP_WIFI_STA_PASS "11223344" //无线密码
18
19
20 void wifi_init_softap_sta(void)
21 {
22     esp_event_handler_instance_t instance_any_id = {0}; //处理ID 实例句柄
23     esp_event_handler_instance_t instance_got_ip = {0}; //处理IP 实例句柄
24
25     esp_netif_create_default_wifi_ap(); //创建 TCP/IP 堆栈的默认网络接口实例绑定
26     esp_netif_create_default_wifi_sta(); //创建 TCP/IP 堆栈的默认网络接口实例绑定
27
28     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
29     ESP_ERROR_CHECK(esp_wifi_init(&cfg)); //创建 Wi-Fi 驱动程序任务，并初始化 Wi-Fi
30
31     /*注册系统事件回调函数*/
32     ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT, //
33                                                         ESP_EVENT_ANY_ID, //
```

用户可以在这里设置TCP服务器监听的端口号: 现在监听的是8888



```
19 PRIVILEGED_DATA static portMUX_TYPE xTaskQueueMutex = portMUX
20
21 void app_main(void)
22 {
23     //初始化 NVS(配置WiFi的参数存储需要用到NVS)
24     esp_err_t ret = nvs_flash_init();
25     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS
26         ESP_ERROR_CHECK(nvs_flash_erase());
27     ret = nvs_flash_init();
28 }
29 ESP_ERROR_CHECK(ret);
30
31 //初始化内部的lwip
32 ESP_ERROR_CHECK(esp_netif_init());
33 //创建系统事件任务
34 ESP_ERROR_CHECK(esp_event_loop_create_default());
35 //初始化配置AP+STA
36 wifi_init_softap_sta();
37
38 portENTER_CRITICAL(&xTaskQueueMutex); //进入临界点
39
40 usart_init(20);
41 tcp_server_select_task_init(8888, 4096, 5, 6);
42
43 portEXIT_CRITICAL(&xTaskQueueMutex); //退出临界点
44 }
```

4.编译下载到开发板(第一次编译时间有点长)



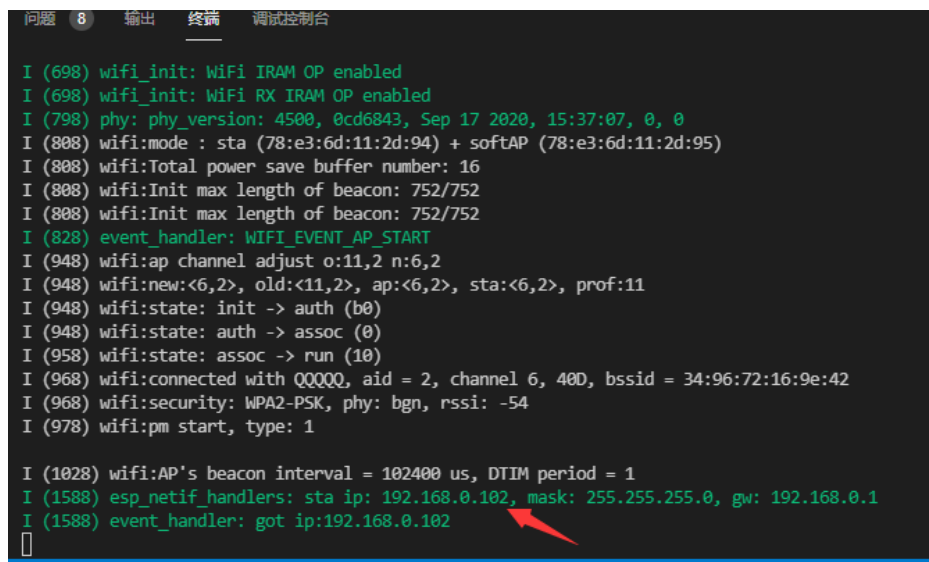
测试

1.程序下载以后会有个名称为 ESP32_WIFI 的热点

选取网络



2.如果让模块连接了路由器,日志里面也会打印连接路由器之后的信息



3.提示

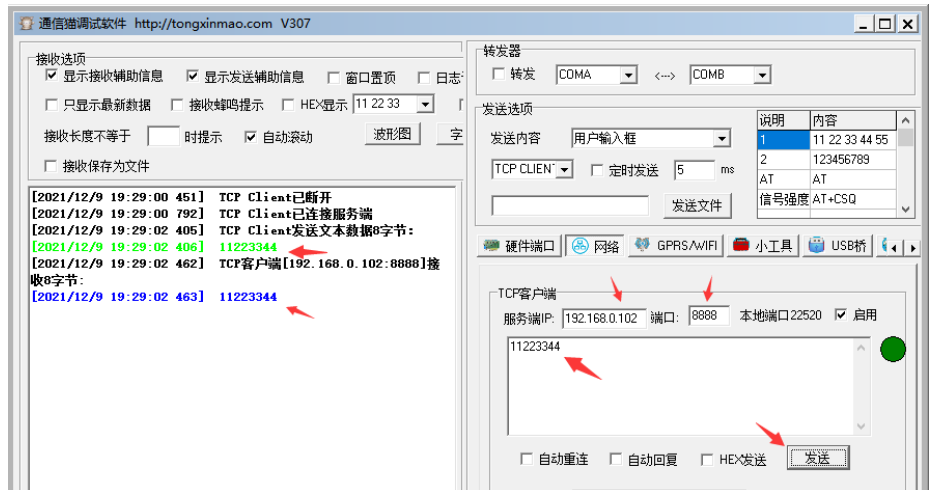
如果大家伙使用手机或者电脑连接模组的热点进行测试,

那么模组的TCP服务器的IP地址是:192.168.4.1 端口号是:8888

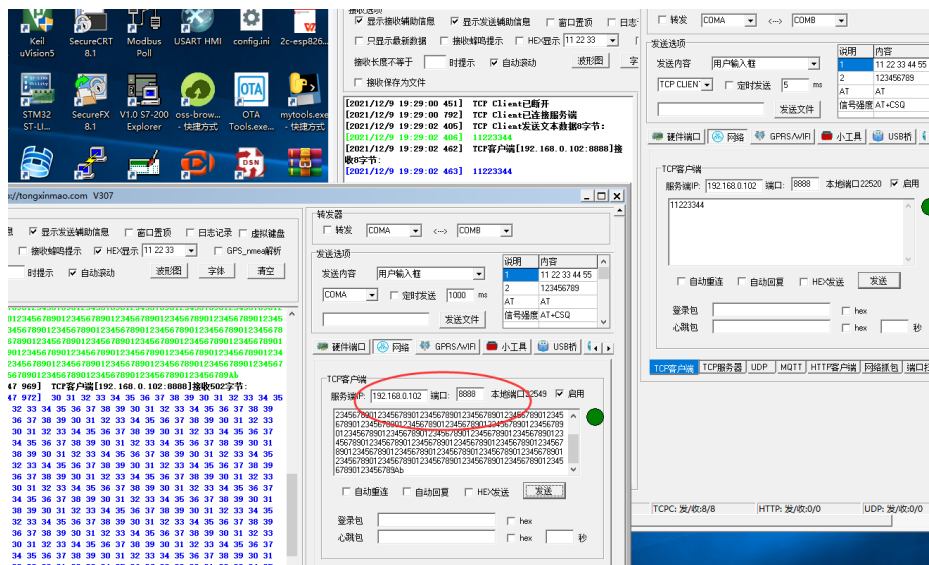
我现在电脑和模组在一个路由器下哈,我就使用那个192.168.0.102地址测试

4.打开网络调试助手测试

程序里面写的是接收到什么数据就返回什么数据

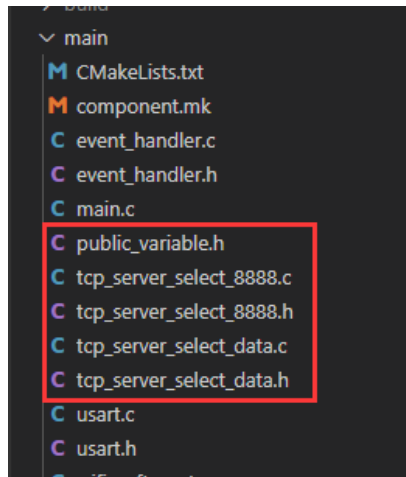


再加个客户端

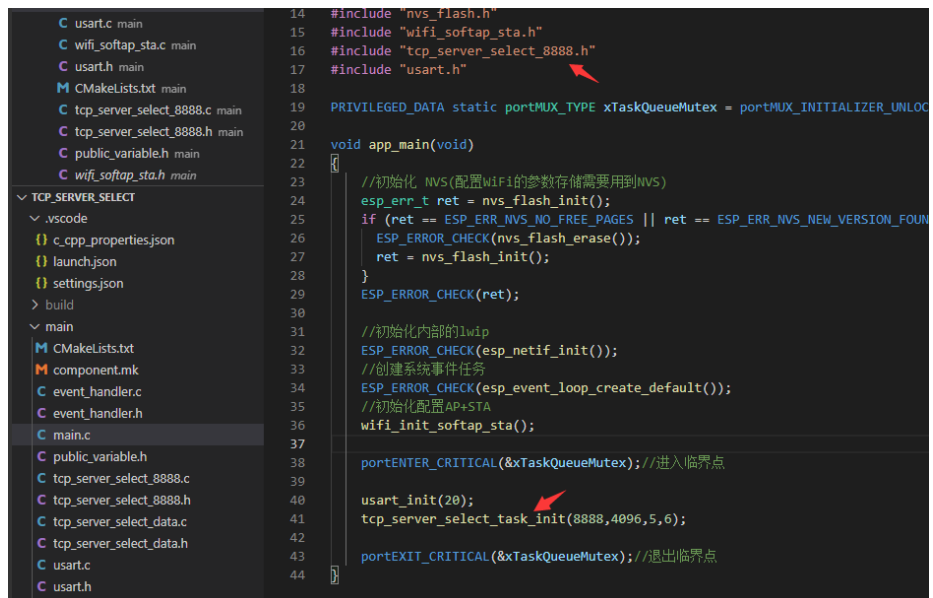


程序使用说明(先说下如何使用)

1.如果用户需要移植使用的话直接把下面的文件放到自己的工程里面就可以



2.创建TCP服务器(各个参数见下下图)

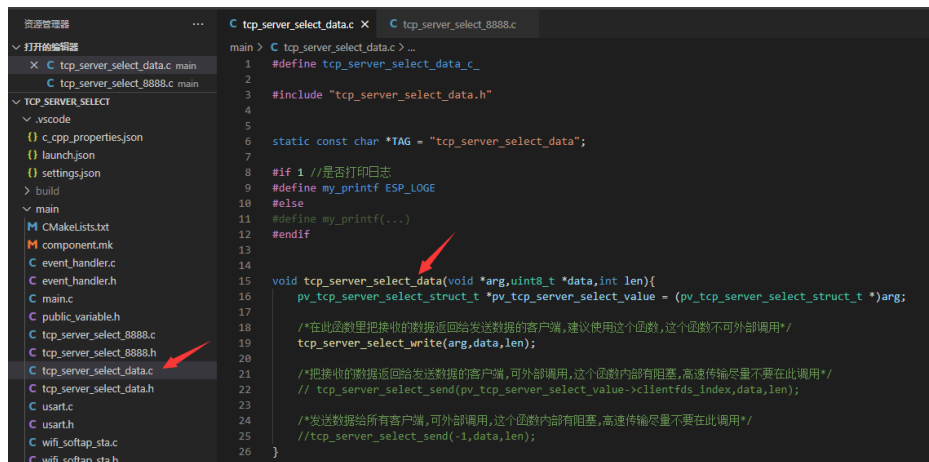



```

251  /**
252   * @brief 初始化TCP服务器
253   * @param tcp_server_select_port 服务器监听的端口号
254   * @param RingbufferSize Ringbuffer 大小(该大小直接影响服务器发送给客户端的数据个数)
255   * @param tcp_server_select_task_priority TCP服务器任务优先级
256   * @param tcp_server_select_send_task TCP服务器发送数据任务优先级
257   * @retval none
258   */
259 void tcp_server_select_task_init(int tcp_server_select_port,int RingbufferSize,int tcp_server_select_task_priority,int tcp_server_select_send_task_priority)
260 {
261     //监听的端口号
262     pv_tcp_server_select_struct_t1.tcp_server_select_port = tcp_server_select_port;
263     //创建Ringbuffer缓存用于发送数据
264     pv_tcp_server_select_struct_t1.ringbuf_handle = xRingbufferCreate(RingbufferSize, RINGBUF_TYPE_NOSPLIT);
265     if (pv_tcp_server_select_struct_t1.ringbuf_handle == NULL) {
266         my_printf(TAG, "Failed to create ring buffer\n");
267         while (1); //等待重启
268     }
269     //创建信号量
270     pv_tcp_server_select_struct_t1.SemaphoreHandle_t_count = xSemaphoreCreateCounting( 20, 0 );
271     //创建TCP服务器监听任务
272     xTaskCreate(tcp_server_select_task, "tcp_server_select_task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_task_priority, NULL);
273     //创建TCP发送数据任务
274     xTaskCreate(tcp_server_select_send_task, "tcp_server_select_send_task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_send_task_priority, NULL);
275 }
276

```

3.服务器接收到数据在这个里面(这个函数是在TCP监听任务里面的,注意不要在这个里面阻塞哈)



4.关于发送数据给客户端

1,发送数据给客户端有两个函数 tcp_server_select_write 和 tcp_server_select_send

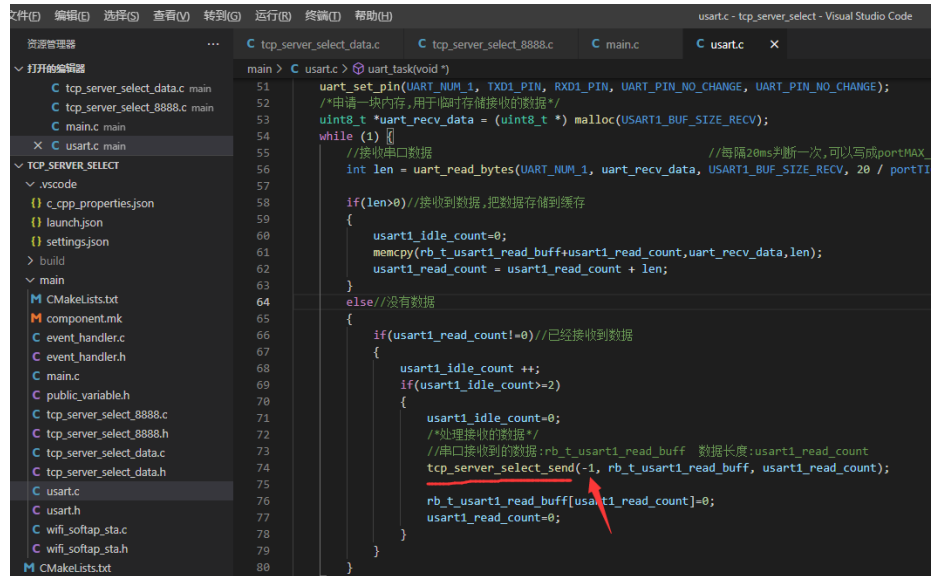
```

14 void tcp_server_select_data(void *arg,uint8_t *data,int len){
15     pv_tcp_server_select_struct_t *pv_tcp_server_select_value = (pv_tcp_server_select_struct_t *)arg;
16
17     /*在此函数里把接收的数据返回给发送数据的客户端,建议使用这个函数,这个函数不可外部调用*/
18     tcp_server_select_write(arg,data,len);
19
20     /*把接收的数据返回给发送数据的客户端,可外部调用,这个函数内部有阻塞,高速传输尽量不要在此调用*/
21     // tcp_server_select_send(pv_tcp_server_select_value->clientfds_index,data,len);
22
23     /*发送数据给所有客户端,可外部调用,这个函数内部有阻塞,高速传输尽量不要在此调用*/
24     //tcp_server_select_send(-1,data,len);
25 }
26

```

2, tcp_server_select_write 就是上面说的只能在接收数据里面调用才可以使用

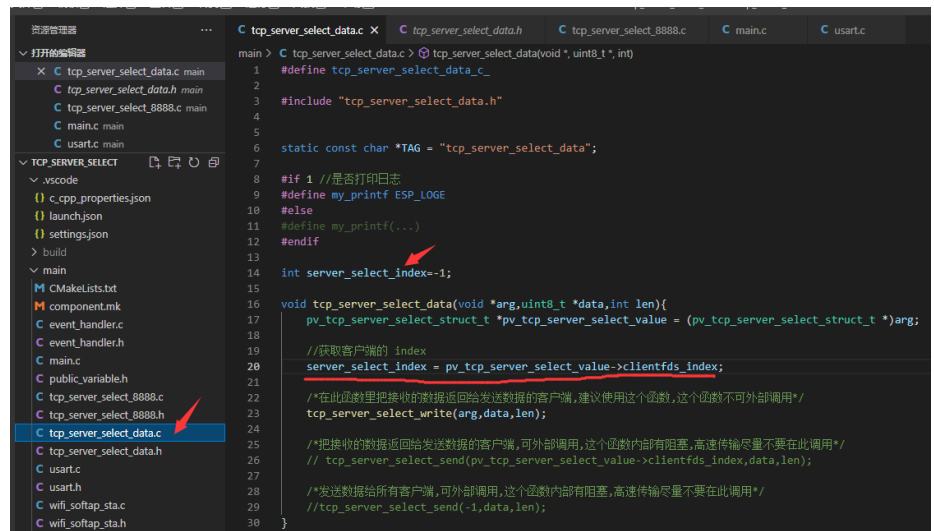
3.假设现在需要把串口接到的数据发送给所有TCP客户端 端 tcp_server_select_send(-1, 数据地址,数据长度)



```
51  uart_set_pin(UART_NUM_1, TXD1_PIN, RXD1_PIN, UART_PIN_NO_CHANGE, UART_PIN_NO_CHANGE);
52  /**申请一块内存,用于临时存储接收的数据*/
53  uint8_t *uart_rcv_data = (uint8_t *) malloc(USART1_BUF_SIZE_RECV);
54  while (1) {
55      //接收串口数据
56      int len = uart_read_bytes(UART_NUM_1, uart_rcv_data, USART1_BUF_SIZE_RECV, 20 / portTII);
57      //每隔20ms判断一次,可以写成portMAX
58      if(len>0)//接收到数据,把数据存储到缓存
59      {
60          usart1_idle_count=0;
61          memcpy(rb_t_usart1_read_buff+usart1_read_count,uart_rcv_data,len);
62          usart1_read_count = usart1_read_count + len;
63      }
64      else//没有数据
65      {
66          if(usart1_read_count!=0)//已经接收到数据
67          {
68              usart1_idle_count ++;
69              if(usart1_idle_count>=2)
70              {
71                  usart1_idle_count=0;
72                  /**处理接收的数据*/
73                  //串口接收到的数据:rb_t_usart1_read_buff 数据长度:usart1_read_count
74                  tcp_server_select_send(-1, rb_t_usart1_read_buff, usart1_read_count);
75                  rb_t_usart1_read_buff[usart1_read_count]=0;
76                  usart1_read_count=0;
77              }
78          }
79      }
80  }
```

4.假设现在需要把串口接到的数据发送给指定的TCP客户端,则需要先在接收函数里面获取客户端的 index

我只是举例子哈,一般是接收到什么数据以后再去赋值后面的数据发给哪个客户端



```
1  #define tcp_server_select_data_c_
2
3  #include "tcp_server_select_data.h"
4
5
6  static const char *TAG = "tcp_server_select_data";
7
8  #if 1 //是否打印日志
9  #define my_printf ESP_LOGE
10 #else
11 #define my_printf(...)
12 #endif
13
14 int server_select_index=-1;
15
16 void tcp_server_select_data(void *arg,uint8_t *data,int len){
17     pv_tcp_server_select_struct_t *pv_tcp_server_select_value = (pv_tcp_server_select_struct_t *)arg;
18
19     //获取客户端的 index
20     server_select_index = pv_tcp_server_select_value->clientfds_index;
21
22     /**在此函数里把接收的数据返回给发送数据的客户端,建议使用这个函数,这个函数不可外部调用*/
23     tcp_server_select_write(arg,data,len);
24
25     /**把接收的数据返回给发送数据的客户端,可外部调用,这个函数内部有阻塞,高速传输尽量不在此调用*/
26     // tcp_server_select_send(pv_tcp_server_select_value->clientfds_index,data,len);
27
28     /**发送数据给所有客户端,可外部调用,这个函数内部有阻塞,高速传输尽量不在此调用*/
29     //tcp_server_select_send(-1,data,len);
30 }
```

```
main > C usart.c > uart_task(void *)
/*申请一块内存,用于临时存储接收的数据*/
uint8_t *uart_recv_data = (uint8_t *) malloc(USART1_BUF_SIZE_RECV);
while (1) {
    //接收串口数据 //每隔20ms判断一次,可以写成portMAX_DELAY(一直判断)
    int len = uart_read_bytes(UART_NUM_1, uart_recv_data, USART1_BUF_SIZE_RECV, 20 / portTICK_RATE_MS);

    if(len>0)//接收到数据,把数据存到缓存
    {
        usart1_idle_count=0;
        memcpy(rb_t_usart1_read_buff+usart1_read_count,uart_recv_data,len);
        usart1_read_count = usart1_read_count + len;
    }
    else//没有数据
    {
        if(usart1_read_count!=0)//已经接收到数据
        {
            usart1_idle_count++;
            if(usart1_idle_count>=2)
            {
                usart1_idle_count=0;
                /*处理接收的数据*/
                //串口接收到的数据:rb_t_usart1_read_buff 数据长度:usart1_read_count

                if (server_select_index!=-1){
                    tcp_server_select_send(server_select_index, rb_t_usart1_read_buff, usart1_read_count);
                }

                rb_t_usart1_read_buff[usart1_read_count]=0;
                usart1_read_count=0;
            }
        }
    }
}
```

程序说明

1,创建TCP服务器

```
main > C main.c > app_main(void)
15 #include "wifi_softap_sta.h"
16 #include "tcp_server_select_8888.h"
17 #include "usart.h"
18
19 PRIVILEGED_DATA static portMUX_TYPE xTaskQueueMutex = portMUX_INITIALIZER_UNLOCKED;
20
21 void app_main(void)
22 {
23     //初始化 NVS(配置WiFi的参数存储需要用到NVS)
24     esp_err_t ret = nvs_flash_init();
25     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
26         ESP_ERROR_CHECK(nvs_flash_erase());
27         ret = nvs_flash_init();
28     }
29     ESP_ERROR_CHECK(ret);
30
31     //初始化内部的lwip
32     ESP_ERROR_CHECK(esp_netif_init());
33     //创建系统事件任务
34     ESP_ERROR_CHECK(esp_event_loop_create_default());
35     //初始化配置AP+STA
36     wifi_init_softap_sta();
37
38     portENTER_CRITICAL(&xTaskQueueMutex); //进入临界点
39
40     usart_init(20);
41     tcp_server_select_task_init(8888,4096,5,6);
42
43     portEXIT_CRITICAL(&xTaskQueueMutex); //退出临界点
44 }
```

```

C tcp_server_select_data.c  C tcp_server_select_data.h  C tcp_server_select_8888.c X  C main.c  C usart.c
main > C tcp_server_select_8888.c > tcp_server_select_task_init(int, int, int)
244     }
245     }
246     }
247     vTaskDelete(NULL);
248 }
249 }
250
251 /**
252  * @brief 初始化TCP服务器
253  * @param tcp_server_select_port 服务器监听的端口号
254  * @param RingbufferSize Ringbuffer 大小(该大小直接影响服务器发送给客户端的数据个数)
255  * @param tcp_server_select_task_priority TCP服务器任务优先级
256  * @param tcp_server_select_send_task TCP服务器发送数据任务优先级
257  * @retval none
258  */
259 void tcp_server_select_task_init(int tcp_server_select_port, int RingbufferSize, int tcp_server_select_task_priority, int tcp_server_select_send_task_priority)
260 {
261     //监听的端口号
262     pv_tcp_server_select_struct_t1.tcp_server_select_port = tcp_server_select_port;
263     //创建Ringbuffer缓存用来发送数据
264     pv_tcp_server_select_struct_t1.ringbuf_handle = xRingbufferCreate(RingbufferSize, RINGBUF_TYPE_NOSPLIT);
265     if (pv_tcp_server_select_struct_t1.ringbuf_handle == NULL) {
266         my_printf(TAG, "Failed to create ring buffer\n");
267         while (1); //等待重启
268     }
269     //创建信号量
270     pv_tcp_server_select_struct_t1.SemaphoreHandle_t_count = xSemaphoreCreateCounting( 20, 0 );
271     //创建TCP服务器监听任务
272     xTaskCreate(tcp_server_select_task, "tcp_server_select_task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_task_priority, NULL);
273     //创建TCP发送数据任务
274     xTaskCreate(tcp_server_select_send_task, "tcp_server_select_send_task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_send_task_priority, NULL);
275 }
276

```

2, TCP服务器监听任务, 在里面监听连接 和 接收数据

```

251 /**
252  * @brief 初始化TCP服务器
253  * @param tcp_server_select_port 服务器监听的端口号
254  * @param RingbufferSize Ringbuffer 大小(该大小直接影响服务器发送给客户端的数据个数)
255  * @param tcp_server_select_task_priority TCP服务器任务优先级
256  * @param tcp_server_select_send_task TCP服务器发送数据任务优先级
257  * @retval none
258  */
259 void tcp_server_select_task_init(int tcp_server_select_port, int RingbufferSize, int tcp_server_select_task_priority, int tcp_server_select_send_task_priority)
260 {
261     //监听的端口号
262     pv_tcp_server_select_struct_t1.tcp_server_select_port = tcp_server_select_port;
263     //创建Ringbuffer缓存用来发送数据
264     pv_tcp_server_select_struct_t1.ringbuf_handle = xRingbufferCreate(RingbufferSize, RINGBUF_TYPE_NOSPLIT);
265     if (pv_tcp_server_select_struct_t1.ringbuf_handle == NULL) {
266         my_printf(TAG, "Failed to create ring buffer\n");
267         while (1); //等待重启
268     }
269     //创建信号量
270     pv_tcp_server_select_struct_t1.SemaphoreHandle_t_count = xSemaphoreCreateCounting( 20, 0 );
271     //创建TCP服务器监听任务
272     xTaskCreate(tcp_server_select_task, "tcp_server_select_task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_task_priority, NULL);
273     //创建TCP发送数据任务
274     xTaskCreate(tcp_server_select_send_task, "tcp_server_select_send_task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_send_task_priority, NULL);
275 }
276

```

```

122  /**
123   * @brief select TCP服务器
124   * @param none
125   * @retval none
126   */
127  void tcp_server_select_task(void *arg)
128  {
129      char addr_str[128];
130      int sfd, cfd, maxfd, i, nready, len, err;
131      struct sockaddr_in server_addr, client_addr;
132      socklen_t client_addr_len;
133      fd_set read_set;
134
135      pv_tcp_server_select_struct_t *pv_tcp_server_select_value = (pv_tcp_server_select_struct_t *)arg;
136
137      //创建socket
138      sfd = socket(AF_INET, SOCK_STREAM, 0);
139      if(sfd<0){
140          my_printf(TAG, "Failed to allocate socket.");
141          while(1); //等待重启
142      }
143
144      server_addr.sin_family = AF_INET;
145      server_addr.sin_port = htons(pv_tcp_server_select_struct_t1.tcp_server_select_port);
146      server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
147      //绑定socket
148      err=bind(sfd, (struct sockaddr *)&server_addr, sizeof(server_addr));
149      if (err<0)
150      {
151          my_printf(TAG, "Failed to bind socket");
152          close(sfd);
153          while(1); //等待重启
154      }
155      //监听socket
156      err=listen(sfd, 5);
157      if (err<0)
158      {
159          my_printf(TAG, "Failed to listen socket");
160          close(sfd);
161          while(1); //等待重启

```

```

main / C:\tcp_server_select_8888.c / tcp_server_select_task(void *)
152      close(sfd);
153      while(1); //等待重启
154  }
155  //监听socket
156  err=listen(sfd, 5);
157  if (err<0)
158  {
159      my_printf(TAG, "Failed to listen socket");
160      close(sfd);
161      while(1); //等待重启
162  }
163  client_addr_len = sizeof(client_addr);
164  //初始化 maxfd 等于 sfd
165  maxfd = sfd;
166  //清空fdset
167  FD_ZERO(&pv_tcp_server_select_value->all_set);
168  //把sfd文件描述符添加到集合中
169  FD_SET(sfd, &pv_tcp_server_select_value->all_set);
170  //初始化客户端fd的集合
171  for(i = 0; i < FD_SETSIZE - 1; i++){
172      //初始化为-1
173      pv_tcp_server_select_value->clientfds[i] = -1;
174  }
175  while(1)
176  {
177      //每次select返回之后，fd_set集合就会变化，再select时，就不能使用，
178      //所以我们要保存设置fd_set 和 读取的fd_set
179      read_set = pv_tcp_server_select_value->all_set;
180      nready = select(maxfd + 1, &read_set, NULL, NULL, NULL);
181      //没有超时机制，不会返回0
182      if(nready < 0){
183          my_printf(TAG, "select error \r\n");
184          vTaskDelete(NULL);
185      }
186      //判断监听的套接字是否有数据
187      if(FD_ISSET(sfd, &read_set)){
188          //有了客户端进行连接了
189          cfd = accept(sfd, (struct sockaddr *)&client_addr, &client_addr_len);
190          if(cfd < 0){
191              if ((errno == ECONNABORTED) || (errno == EINTR)){
192                  //继续select
193                  continue;

```

```

C tcp_server_select_data.c C tcp_server_select_data.h C tcp_server_select_8888.c X C main.c C usart.c
main > C tcp_server_select_8888.c > tcp_server_select_task(void *)
210     if(pv_tcp_server_select_value->clientfds[i] == -1){
211         pv_tcp_server_select_value->clientfds[i] = cfd;
212         break;
213     }
214 }
215 //没有其它连接需要处理：这里防止重复工作，就不去执行其他任务
216 if(--nready == 0){
217     //继续select
218     continue;
219 }
220 }
221 //遍历所有的客户端
222 for(i = 0; i < FD_SETSIZE - 1; i++){
223     if(pv_tcp_server_select_value->clientfds[i] == -1){
224         continue;
225     }
226     if(FD_ISSET(pv_tcp_server_select_value->clientfds[i], &read_set)){//存在
227         again:
228         len = read(pv_tcp_server_select_value->clientfds[i], pv_tcp_server_select_value->read_buff, sizeof(pv_tcp_server_select_value->read_buff));
229         if(len < 0){
230             if (errno == EINTR){
231                 goto again;
232             }
233         }
234         if(len <= 0){
235             close(pv_tcp_server_select_value->clientfds[i]);
236             //从集合里面删除
237             FD_CLR(pv_tcp_server_select_value->clientfds[i], &pv_tcp_server_select_value->all_set);
238             //当前的客户端fd 赋值为-1
239             pv_tcp_server_select_value->clientfds[i] = -1;
240         }else{//接收到客户端消息
241             pv_tcp_server_select_value->clientfds_index = i;
242             tcp_server_select_data(pv_tcp_server_select_value, pv_tcp_server_select_value->read_buff, len);
243         }
244     }
245 }
246 }
247 vTaskDelete(NULL);
248 }
249 }

```

```

13
14 int server_select_index=-1;
15
16 void tcp_server_select_data(void *arg,uint8_t *data,int len){
17     pv_tcp_server_select_struct_t *pv_tcp_server_select_value = (pv_tcp_server_select_struct_t *)arg;
18
19     //获取客户端的 index
20     server_select_index = pv_tcp_server_select_value->clientfds_index;
21
22     /*在此函数里把接收的数据返回给发送数据的客户端,建议使用这个函数,这个函数不可外部调用*/
23     tcp_server_select_write(arg,data,len);
24
25     /*把接收的数据返回给发送数据的客户端,可外部调用,这个函数内部有阻塞,高速传输尽量不要在此调用*/
26     // tcp_server_select_send(pv_tcp_server_select_value->clientfds_index,data,len);
27
28     /*发送数据给所有客户端,可外部调用,这个函数内部有阻塞,高速传输尽量不要在此调用*/
29     //tcp_server_select_send(-1,data,len);
30 }
31

```

3,发送数据

发送数据是使用 Ringbuffer + 信号量 + 任务

```

250 /**
251  * @brief 初始化TCP服务器
252  * @param tcp_server_select_port 服务器监听的端口号
253  * @param RingbufferSize Ringbuffer 大小(该大小直接影响服务器发送给客户端的数据个数)
254  * @param tcp_server_select_task_priority TCP服务器任务优先级
255  * @param tcp_server_select_send_task TCP服务器发送数据任务优先级
256  * @retval none
257  */
258 void tcp_server_select_task_init(int tcp_server_select_port,int RingbufferSize,int tcp_server_select_task_priority,int tcp_server_select_send_task_priority)
259 {
260     //监听的端口号
261     pv_tcp_server_select_struct_t1.tcp_server_select_port = tcp_server_select_port;
262     //创建Ringbuffer缓存用来发送数据
263     pv_tcp_server_select_struct_t1.ringbuf_handle = xRingbufferCreate(RingbufferSize, RINGBUF_TYPE_NOSPLIT);
264     if (pv_tcp_server_select_struct_t1.ringbuf_handle == NULL) {
265         my_printf(TAG, "Failed to create ring buffer\n");
266         while (1); //等待重启
267     }
268     //创建信号量
269     pv_tcp_server_select_struct_t1.SemaphoreHandle.t_count = xSemaphoreCreateCounting( 20, 0 );
270     //创建TCP服务器监听任务
271     xTaskCreate(tcp_server_select_task, "tcp_server_select_task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_task_priority, NULL);
272     //创建TCP发送数据任务
273     xTaskCreate(tcp_server_select_send_task, "tcp_server_select_send task", 4096, &pv_tcp_server_select_struct_t1, tcp_server_select_send_task_priority, NULL);
274 }
275
276

```

发送数据的时候是把数存储到 Ringbuffer 然后 信号量 +1

```
C tcp_server_select_data.c C tcp_server_select_data.h C tcp_server_select_8888.c X C main.c C usart.c
main > C tcp_server_select_8888.c > ...
18
19
20 /** 发送数据给客户端
21 * @param index 把数据发给哪一个客户端(填写-1是发送给全部客户端)
22 * @param data 要发送的数据
23 * @param length 要发送的数据长度
24 * @retval none
25 */
26 void tcp_server_select_send(int index,uint8_t *data,int len)
27 {
28     pv_tcp_server_select_struct_t1.index = index;
29     UBaseType_t res = xRingbufferSend(pv_tcp_server_select_struct_t1.ringbuf_handle, data, len, portMAX_DELAY);
30     if (res != pdTRUE) {
31         my_printf(TAG, "Failed to send item\n");
32     }
33     else{
34         xSemaphoreGive(pv_tcp_server_select_struct_t1.SemaphoreHandle_t_count);
35     }
36 }
37
38
```

任务里面获取信号量然后获取缓存里面的数据,然后发送数据给客户端

```
C tcp_server_select_data.c C tcp_server_select_data.h C tcp_server_select_8888.c X C main.c C usart.c
main > C tcp_server_select_8888.c > ...
39
40 /** 发送数据给客户端
41 * @param arg 实际为pv_tcp_server_select_struct_t结构体
42 * @param data 要发送的数据
43 * @param length 要发送的数据长度
44 * @retval none
45 */
46 void tcp_server_select_send_task(void *arg){
47     pv_tcp_server_select_struct_t *pv_tcp_server_select_value = (pv_tcp_server_select_struct_t *)arg;
48     int i,len;
49     size_t item_size;
50     char *item;
51     while (true){
52         //获取一个信号量
53         if(xSemaphoreTake(pv_tcp_server_select_value->SemaphoreHandle_t_count,portMAX_DELAY)){
54             //获取从队列里面的数据
55             item = (char *)xRingbufferReceive(pv_tcp_server_select_value->ringbuf_handle, &item_size, 1);
56             if (item_size>0){//从队列里面有数据
57                 if (pv_tcp_server_select_value->index!=-1){//给指定的客户端发送数据
58                     again1:
59                     //发送数据给客户端
60                     len = write(pv_tcp_server_select_value->clientfds[pv_tcp_server_select_value->index], item, item_size);
61                     if(len < 0){
62                         if (errno == EINTR){
63                             goto again1;
64                         }
65                         //从集合里面清除
66                         fd_clr(pv_tcp_server_select_value->clientfds[pv_tcp_server_select_value->index], &pv_tcp_server_select_value->all_set);
67                         //当前的客户端fd 赋值为-1
68                         pv_tcp_server_select_value->clientfds[pv_tcp_server_select_value->index] = -1;
69                     }
70                 }
71             }
72             else{//发送数据给全部客户端
73                 for(i = 0; i < FD_SETSIZE -1 ; i++){//遍历所有的客户端
74                     if(pv_tcp_server_select_value->clientfds[i] != -1){
75                         again:
76                         //发送数据给客户端
77                         len = write(pv_tcp_server_select_value->clientfds[i], item, item_size);
78                         if(len < 0){
79                             if (errno == EINTR){
80                                 goto again;
81                             }
82                         }
83                     }
84                 }
85             }
86         }
87     }
88 }
```

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 - 693

0

0


« 上一篇: 2-6-2-视频传输,监控,直播方案-android手机连接ESP32的热点,使用手机APP查看摄像头图像(WiFi视频小车,局域网视频监控)

posted on 2021-12-09 20:21 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

支持 Markdown

 自动补全

提交评论 退出

[Ctrl+Enter快捷键提交]

- 【推荐】字节跳动旗下火山引擎，邀您共同揭秘“云+增长”
- 【推荐】冬天里的一把火，园子里的一朵云：满园尽是火山引擎
- 【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】华为 HMS Core 线上 Codelabs 挑战赛第4期，探索“智”感生活

编辑推荐：

- 理解ASP.NET Core - 模型绑定&验证
- [翻译].NET 6 中的 dotnet monitor
- .NET Core 如何配置 TLS Cipher (套件) ?
- 记一次 .NET 某智能服装智造系统 内存泄漏分析
- 大学毕业三年的一些经历与思考



最新新闻：

- 快手裁员30%？员工：现在管理严了，有人盯紧日报和周报 (2021-12-09 18:12)
 - 全球排名系统Alexa Rank网站将于2022年5月1日关闭 (2021-12-09 18:00)
 - Kickstarter押注区块链技术：欲打造去中心化的众筹平台 (2021-12-09 17:50)
 - 物理学家利用空间反射和时间反转对称性对量子材料实现了更好控制 (2021-12-09 17:44)
 - 对抗肝癌的新组合：CRISPR基因疗法、超声波和药物 (2021-12-09 17:33)
- » 更多新闻...

历史上的今天：

2018-12-09 Wi-Fi无线控制器开发例程(基础篇)

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。