

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 - 816, 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

- 渡我不渡她
- 小镇姑娘
- 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(8)
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(27)
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)

206-ESP32_SDK开发-TCP客户端(select方式,自动重连)

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

ESP32_SDK开发(源码见资料源码)

开发板链接:[ESP32开发板链接](#)

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=fXgd3U0o>

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热点](#)
- [203-softAP+station共存模式](#)
- [204-TCP服务器\(模组AP热点模式,支持多个客户端连接通信\)\(废弃,项目绝对不能用\)](#)
- [205-TCP服务器\(select方式,支持多连接,高速高并发传输\)](#)
- [206-TCP客户端\(select方式,自动重连\)](#)

-----图片传输-----

- [801-ESP32\(WiFi\)把采集的摄像头照片数据通过串口输出到串口上位机显示\(C# 串口上位机\)](#)
- [802-ESP32\(WiFi\)把采集的摄像头照片数据通过UDP发送给UDP客户端\(C# UDP客户端\)](#)

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)物联网
开发(9)
HC32F460(华大单片机)学习开
发(8)
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(ZLESP826
6A)物联网开发(1)
STM32+ESP8266+AIR202/3
02远程升级方案(16)
STM32+ESP8266+AIR202/3
02终端管理方案(6)
STM32+ESP8266+Air302物
联网开发(65)
STM32+W5500+AIR202/30
2基本控制方案(25)
STM32+W5500+AIR202/30
2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)
更多

阅读排行榜

1. ESP8266使用详解(AT,LUA, SDK)(174312)
2. 1-安装MQTT服务器(Windows),并连接测试(105197)
3. 用ESP8266+android,制作自己的WIFI小车(ESP8266篇)(67860)
4. ESP8266刷AT固件与node mcu固件(66934)
5. 有人WIFI模块使用详解(39426)
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)(37088)
7. C#中public与private与static(35949)
8. 关于TCP和MQTT之间的转换(35303)
9. android 之TCP客户端编程(33032)
10. android服务端+eps8266+单片机+路由器之远程控制系统(31700)

推荐排行榜

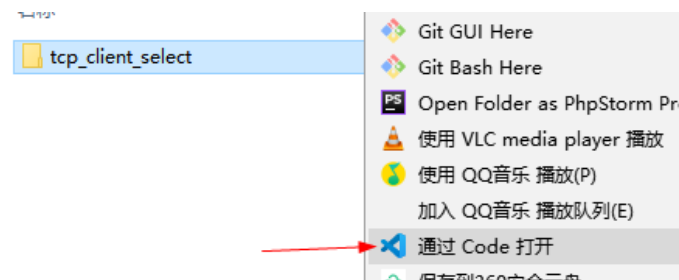
- [803-手机连接ESP32的热点,使用微信小程序查看摄像头图像\(WiFi视频小车,局域网视频监控\)](#)
- [804-手机连接ESP32的热点,使用android APP查看摄像头图像\(WiFi视频小车,局域网视频监控\)](#)
-
-
-

下载程序到开发板

1.把这节的代码放到英文目录

tcp_client_select

2.鼠标右键选择使用VScode打开



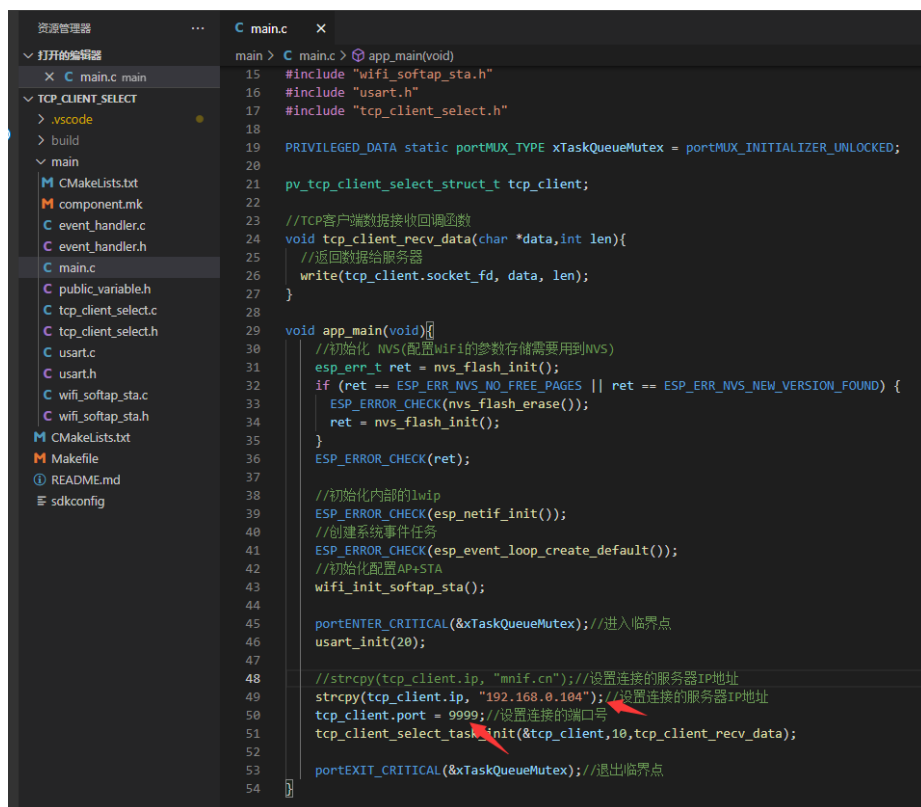
3.鼠标右键选择使用VScode打开

设置所连接服务器的IP地址和端口号

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)

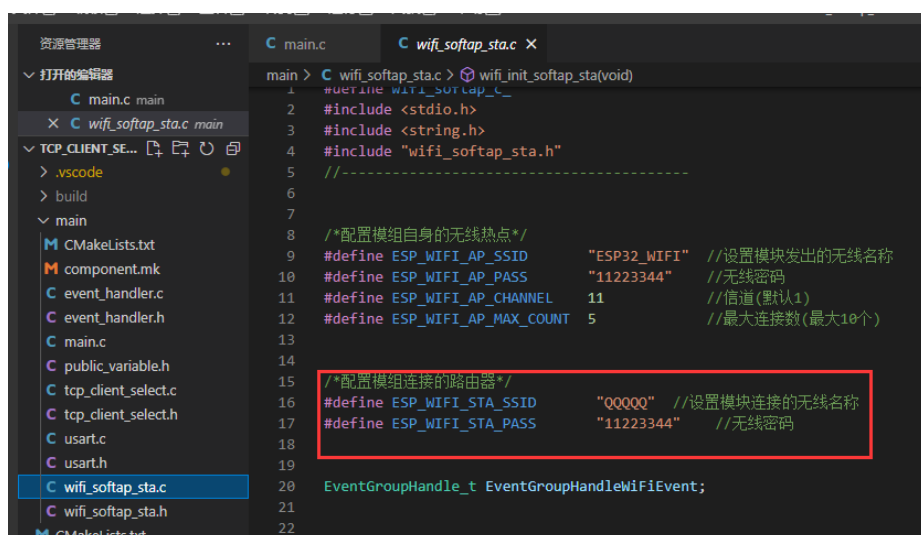
最新评论

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



```
main > C main.c > app_main(void)
15 #include "wifi_softap_sta.h"
16 #include "usart.h"
17 #include "tcp_client_select.h"
18
19 PRIVILEGED_DATA static portMUX_TYPE xTaskQueueMutex = portMUX_INITIALIZER_UNLOCKED;
20
21 pv_tcp_client_select_struct_t tcp_client;
22
23 //TCP客户端数据接收回调函数
24 void tcp_client_recv_data(char *data,int len){
25     //返回数据给服务器
26     write(tcp_client.socket_fd, data, len);
27 }
28
29 void app_main(void){
30     //初始化 NVS(配置WiFi的参数存储需要用到NVS)
31     esp_err_t ret = nvs_flash_init();
32     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
33         ESP_ERROR_CHECK(nvs_flash_erase());
34         ret = nvs_flash_init();
35     }
36     ESP_ERROR_CHECK(ret);
37
38     //初始化内部的lwip
39     ESP_ERROR_CHECK(esp_netif_init());
40     //创建系统事件任务
41     ESP_ERROR_CHECK(esp_event_loop_create_default());
42     //初始化配置AP+STA
43     wifi_init_softap_sta();
44
45     portENTER_CRITICAL(&xTaskQueueMutex); //进入临界点
46     usart_init(20);
47
48     //strcpy(tcp_client.ip, "mnif.cn"); //设置连接的服务器IP地址
49     strcpy(tcp_client.ip, "192.168.0.104"); //设置连接的服务器IP地址
50     tcp_client.port = 9999; //设置连接的端口号
51     tcp_client_select_task_init(&tcp_client,10,tcp_client_recv_data);
52
53     portEXIT_CRITICAL(&xTaskQueueMutex); //退出临界点
54 }
```

4.注意,如果设置的地址是域名,因为需要DNS把域名解析成IP,所以需要设置一下连接的路由器



```
main > C wifi_softap_sta.c > wifi_init_softap_sta(void)
1 #define WIFISOFTAP_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 "QQQQQ" //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 EventGroupHandle_t EventGroupHandleWifiEvent;
21
22
```

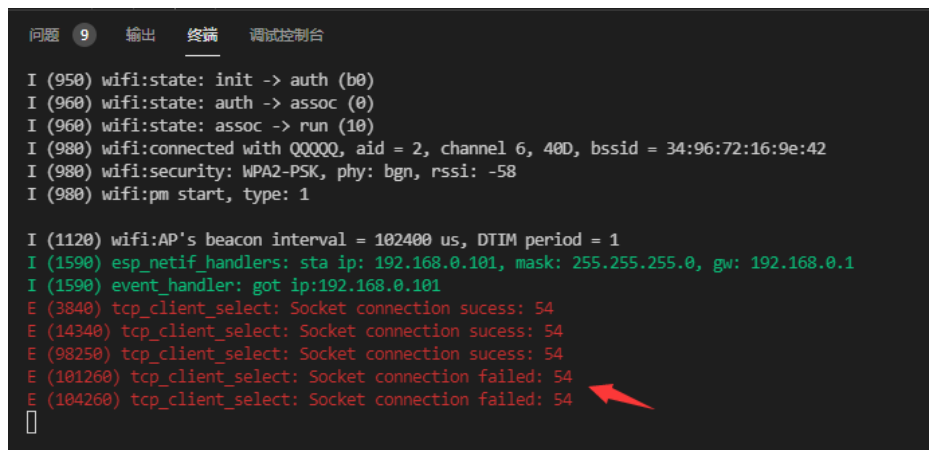
5.下载测试

现在是服务器发送给客户端什么,客户端就回复什么



5.测试断线重连

把服务器关掉,会看到客户端每隔一段时间尝试重新连接



重新打开服务器,客户端就连接了

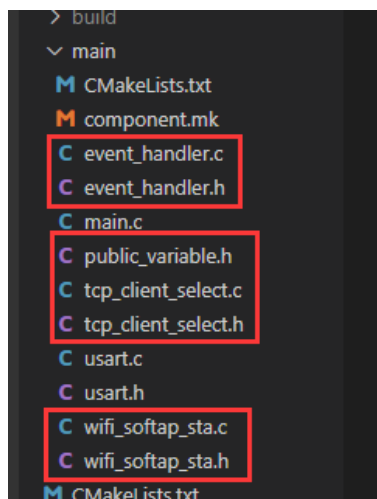
```
问题 9 输出 终端 调试控制台

E (98250) tcp_client_select: Socket connection sucess: 54
E (101260) tcp_client_select: Socket connection failed: 54
E (104260) tcp_client_select: Socket connection failed: 54
E (107260) tcp_client_select: Socket connection failed: 54
E (110260) tcp_client_select: Socket connection failed: 54
E (113260) tcp_client_select: Socket connection failed: 54
E (116260) tcp_client_select: Socket connection failed: 54
E (119260) tcp_client_select: Socket connection failed: 54
E (122270) tcp_client_select: Socket connection failed: 54
E (125270) tcp_client_select: Socket connection failed: 54
E (128270) tcp_client_select: Socket connection failed: 54
E (131270) tcp_client_select: Socket connection failed: 54
E (134290) tcp_client_select: Socket connection failed: 54
E (137290) tcp_client_select: Socket connection failed: 54
E (140300) tcp_client_select: Socket connection failed: 54
E (143300) tcp_client_select: Socket connection failed: 54
E (146300) tcp_client_select: Socket connection failed: 54
E (149320) tcp_client_select: Socket connection failed: 54
E (152320) tcp_client_select: Socket connection failed: 54
E (155330) tcp_client_select: Socket connection failed: 54
E (158330) tcp_client_select: Socket connection failed: 54
E (161330) tcp_client_select: Socket connection failed: 54
E (164330) tcp_client_select: Socket connection failed: 54
E (167340) tcp_client_select: Socket connection failed: 54
E (170340) tcp_client_select: Socket connection sucess: 54
```



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.配置连接服务器

```
资源管理器  ...  C main.c  X  C tcp_client_select.c  ① README.md

打开的编辑器
X C main.c main
C tcp_client_select.c main
① README.md

TCP_CLIENT_SELECT
> .vscode
> build
> main
M CMakeLists.txt
M component.mk
C event_handler.c
C event_handler.h
C main.c
C public_variable.h
C tcp_client_select.c
C tcp_client_select.h
C usart.c
C usart.h
C wifi_softap_sta.c
C wifi_softap_sta.h
M CMakeLists.txt
M Makefile
① README.md
≡ sdkconfig

main > C main.c > app_main(void)
15 #include "wifi_softap_sta.h"
16 #include "usart.h"
17 #include "tcp_client_select.h"
18
19 PRIVILEGED_DATA static portMUX_TYPE xTaskQueueMutex = portMUX_INITIALIZER_UNLOCKED;
20
21 pv_tcp_client_select_struct_t tcp_client;
22
23 //TCP客户端数据接收回调函数
24 void tcp_client_recv_data(char *data,int len){
25     //返回数据给服务器
26     write(tcp_client.socket_fd, data, len);
27 }
28
29 void app_main(void){
30     //初始化 NVS(配置WiFi的参数存储需要用到NVS)
31     esp_err_t ret = nvs_flash_init();
32     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
33         ESP_ERROR_CHECK(nvs_flash_erase());
34         ret = nvs_flash_init();
35     }
36     ESP_ERROR_CHECK(ret);
37
38     //初始化内部的lwip
39     ESP_ERROR_CHECK(esp_netif_init());
40     //创建系统事件任务
41     ESP_ERROR_CHECK(esp_event_loop_create_default());
42     //初始化配置AP+STA
43     wifi_init_softap_sta();
44
45     portENTER_CRITICAL(&xTaskQueueMutex); //进入临界点
46     usart_init(20);
47
48     //strcpy(tcp_client.ip, "mnif.cn"); //设置连接的服务器IP地址
49     strcpy(tcp_client.ip, "192.168.0.104"); //设置连接的服务器IP地址
50     tcp_client.port = 9999; //设置连接的端口号
51     tcp_client_select_task_init(&tcp_client,10,tcp_client_recv_data);
52
53     portEXIT_CRITICAL(&xTaskQueueMutex); //退出临界点
54 }
```

3.关于 tcp_client_select_task_init 函数

```
42 //初始化配置AP+STA
43 wifi_init_softap_sta();
44
45 portENTER_CRITICAL(&xTaskQueueMutex); //进入临界点
46 usart_init(20);
47
48 //strcpy(tcp_client.ip, "mnif.cn"); //设置连接的服务器IP地址
49 strcpy(tcp_client.ip, "192.168.0.104"); //设置连接的服务器IP地址
50 tcp_client.port = 9999; //设置连接的端口号
51 tcp_client_select_task_init(&tcp_client,10,tcp_client_recv_data);
52
53 portEXIT_CRITICAL(&xTaskQueueMutex); //退出临界点
54 }
```

```
/**
 * @brief select TCP客户端
 * @param pv_tcp_client_select_value 客户端结构体
 * @param task_priority TCP客户端任务优先级
 * @param RecvDataCallBack 接收回调函数
 * @retval none
 */
void tcp_client_select_task_init(pv_tcp_client_select_struct_t *pv_tcp_client_select_value,int task_priority,void(*RecvDataCallBack)(char *data,int len)){
    pv_tcp_client_select_value->socket_fd=-1;
    pv_tcp_client_select_value->connected=0;
    pv_tcp_client_select_value->CallBack = RecvDataCallBack;
    //创建TCP任务
    xTaskCreate(tcp_client_select_task, "tcp_client_select_task", 4096, pv_tcp_client_select_value , task_priority, NULL);
}
```

4.客户端发送数据

如果在接收回调函数里面发送直接调用下面的函数就可以

第一个参数 tcp_client.socket_fd 固定,后面分别是发送的数据和数据长度

```
C main.c X C tcp_client_select.c README.md
main > C main.c > ...
10 #include "esp_wifi.h"
11 #include "esp_netif.h"
12 #include "esp_log.h"
13
14 #include "nvs_flash.h"
15 #include "wifi_softap_sta.h"
16 #include "usart.h"
17 #include "tcp_client_select.h"
18
19 PRIVILEGED_DATA static portMUX_TYPE xTaskQueueMutex = portMUX_INITIALIZER_UNLOCKED;
20
21 pv_tcp_client_select_struct_t tcp_client;
22
23 //TCP客户端数据接收回调函数
24 void tcp_client_recv_data(char *data,int len){
25     //返回数据给服务器
26     write(tcp_client.socket_fd, data, len);
27 }
28
29 void app_main(void){
30     //初始化 NVS (配置WiFi的参数存储需要用到NVS)
31     esp_err_t ret = nvs_flash_init();
32     if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
```

假设把串口接收的数据发送给服务器, 引用一下 tcp_client 变量

```
资源管理器 ... C main.c X C usart.c C tcp_client_select.c
打开的编辑器
X C main.c main
C usart.c main
C tcp_client_select.c main
TCP_CLIENT_SELECT
> .vscode
> build
main
CMakeLists.txt
component.mk
event_handler.c
event_handler.h
main.c
public_variable.h
tcp_client_select.c
tcp_client_select.h
usart.c
usart.h
main > C main.c > ...
10 #include "esp_wifi.h"
11 #include "esp_netif.h"
12 #include "esp_log.h"
13
14 #include "nvs_flash.h"
15 #include "wifi_softap_sta.h"
16 #include "usart.h"
17 #include "tcp_client_select.h"
18
19 PRIVILEGED_DATA static portMUX_TYPE xTaskQueueMutex = portMUX_INITIALIZER_UNLOCKED;
20
21 pv_tcp_client_select_struct_t tcp_client;
22
23 //TCP客户端数据接收回调函数
24 void tcp_client_recv_data(char *data,int len){
25     //返回数据给服务器
26     write(tcp_client.socket_fd, data, len);
27 }
28
```

```
资源管理器 ... C main.c C usart.c X C tcp_client_select.c
打开的编辑器
C main.c main
X C usart.c main
C tcp_client_select.c main
TCP_CLIENT_SELECT
> .vscode
> build
main
CMakeLists.txt
component.mk
event_handler.c
event_handler.h
main.c
public_variable.h
tcp_client_select.c
tcp_client_select.h
C usart.c
C usart.h
C wifi_softap_sta.c
C wifi_softap_sta.h
CMakeLists.txt
Makefile
README.md
sdkconfig
main > C usart.c > ...
1 #define usart_c_
2
3 #include "usart.h"
4 #include "main.c"
5
6 static const char *TAG = "usart_function";
7
8
9
10 #define TXD1_PIN (GPIO_NUM_17) //串口1的发送数据引脚
11 #define RXD1_PIN (GPIO_NUM_16) //串口1的接收数据引脚
12
13 #define USART1_BUF_SIZE_RECV (1024) //接收数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN
14 #define USART1_BUF_SIZE_SEND (1024) //接收数据缓存大小,该大小需要大于内部FIFO大小:UART_FIFO_LEN
15
16 uint8_t rb_t_usart1_read_buff[USART1_BUF_SIZE_RECV];
17
18 extern pv_tcp_client_select_struct_t tcp_client;
19
20
21 /*串口发送数据*/
22 void usart_send(uint8_t *data ,int len)
23 {
24     uart_write_bytes(UART_NUM_1, (const char *) data, len);
25 }
26
```

```
资源管理器  ...  C main.c  C usart.c  C tcp_client_select.c

打开的编辑器
  C main.c main
  X C usart.c main
  C tcp_client_select.c main
TCP_CLIENT_SELECT
  > .vscode
  > build
  > main
  M CMakeLists.txt
  M component.mk
  C event_handler.c
  C event_handler.h
  C main.c
  C public_variable.h
  C tcp_client_select.c
  C tcp_client_select.h
  C usart.c
  C usart.h
  C wifi_softap_sta.c
  C wifi_softap_sta.h
  M CMakeLists.txt
  M Makefile
  ① README.md
  ② sdkconfig

main > C usart.c > uart_task(void *)
55  uint8_t *uart_rcv_data = (uint8_t *) malloc(USART1_BUF_SIZE_RECV);
56  while (1) {
57      //接收串口数据 //每隔20ms判断一次,可以写成portMAX_DELAY
58      int len = uart_read_bytes(USART_NUM_1, uart_rcv_data, USART1_BUF_SIZE_RECV, 20 / portTICK_
59
60      if(len>0)//接收到数据,把数据存到缓存
61      {
62          usart1_idle_count=0;
63          memcpy(rb_t_usart1_read_buff+usart1_read_count,uart_rcv_data,len);
64          usart1_read_count = usart1_read_count + len;
65      }
66      else//没有数据
67      {
68          if(usart1_read_count!=0)//已经接收到数据
69          {
70              usart1_idle_count ++;
71              if(usart1_idle_count>=2)
72              {
73                  usart1_idle_count=0;
74                  /*处理接收的数据*/
75                  //串口接收到的数据:rb_t_usart1_read_buff 数据长度:usart1_read_count
76
77                  if (tcp_client.connected==1)//判断一下是否已经连接
78                  {
79                      write(tcp_client.socket_fd, rb_t_usart1_read_buff, usart1_read_count);
80                  }
81
82                  rb_t_usart1_read_buff[usart1_read_count]=0;
83                  usart1_read_count=0;
84              }
85          }
86      }
87  }
```

程序说明

1.

```
C main.c  X  C usart.c  C tcp_client_select.c

main > C main.c > ...
12  #include "esp_log.h"
13
14  #include "nvs_flash.h"
15  #include "wifi_softap_sta.h"
16  #include "usart.h"
17  #include "tcp_client_select.h"
18
19  PRIVILEGED_DATA static portMUX_TYPE xTaskQueueMutex = portMUX_INITIALIZER_UNLOCKED;
20
21  pv_tcp_client_select_struct_t tcp_client;  创建tcp客户端结构体变量
22
23  //TCP客户端数据接收回调函数
24  void tcp_client_rcv_data(char *data,int len){  设置数据接收回调函数
25      //返回数据给服务器
26      write(tcp_client.socket_fd, data, len);
27  }
28
29  void app_main(void){
30      //初始化 NVS(配置wifi的参数存储需要用到NVS)
31      esp_err_t ret = nvs_flash_init();
32      if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
33          ESP_ERROR_CHECK(nvs_flash_erase());
34          ret = nvs_flash_init();
35      }
36      ESP_ERROR_CHECK(ret);
37
38      //初始化内部的lwip
39      ESP_ERROR_CHECK(esp_netif_init());
40      //创建系统事件任务
41      ESP_ERROR_CHECK(esp_event_loop_create_default());
42      //初始化配置AP+STA
43      wifi_init_softap_sta();
44
45      portENTER_CRITICAL(&xTaskQueueMutex);//进入临界点
46      usart_init(20);
47
48      //strcpy(tcp_client.ip, "mnif.cn");//设置连接的服务器IP地址
49      strcpy(tcp_client.ip, "192.168.0.104");//设置连接的服务器IP地址
50      tcp_client.port = 9999;//设置连接的端口号
51      tcp_client_select_task_init(&tcp_client,10,tcp_client_rcv_data);  初始化配置
52
53      portEXIT_CRITICAL(&xTaskQueueMutex);//退出临界点
54  }
```


2.启动任务

```
/**
 * @brief select TCP客户端
 * @param pv_tcp_client_select_value 客户端结构体
 * @param task_priority TCP客户端任务优先级
 * @param RecvDataCallback 接收回调函数
 * @retval none
 */
void tcp_client_select_task_init(pv_tcp_client_select_struct_t *pv_tcp_client_select_value, int task_priority, void(*RecvDataCallback)(char *data, int len)){
    pv_tcp_client_select_value->socket_fd=-1;
    pv_tcp_client_select_value->connected=0;
    pv_tcp_client_select_value->CallBack = RecvDataCallback;
    //创建TCP任务
    xTaskCreate(tcp_client_select_task, "tcp_client_select_task", 4096, pv_tcp_client_select_value, task_priority, NULL);
}
```

3.如果需要解析域名,则等待连接上路由器

```
19 //
20 void tcp_client_select_task(void *arg){
21
22     pv_tcp_client_select_struct_t *pv_tcp_client_select_value = (pv_tcp_client_select_struct_t *)arg;
23     uint8_t need_dns=0;
24
25     struct addrinfo hints = {
26         .ai_family = AF_INET,
27         .ai_socktype = SOCK_STREAM,
28     };
29     struct addrinfo *res;
30
31     int err,len;
32     int socket_fd;
33     char connection_state=0;
34     struct sockaddr_in server_addr, local_addr;
35     fd_set read_set;
36
37     //尝试DNS解析,如果解析失败说明确实需要DNS解析,但是需要先连接路由器
38     err = getaddrinfo(pv_tcp_client_select_value->ip, NULL, &hints, &res);
39     if(err != 0 || res == NULL) {
40         my_printf(TAG, "DNS lookup failed err=%d res=%p", err, res);
41         need_dns=1;
42     }
43     if (need_dns){//等待模组连接上路由器
44         xEventGroupWaitBits(EventGroupHandleWifiEvent, EventBitsWifiStaConnected, false, true, portMAX_DELAY);
45     }
46     err = getaddrinfo(pv_tcp_client_select_value->ip, NULL, &hints, &res);
47     if(err != 0 || res == NULL) {
48         my_printf(TAG, "DNS lookup failed err=%d res=%p", err, res);
49         return;
50     }
51
52     /*本地地址配置*/
53     local_addr.sin_family = AF_INET;
54
55     /*服务器配置*/
56     server_addr.sin_family = AF_INET;
```

4.创建socket 尝试连接服务器,连接成功以后就进入 select 阻塞函数

```
main > C tcp_client_select.c > tcp_client_select_task(void *)
58 server_addr.sin_addr.s_addr = ((struct sockaddr_in *) (res->ai_addr))>>sin_addr.s_addr;
59 freeaddrinfo(res);
60 while (true){
61     //创建socket
62     socket_fd = socket(AF_INET, SOCK_STREAM, 0);
63     if(socket_fd<0){
64         my_printf(TAG, "Failed to allocate socket.");
65     }
66     //绑定socket
67     err=bind(socket_fd, (struct sockaddr *)&local_addr, sizeof(local_addr));
68     if (err<0){
69         my_printf(TAG, "Failed to bind socket");
70         close(socket_fd);
71     }
72     //清空fdset
73     FD_ZERO(&read_set);
74     //把sfd文件描述符添加到集合中
75     FD_SET(socket_fd, &read_set);
76
77     if (connect(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) != 0) {
78         ESP_LOGE(TAG, "Socket connection failed: %d", socket_fd);
79         close(socket_fd);
80     }
81     else{
82         ESP_LOGE(TAG, "Socket connection success: %d", socket_fd);
83         connection_state=1;
84         pv_tcp_client_select_value->socket_fd=socket_fd;
85         pv_tcp_client_select_value->connected=1;
86     }
87     vTaskDelay(3000 / portTICK_PERIOD_MS); //延时
88     while (connection_state)
89     {
90         err = select(socket_fd+1,&read_set,NULL,NULL,NULL);
91         //没有超时机制，不会返回0
92         if(err < 0){
93             my_printf(TAG, "select error \r\n");
94             close(socket_fd);
95             connection_state=0;
96             pv_tcp_client_select_value->connected=0;
97         }
98
99         if(FD_ISSET(socket_fd,&read_set))
100         {
101             again:
102             len = read(socket_fd, pv_tcp_client_select_value->read_buff, sizeof(pv_tcp_client_select_value->read_buff));
103             if(len<0){
```

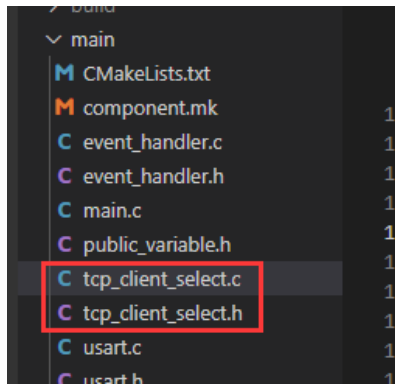
5.读取数据,调用数据回调函数

```
    }
    vTaskDelay(3000 / portTICK_PERIOD_MS); //延时
    while (connection_state)
    {
        err = select(socket_fd+1,&read_set,NULL,NULL,NULL);
        //没有超时机制，不会返回0
        if(err < 0){
            my_printf(TAG, "select error \r\n");
            close(socket_fd);
            connection_state=0;
            pv_tcp_client_select_value->connected=0;
        }

        if(FD_ISSET(socket_fd,&read_set)) //有消息
        {
            again: //读取数据
            len = read(socket_fd, pv_tcp_client_select_value->read_buff, sizeof(pv_tcp_client_select_value->read_buff));
            if(len<0){
                if (errno == EINTR){
                    goto again;
                }
            }
            if(len <= 0){
                close(socket_fd);
                connection_state=0;
                pv_tcp_client_select_value->connected=0;
            }else{//接收到客户端消息
                pv_tcp_client_select_value->read_buff[len]=0;
                pv_tcp_client_select_value->CallBack(pv_tcp_client_select_value->read_buff, len);
                //调用回调函数
                read(socket_fd, pv_tcp_client_select_value->read_buff, len);
            }
        }
    }
}
vTaskDelete(NULL);
```

如果需要多个客户端

复制出来几份就OK了



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: [803-Air724UG模块\(4G全网通GPRS开发\)-Air724UG\(4G\)把采集的摄像头照片发送到FTP服务器](#)

posted on 2021-12-12 01:17 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

B



支持 Markdown

自动补全

提交评论

退出

[Ctrl+Enter快捷键提交]

【推荐】华为开发者专区，与开发者一起构建万物互联的智能世界

【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：

- 一次缓存雪崩的灾难复盘
- 如何在 ASP.NET Core 中构建轻量级服务
- 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) ?

最新新闻：

- 360发现全球汽车操作系统多个高危漏洞：获宝马和系统商双重致谢 (2021-12-12 00:21)
 - 商汤科技回应被美国“拉黑”：强烈反对 相关指控毫无根据 (2021-12-12 00:07)
 - 苹果Apple Music年度榜单引热议：20年了还是周杰伦霸榜年度最热歌曲 (2021-12-12 00:01)
 - 硅谷直播带货元年：巨头集体下场，搬运中国模式，分阵营厮杀 (2021-12-11 13:22)
 - 苹果正在为旗下古典音乐服务Primephonic开发新音乐应 (2021-12-11 11:46)
- » 更多新闻...

历史上的今天：

2017-12-12 基于阿里云的MQTT远程控制

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，加入群聊。