**3详细设计**

**一、（每次开机挂载：sudo mount -t vboxsf anna /home/esp8266/Share/demo）**

1. 在SDK目录下新建app目录，在SDK的examples目录下的smart\_config下所有文件拷贝到app下。
2. 修改app目录下的gen\_misc.sh这个shell脚本。来配置实际生成的工程参数，此处我修改工程路径，与此同时也要修改Makefile，因为分离了SDK和工程目录，所以要指定路径（笔记本有写详细的添加代码）。
3. 编译的过程是运行./gen\_misc.sh的shell脚本进行编译，编译过程中会让我们选择参数：

①是否检查目录？Y

②是否包含云端升级？包含后生成user1.bin还是user2.bin？选择user1.bin

③选择SPI的速率，选择40MHZ。

④选择SPI的模式，选择DOUT。DIO -> Dual IO 两倍的数据输入输出

⑤选择对应芯片的Flash大小，选择32Mbit，也就是4096KB

4、编译完成后，就在upgrade下生成user1.4096.new.4.bin文件，可烧写。

**二、**

1. 在SDK目录下新建helloworld目录，将app目录下的所有文件复制到helloworld里。
2. User\_main.c是SDK自带的boot的主函数入口，在这里可以实现helloworld任务：用sourceinsight打开user\_main.c,在函数user\_init里新建helloworld任务

详细代码如下：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : ATaskHelloworld

\* Description : Helloworld 任务

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskHelloworld( void \*pvParameters )

{

int iVariableExample = 0;

for(;;)//用到了FreeRTOS里的任务函数，

{

printf("hello world!\n");

vTaskDelay(100);//10ms一个周期，想延迟1s，所以10\*10（是单位换算）=100

}

vTaskDelete( NULL );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

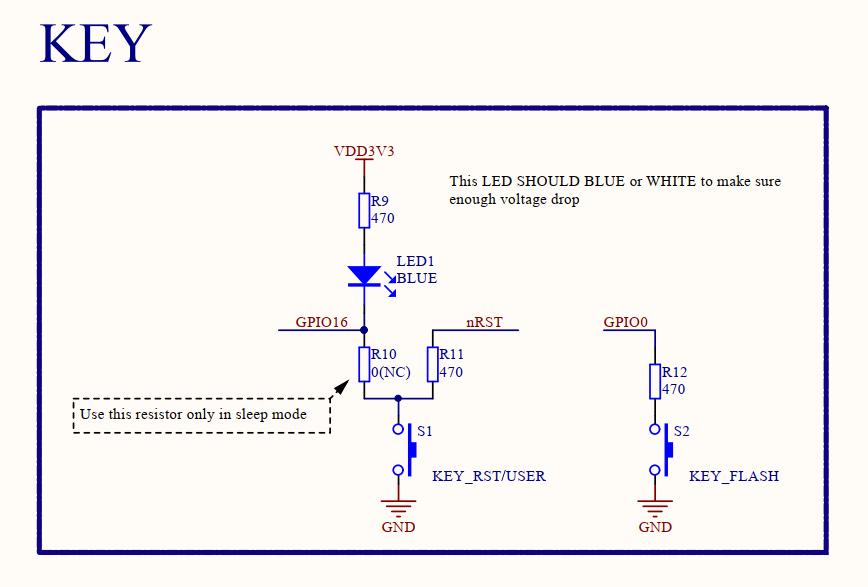
xTaskCreate(ATaskHelloworld, "hello world", 256, NULL, 2, NULL);

}

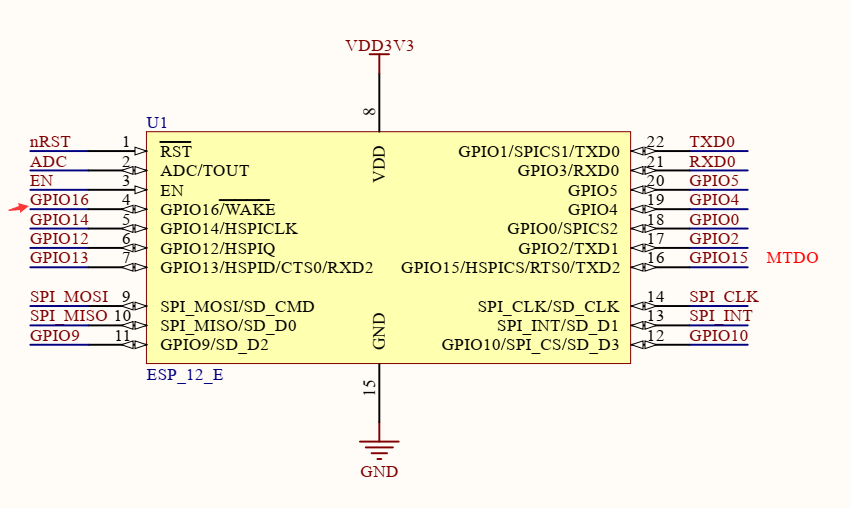
解释：

任务是由 C 语言函数实现的。唯一特别的只是任务的函数原型，其必须返回 void，  
而且带有一个 void 指针参数。其函数原型参见程序清单 1。  
void ATaskFunction( void \*pvParameters );  
程序清单 1 任务函数原型  
每个任务都是在自己权限范围内的一个小程序。其具有程序入口，通常会运行在一  
个死循环中，也不会退出。一个典型的任务结构如程序清单 2 所示。  
FreeRTOS 任务不允许以任何方式从实现函数中返回——它们绝不能有一  
条”return”语句，也不能执行到函数末尾。如果一个任务不再需要，可以显式地将其删  
除。这也在程序清单 2 展现。  
一个任务函数可以用来创建若干个任务——创建出的任务均是独立的执行实例，拥  
有属于自己的栈空间，以及属于自己的自动变量(栈变量)，即任务函数本身定义的变量。  
void ATaskFunction( void \*pvParameters )  
{  
/\* 可以像普通函数一样定义变量。用这个函数创建的每个任务实例都有一个属于自己的iVarialbleExample变  
量。但如果iVariableExample被定义为static，这一点则不成立 – 这种情况下只存在一个变量，所有的任务实  
例将会共享这个变量。 \*/  
int iVariableExample = 0;  
/\* 任务通常实现在一个死循环中。 \*/  
for( ;; )  
{  
/\* 完成任务功能的代码将放在这里。 \*/  
}  
/\* 如果任务的具体实现会跳出上面的死循环，则此任务必须在函数运行完之前删除。传入NULL参数表示删除  
的是当前任务 \*/  
vTaskDelete( NULL );  
}

1. 点灯任务
2. 分析ESP8266的原理图，找到KEY下的LED1的电路图



LED1连接到GPIO16上，当GPIO16拉低电平时，灯就亮，反之高电平则灭



1. 在SDK目录下新建led目录，将helloworld目录所有文件复制过去。
2. 因为GPIO是外设，所以在led目录下添加外设驱动文件，操作：在led目录下新建driver目录，复制SDK下的driver\_lib目录下的driver所有文件到新建的driver目录下，还要添加头文件，所以把driver\_lib目录下的include所有文件到led的include目录下。
3. 修改Makefile，因为新加了driver子目录。
4. 在user\_main.c中，添加Led的初始化

①头文件添加”gpio.h”

②配置GPIO16为输出模式

③设置GPIO16为高电平输出，一上电先让led灭

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : led\_init

\* Description : led初始化

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Led\_init( void)

{

gpio16\_output\_conf();//配置GPIO16为输出模式

gpio16\_output\_set(1);//此处1拉高电平，灭

}

1. Led任务的实现，for循环中，打开，延时，关闭，延时。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* FunctionName : ATaskLed

\* Description : Led 任务

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskLed( void \*pvParameters )

{

int iVariableExample = 0;

for(;;)

{

gpio16\_output\_set(0);//0低电平亮

printf("led in \n");

vTaskDelay(100);

gpio16\_output\_set(1);

printf("led off \n");

vTaskDelay(100);

}

vTaskDelete( NULL );

}

1. 在函数user\_init中，调用初始化好的Led函数和新建任务。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}

四、Station连接

1、Station是站点的意思，比如手机、平板配置为WIFI的Station模式，连接到本地网络，就是加入网络！

2、通过WIFI名称、密码就可以入网

3、入网流程：

配置WIFI为Station模式

动态内存分配，配置参数结构体

赋值SSID、PASSWD到参数中

设置参数到寄存器

释放内存

开启WIFI连接

4、在SDK目录下新建Station目录，把之前的led目录下所有文件复制过去

5、在user\_init下完成WIFI Station的功能开发

6、功能测试：用串口工具查看开机后是否被路由器分配了IP

#define SSID "CMCC-fX6F"

#define PASSWORLD "13977230634"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

wifi\_set\_opmode(STATION\_MODE);

struct station\_config \*config = (struct station\_config \*)\

zalloc(sizeof(struct station\_config));

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

wifi\_station\_set\_config(config);//设置到寄存器里才能对WIFI更改

free(config);

wifi\_station\_set\_auto\_connect(TRUE);//启用WIFI的自动连接

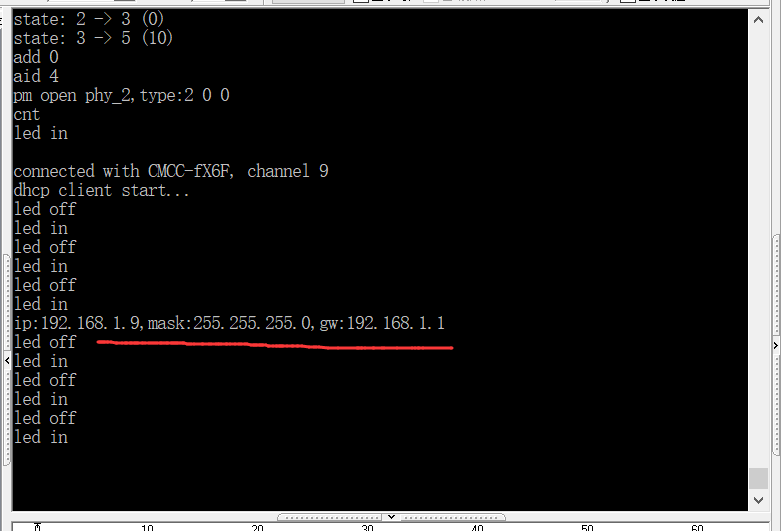
/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}



我的路由器网段是在1下

五、AP模式

1、配置WIFI为AP模式，就是别人来加入这个WIFI。

2、建立条件：SSID为自定义的WIFI名称

PASSWORD为自定义的WIFI密码

加密模式为WAP2\_PSK

1. 建立流程：

配置WIFI为AP

动态内存分配，配置参数结构体

获取当前参数到内存

赋值SSID和PASSWORD到参数中

配置加密模式为WAP2\_PSK

配置SSID长度为0（因为0的时候会在遇到字符串结尾的\0才计数，若传入长度不为0，就按那个规定长度来计算）

配置最大连接数（也就是上一个说的Station站点）为4

设置参数到寄存器

1. 在SDK下新建AP目录，把Station目录下所有文件复制过去。
2. 在user\_init下完成功能开发
3. 测试：电脑连接AP，获取AP分配的IP地址（注意，ESP8266默认的静态IP是192.168.4.1）

#define SSID "mine test WIFI"

#define PASSWORLD "88888888"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

wifi\_set\_opmode(SOFTAP\_MODE);

struct softap\_config \*config = (struct softap\_config \*)\

zalloc(sizeof(struct softap\_config));

wifi\_softap\_get\_config(config);//获取当前参数到内存

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

config->authmode = AUTH\_WPA\_WPA2\_PSK;

config->ssid\_len = 0;

config->max\_connection = 4;

wifi\_softap\_set\_config(config);

free(config);

//wifi\_station\_set\_auto\_connect(TRUE);

/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}

1. UDP的客户端Client编程
2. socket() 建立套接字

Sendto() 发送数据

Recvform() 接收数据

Close() 关闭套接字

1. 功能实现：

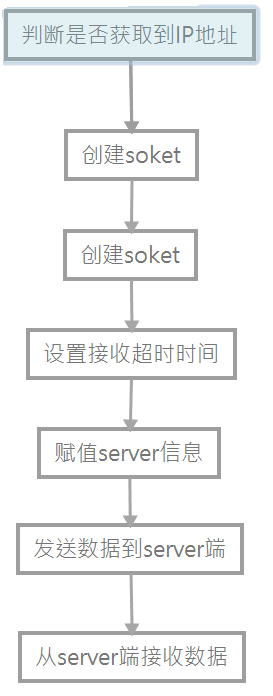
电脑模拟UDP的服务器端，指定IP、PORT，等待客户端的数据

客户端向服务器发送“I am client”

服务器收到后，向client发送“I am server”

1. 在SDK下建立udpclient目录，把之前的Station目录下的所有文件都复制过去
2. 在udpclient目录的user目录下创建udpclient.c文件，在include目录下创建udpclient.h文件
3. 代码实现：

在udpclient.c下，对udpclient初始化，实现任务，任务流程：



在udpclient.h中声明

Udpclient.c文件的代码：

#include "esp\_common.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "lwip/sockets.h"

#include "lwip/dns.h"

#include "lwip/netdb.h"

#include "udpclient.h"

#define SERVERADDR "192.168.1.6"

#define SERVERPORT 8000

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : ATaskUdpclient

\* Description : ATaskUdpclient 浠诲姟

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskUdpclient( void \*pvParameters )

{

int iVariableExample = 0;

int fd = -1;

int NetTimeOut = 5000;//超时参数

int ret;

struct sockaddr\_in ServerAddr;//定义结构体

char udpmsg[48];

STATION\_STATUS StaStatus;

do

{

StaStatus = wifi\_station\_get\_connect\_status();//接口函数

vTaskDelay(100);

}while(StaStatus != STATION\_GOT\_IP);//循环不停的检测是否获取WIFI

fd = socket(PF\_INET, SOCK\_DGRAM, 0);

if(fd == -1)

{

printf("get socket failed \n");

vTaskDelete(NULL);

}

setsockopt(fd, SOL\_SOCKET, SO\_RCVTIMEO, &NetTimeOut, sizeof(int));//设置超时接收时间

memset(&ServerAddr,0,sizeof(ServerAddr));//清空填充位

ServerAddr.sin\_family = AF\_INET;//IPV4

ServerAddr.sin\_addr.s\_addr = inet\_addr(SERVERADDR);//地址转换

ServerAddr.sin\_port = htons(SERVERPORT);//主机字节序到网络字节序的转换

ServerAddr.sin\_len = sizeof(ServerAddr);//长度

for(;;)//发送数据到server端

{

sendto(fd, "I am UpClient!", sizeof("I am UpClient!"),\

0, (struct sockaddr \*)&ServerAddr, (socklen\_t)ServerAddr.sin\_len);

do

{

ret = recvfrom(fd, udpmsg, 48,\

0, (struct sockaddr \*)&ServerAddr, (socklen\_t \*)(&ServerAddr.sin\_len));

if(ret > 0)//ret用来判断是否超时的返回值

{

printf("udpserver:%s\n",udpmsg);

}

else

{

printf("udpserver data is no\n");

}

}while(ret == -1);

}

vTaskDelete( NULL );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : Udpclient\_init

\* Description : Udpclient\_init 初始化任务

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Udpclient\_init( void )

{

xTaskCreate(ATaskUdpclient, "UdpClient", 256, NULL, 4, NULL);//新建任务 参数：任务函数名，传入的字符串，堆栈大小256，传入的参数为空，优先级为4，句柄为空。

}

Udpclient.h的代码：

#ifndef \_\_UDPCLIENT\_H\_\_

#define \_\_UDPCLIENT\_H\_\_

#ifdef \_\_cplusplus

extern "C" {

#endif

void ATaskUdpclient(void \* pvParameters);

void Udpclient\_init(void);

#ifdef \_\_cplusplus

}

#endif

#endif

User\_main.c文件的user\_init代码：

#define SSID "CMCC-fX6F"

#define PASSWORLD "13977230634"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

wifi\_set\_opmode(STATION\_MODE);

struct station\_config \*config = (struct station\_config \*)\

zalloc(sizeof(struct station\_config));

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

wifi\_station\_set\_config(config);

free(config);

wifi\_station\_set\_auto\_connect(TRUE);

Udpclient\_init();

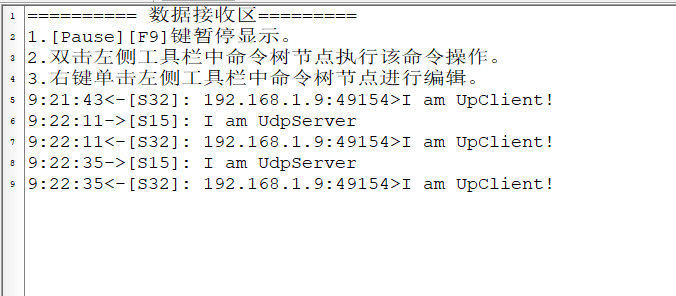
/\* need to set opmode before you set config \*/

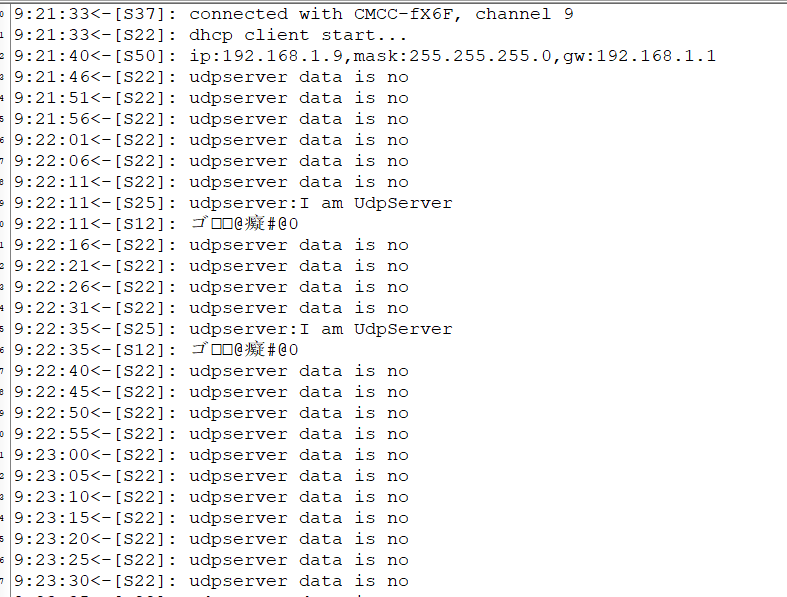
//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

//xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}



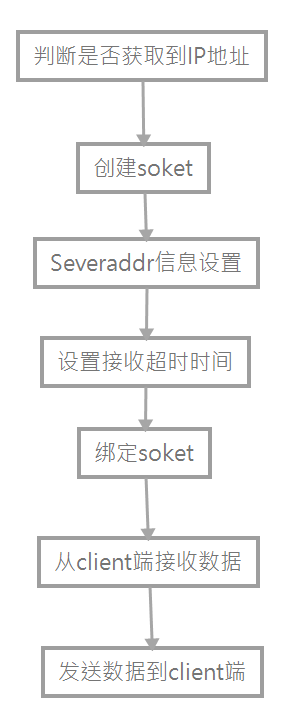


1. UDP的客户端Server编程
2. 完成UDP Server功能开发：

电脑模拟UDP Client，指定UDP Server(IP,PORT)，发送“I am Client”

Server收到数据后，向Client发送“I am Server”

1. 在SDK目录下新建udpserver目录，把udpclient目录下的所有文件复制过去，在user目录下新建udpserver.c文件，在include目录下新建udpserver.h文件
2. 编辑代码时，把udpclient的.c和.h的代码复制过去，并替换成udpserver，user\_main.c头文件包含”udpserver.h”
3. 在user\_main.c中添加“udpserver.h”,user\_init函数中添加UdpServer\_init();
4. 流程：



Udpserver.h代码：

#ifndef \_\_UDPSERVER\_H\_\_

#define \_\_UDPSERVER\_H\_\_

#ifdef \_\_cplusplus

extern "C" {

#endif

void ATaskUdpServer(void \* pvParameters);

void UdpServer\_init(void);

#ifdef \_\_cplusplus

}

#endif

#endif

Udpserver.c中代码：

#include "esp\_common.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "lwip/sockets.h"

#include "lwip/dns.h"

#include "lwip/netdb.h"

#include "udpserver.h"

#define SERVERADDR "192.168.1.6"

#define SERVERPORT 8000

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : ATaskUdpServer

\* Description : ATaskUdpServer 浠诲姟

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskUdpServer( void \*pvParameters )

{

int iVariableExample = 0;

int fd = -1;

int NetTimeOut = 5000;

int ret;

struct sockaddr\_in ServerAddr;

struct sockaddr from;

socklen\_t fromlen = sizeof(struct sockaddr);

char udpmsg[48];

STATION\_STATUS StaStatus;

do

{

StaStatus = wifi\_station\_get\_connect\_status();

vTaskDelay(100);

}while(StaStatus != STATION\_GOT\_IP);

fd = socket(PF\_INET, SOCK\_DGRAM, 0);

if(fd == -1)

{

printf("get socket failed \n");

vTaskDelete(NULL);

return;

}

setsockopt(fd, SOL\_SOCKET, SO\_RCVTIMEO, &NetTimeOut, sizeof(int));

memset(&ServerAddr,0,sizeof(ServerAddr));

ServerAddr.sin\_family = AF\_INET;

ServerAddr.sin\_addr.s\_addr = INADDR\_ANY;//**地址变为任何地址**

ServerAddr.sin\_port = htons(SERVERPORT);

ServerAddr.sin\_len = sizeof(ServerAddr);

if( bind(fd, (struct sockaddr\*)&ServerAddr,ServerAddr.sin\_len ) != 0)//**绑定函数**

{

printf("bind socket failed!\n");

vTaskDelete(NULL);

return;

}

for(;;)

{

do

{

ret = recvfrom(fd, udpmsg, 48,\

0, &from, (&fromlen));

if(ret > 0)

{

printf("udpserver:%s\n",udpmsg);

}

else

{

printf("udpserver data is no\n");

}

}while(ret == -1);

sendto(fd, "I am UpClient!", sizeof("I am UpClient!"),\

0, &from, fromlen);

}

vTaskDelete( NULL );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : UdpServer\_init

\* Description : UdpServer\_init 鍒濆鍖? \* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void UdpServer\_init( void )

{

xTaskCreate(ATaskUdpServer, "UdpServer", 256, NULL, 4, NULL);

}

User\_main.c的user\_init函数代码：

#define SSID "CMCC-fX6F"

#define PASSWORLD "13977230634"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

wifi\_set\_opmode(STATION\_MODE);

struct station\_config \*config = (struct station\_config \*)\

zalloc(sizeof(struct station\_config));

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

wifi\_station\_set\_config(config);

free(config);

wifi\_station\_set\_auto\_connect(TRUE);

UdpServer\_init();

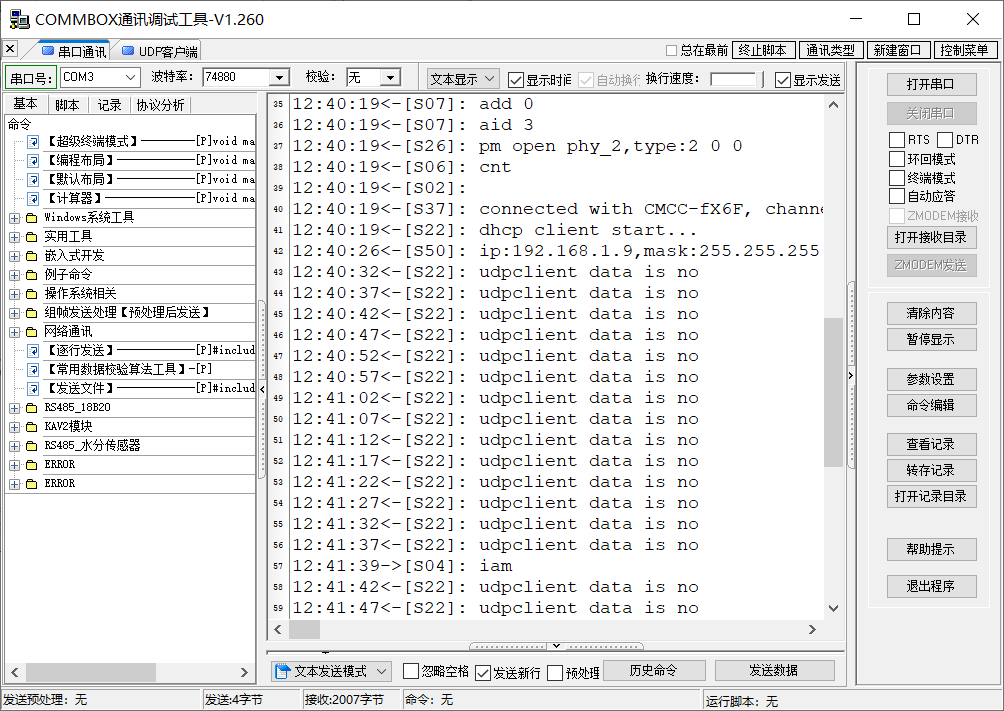
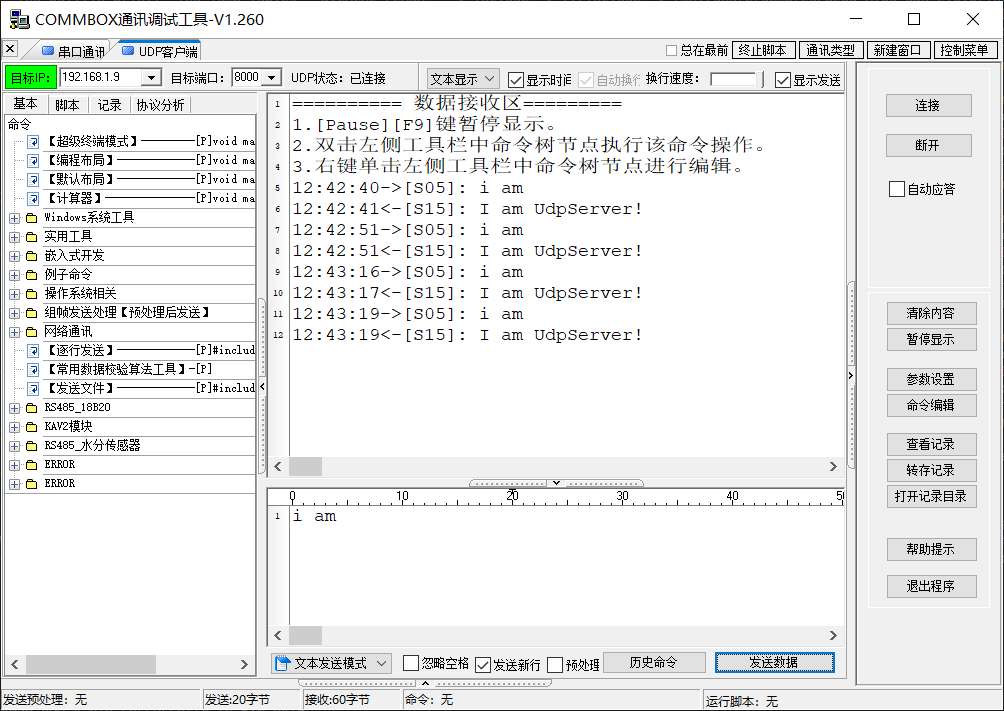
/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

//xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}



1. TCP的client实现

1、完成TCP Client功能开发：

电脑模拟TCP Client，指定TCP Server(IP,PORT)，发送“I am Client”

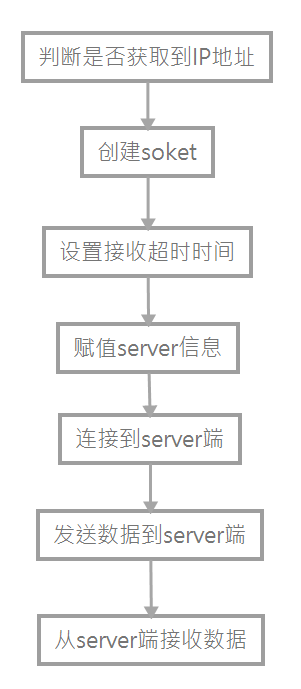
Server收到数据后，向Client发送“I am Server”

2、在SDK目录下新建tcpclient目录，把udpclient目录下的所有文件复制过去在user目录下新建tcpclient.c文件，在include目录下新建tcpclient.h文件

3、编辑代码时，把udpclient的.c和.h的代码复制过去，并替换成tcpclient，user\_main.c头文件包含”tcpclient.h”

4、在user\_main.c中添加“tcpclient.h”,user\_init函数中添加TcpClient\_init()

5、流程：



Tcpclient.h代码：

#ifndef \_\_TCPCLIENT\_H\_\_

#define \_\_TCPCLIENT\_H\_\_

#ifdef \_\_cplusplus

extern "C" {

#endif

void ATaskTcpClient(void \* pvParameters);

void TcpClient\_init(void);

#ifdef \_\_cplusplus

}

#endif

#endif

Tcpclient.c代码：

#include "esp\_common.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "lwip/sockets.h"

#include "lwip/dns.h"

#include "lwip/netdb.h"

#include "tcpclient.h"

#define SERVERADDR "192.168.1.6"

#define SERVERPORT 8000

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : ATaskTcpClient

\* Description : ATaskTcpClient 任务

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskTcpClient( void \*pvParameters )

{

int iVariabSleExample = 0;

int fd = -1;

int NetTimeOut = 5000;

int ret;

struct sockaddr\_in ServerAddr;

char udpmsg[48];

STATION\_STATUS StaStatus;

do

{

StaStatus = wifi\_station\_get\_connect\_status();

vTaskDelay(100);

}while(StaStatus != STATION\_GOT\_IP);

fd = socket(PF\_INET, SOCK\_STREAM, 0);

if(fd == -1)

{

printf("get socket failed \n");

vTaskDelete(NULL);

}

setsockopt(fd, SOL\_SOCKET, SO\_RCVTIMEO, &NetTimeOut, sizeof(int));

memset(&ServerAddr,0,sizeof(ServerAddr));

ServerAddr.sin\_family = AF\_INET;

ServerAddr.sin\_addr.s\_addr = inet\_addr(SERVERADDR);

ServerAddr.sin\_port = htons(SERVERPORT);

ServerAddr.sin\_len = sizeof(ServerAddr);

do

{

ret = connect(fd,(struct sockaddr \*)&ServerAddr, ServerAddr.sin\_len);

if(ret != 1)

{

printf("连接失败\n");

vTaskDelay(100);

}

}

while(ret != 0);

for(;;)

{

send(fd, "I am TcpClient!", sizeof("I am TcpClient!"),0);

do

{

ret = recv(fd, udpmsg, 48,0);

if(ret > 0)

{

printf("Tcpserver:%s\n",udpmsg);

}

else

{

printf("Tcpserver data is no\n");

}

}while(ret == -1);

}

vTaskDelete( NULL );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : TcpClient\_init

\* Description : TcpClient\_init 初始化

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TcpClient\_init( void )

{

xTaskCreate(ATaskTcpClient, "TcpClient", 256, NULL, 4, NULL);

}

User\_main.c的user\_init函数的代码：

#define SSID "CMCC-fX6F"

#define PASSWORLD "13977230634"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

wifi\_set\_opmode(STATION\_MODE);

struct station\_config \*config = (struct station\_config \*)\

zalloc(sizeof(struct station\_config));

//wifi\_softap\_get\_config( config);

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

//config->authmode = AUTH\_WPA\_WPA2\_PSK;

//config->max\_connection = 4;

wifi\_station\_set\_config(config);

free(config);

wifi\_station\_set\_auto\_connect(TRUE);

TcpClient\_init();

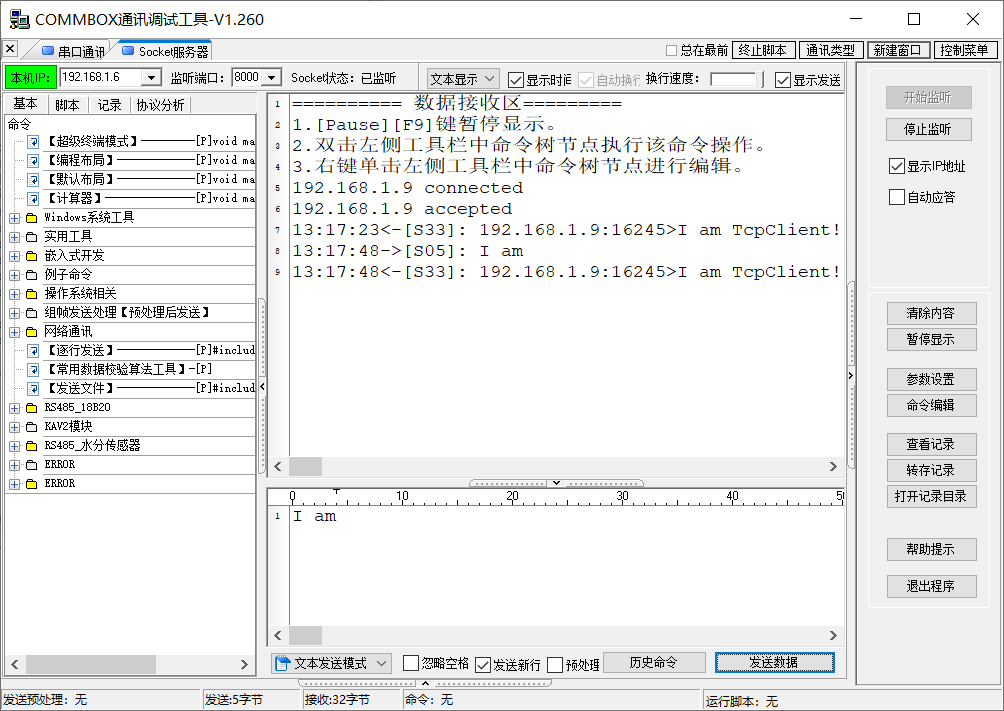
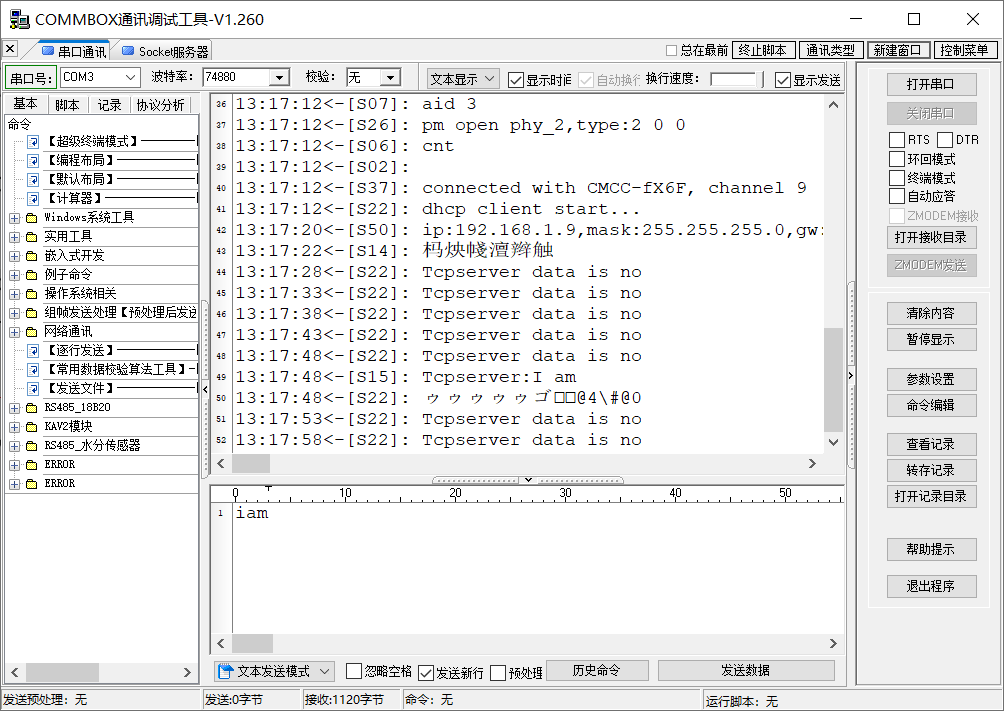
/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

//xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}



1. TCP的Server实现

1、完成TCP Server功能开发：

电脑模拟TCP Server，指定TCP Server(IP,PORT)进行连接，连接成功后发送“I am Server”

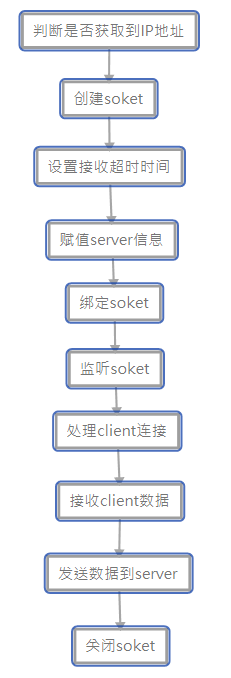
Server收到数据后，向Client发送“I am Server”

2、在SDK目录下新建tcpserver目录，把tcpclient目录下的所有文件复制过去在user目录下新建tcpserver.c文件，在include目录下新建tcpserver.h文件

3、编辑代码时，把tcpclient的.c和.h的代码复制过去，并替换成tcpserver，user\_main.c头文件包含”tcpserver.h”

4、在user\_main.c中添加“tcpserver.h”,user\_init函数中添加TcpServer\_init()

5、流程：



Tcpserver.h文件的代码：

#ifndef \_\_TCPSERVER\_H\_\_

#define \_\_TCPSERVER\_H\_\_

#ifdef \_\_cplusplus

extern "C" {

#endif

void ATaskTcpServer(void \* pvParameters);

void TcpServer\_init(void);

#ifdef \_\_cplusplus

}

#endif

#endif

Tcpserver.c文件的代码：

#include "esp\_common.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "lwip/sockets.h"

#include "lwip/dns.h"

#include "lwip/netdb.h"

#include "tcpserver.h"

#define SERVERADDR "192.168.1.6"

#define SERVERPORT 8000

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : ATaskTcpServer

\* Description : ATaskTcpServer 任务

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskTcpServer( void \*pvParameters )

{

int iVariabSleExample = 0;

int fd = -1;

int NetTimeOut = 20000;

int ret;

int cfd = -1;

struct sockaddr\_in ServerAddr;

struct sockaddr ClientAddr;

socklen\_t ClientAddrlen = sizeof(struct sockaddr);

char Tcpmsg[48];

STATION\_STATUS StaStatus;

do

{

StaStatus = wifi\_station\_get\_connect\_status();

vTaskDelay(100);

}while(StaStatus != STATION\_GOT\_IP);

fd = socket(PF\_INET, SOCK\_STREAM, 0);

if(fd == -1)

{

printf("get socket failed \n");

vTaskDelete(NULL);

return;

}

setsockopt(fd, SOL\_SOCKET, SO\_RCVTIMEO, &NetTimeOut, sizeof(int));

memset(&ServerAddr,0,sizeof(ServerAddr));

ServerAddr.sin\_family = AF\_INET;

ServerAddr.sin\_addr.s\_addr = INADDR\_ANY;

ServerAddr.sin\_port = htons(SERVERPORT);

ServerAddr.sin\_len = sizeof(ServerAddr);

if(bind(fd, (struct sockaddr\*)&ServerAddr, ServerAddr.sin\_len) != 0)

{

printf("get bind failed \n");

vTaskDelete(NULL);

return;

}

if(listen(fd, 5) != 0)

{

printf("listen failed \n");

vTaskDelete(NULL);

return;

}

for(;;)

{

cfd = accept(fd, &ClientAddr, &ClientAddrlen);

if(cfd != -1)

{

ret = recv(cfd, Tcpmsg, 48,0);

if(ret > 0)

{

printf("TcpClient:%s\n",Tcpmsg);

send(cfd, "I am TcpServer!", sizeof("I am TcpServer!"),0);

}

else

{

printf("Tcpserver data is no\n");

}

}

close(cfd);

}

vTaskDelete( NULL );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : TcpServer\_init

\* Description : TcpServer\_init 初始化

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void TcpServer\_init( void )

{

xTaskCreate(ATaskTcpServer, "TcpServer", 256, NULL, 4, NULL);

}

User\_main.c的user\_init函数的代码：

#define SSID "CMCC-fX6F"

#define PASSWORLD "13977230634"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

wifi\_set\_opmode(STATION\_MODE);

struct station\_config \*config = (struct station\_config \*)\

zalloc(sizeof(struct station\_config));

//wifi\_softap\_get\_config( config);

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

//config->authmode = AUTH\_WPA\_WPA2\_PSK;

//config->max\_connection = 4;

wifi\_station\_set\_config(config);

free(config);

wifi\_station\_set\_auto\_connect(TRUE);

TcpServer\_init();

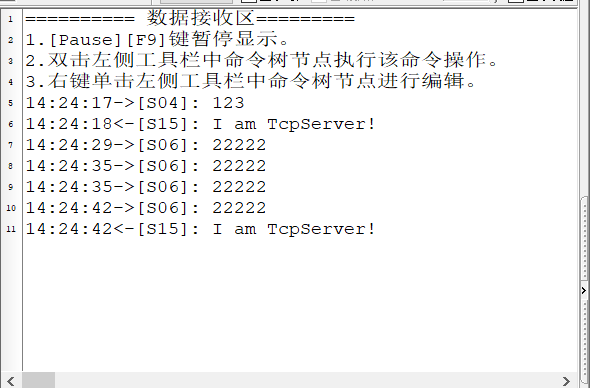
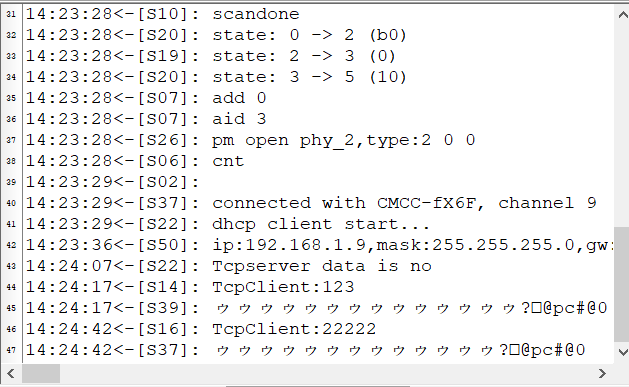
/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

//xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}

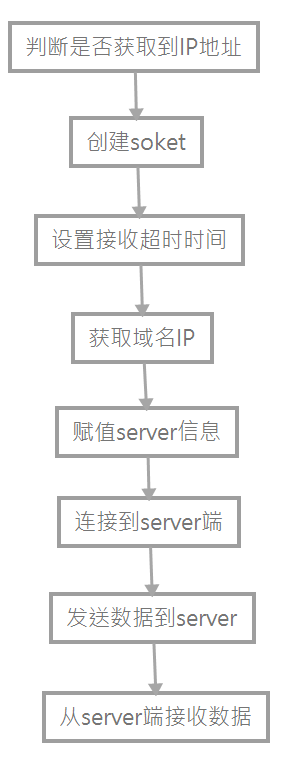


1. WIFI HTTP client实现
2. 因为最终呈现给用户的是web网页版，所有要会这个
3. 在SDK目录下新建httpclient目录，把tcpclient目录下的所有文件复制过去，在user目录下新建httpclient.c文件，在include目录下新建httpclient.h文件

3、编辑代码时，把tcpclient的.c和.h的代码复制过去，并替换成httpserver，user\_main.c头文件包含”httpserver.h”

4、在user\_main.c中添加“httpserver.h”,user\_init函数中添加HttpServer\_init()

5、流程：



1. 代码

**Httpclient.h文件代码：**

#ifndef \_\_HTTPCLIENT\_H\_\_

#define \_\_HTTPCLIENT\_H\_\_

#ifdef \_\_cplusplus

extern "C" {

#endif

void ATaskHttpClient(void \* pvParameters);

void HttpClient\_init(void);

#ifdef \_\_cplusplus

}

#endif

#endif

**Httpclient.c文件代码：**

#include "esp\_common.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "lwip/sockets.h"

#include "lwip/dns.h"

#include "lwip/netdb.h"

#include "httpclient.h"

#define SERVERADDR "192.168.1.6"

#define SERVERPORT 8000

const char GetStr[] = "GET /index.html HTTP/1.1\r\n"

"Host: www.baidu.com\r\n"

"\r\n";

#define HTTP\_RCV\_LEN 50000

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : ATaskHttpClient

\* Description : ATaskHttpClient 浠诲姟

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskHttpClient( void \*pvParameters )

{

int iVariabSleExample = 0;

int fd = -1;

int NetTimeOut = 20000;

int ret;

int i;

struct hostent \*phostent;

struct sockaddr\_in ServerAddr;

char \*Httpmsg;

char \*ipaddr = NULL;

STATION\_STATUS StaStatus;

do

{

StaStatus = wifi\_station\_get\_connect\_status();

vTaskDelay(100);

}while(StaStatus != STATION\_GOT\_IP);

fd = socket(PF\_INET, SOCK\_STREAM, 0);

if(fd == -1)

{

printf("get socket failed \n");

vTaskDelete(NULL);

return;

}

setsockopt(fd, SOL\_SOCKET, SO\_RCVTIMEO, &NetTimeOut, sizeof(int));

phostent = gethostbyname("www.baidu.com");//閫氳繃鍩熷悕鑾峰彇鍦板潃

if(phostent == NULL)

{

printf("get host failed \n");

vTaskDelete(NULL);

return;

}

else

{

for(i = 0;phostent->h\_addr\_list[i];i++)

{

ipaddr = inet\_ntoa(\*(struct in\_addr\*)(phostent->h\_addr\_list[i]));

if(ipaddr != NULL)

{

printf("host addr is:%s\n",ipaddr);

break;

}

}

if(ipaddr == NULL)

{

printf("get ip failed \n");

vTaskDelete(NULL);

return;

}

}

memset(&ServerAddr, 0, sizeof(ServerAddr));

ServerAddr.sin\_family = AF\_INET;

ServerAddr.sin\_addr.s\_addr = inet\_addr(ipaddr);

ServerAddr.sin\_port = htons(80);

ServerAddr.sin\_len = sizeof(ServerAddr);

do

{

ret = connect(fd,(struct sockaddr\*)&ServerAddr,ServerAddr.sin\_len);

if(ret != 0)

{

printf("connect failed \n");

vTaskDelete(NULL);

}

}

while(ret != 0);

Httpmsg = zalloc(HTTP\_RCV\_LEN);

send(fd,GetStr,strlen(GetStr),0);

for(;;)

{

do

{

ret = recv(fd,Httpmsg,50000,0);

if(ret > 0)

{

printf("%s",Httpmsg);

}

else

{

printf("HttpServer data is no!\n");

}

}while(ret == -1);

vTaskDelete( NULL );

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : TcpServer\_init

\* Description : TcpServer\_init 鍒濆鍖?

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void HttpClient\_init( void )

{

xTaskCreate(ATaskHttpClient, "HttpClient", 256, NULL, 4, NULL);

}

**User\_main.c的user\_init函数代码：**

#define SSID "CMCC-fX6F"

#define PASSWORLD "13977230634"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : user\_init

\* Description : entry of user application, init user function here

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ICACHE\_FLASH\_ATTR

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

Led\_init();

wifi\_set\_opmode(STATION\_MODE);

struct station\_config \*config = (struct station\_config \*)\

zalloc(sizeof(struct station\_config));

//wifi\_softap\_get\_config( config);

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

//config->authmode = AUTH\_WPA\_WPA2\_PSK;

//config->max\_connection = 4;

wifi\_station\_set\_config(config);

free(config);

wifi\_station\_set\_auto\_connect(TRUE);

HttpClient\_init();

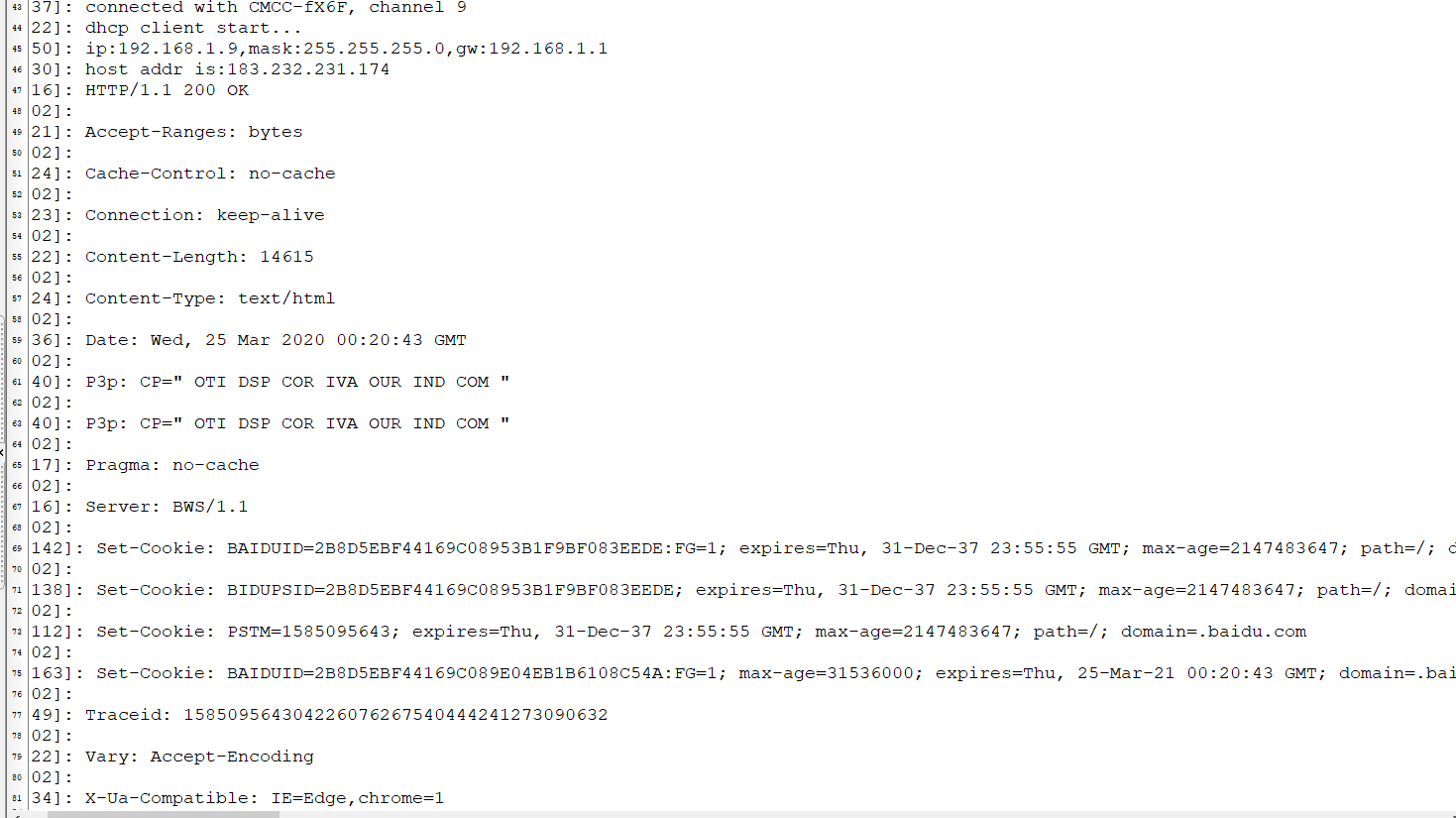
/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

//xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}

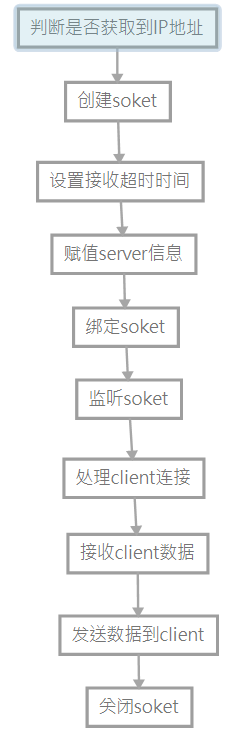




1. WIFI HTTP server实现
2. 功能分析：模拟httpServer等待网页访问

电脑访问网址，查看网页信息

1. 在SDK目录下新建httpserver目录，把tcpserver目录下的所有文件复制过去，在user目录下新建httpserver.c文件，在include目录下新建httpserver.h文件
2. 流程：



1. 代码：

**Httpserver.h的代码：**

#ifndef \_\_HTTPSERVER\_H\_\_

#define \_\_HTTPSERVER\_H\_\_

#ifdef \_\_cplusplus

extern "C" {

#endif

void ATaskHttpServer(void \* pvParameters);

void HttpServer\_init(void);

#ifdef \_\_cplusplus

}

#endif

#endif

**Httpserver.c的代码：**

#include "esp\_common.h"

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "lwip/sockets.h"

#include "lwip/dns.h"

#include "lwip/netdb.h"

#include "httpserver.h"

#define SERVERADDR "192.168.1.6"

#define SERVERPORT 80

const char \*DefaultPage=

"<html>"

"<head>"

"<meta http-equiv=\"Content-Type\" content=\"text/html; charset=utf-8\" />"

"<title>你好呀！</title>"

"</head>"

"<body>"

"<P>这是我的实验!</P>"

"</body>"

"</html>";

int file\_ok(int cfd,long flen)

{

char \*send\_buf = zalloc(sizeof(char)\*100);

sprintf(send\_buf,"HTTP/1.1 200 OK\r\n");

send(cfd,send\_buf,strlen(send\_buf),0);

sprintf(send\_buf,"Connection: keep-alive\r\n");

send(cfd,send\_buf,strlen(send\_buf),0);

sprintf(send\_buf,"Content-Length: %ld\r\n",flen);

send(cfd,send\_buf,strlen(send\_buf),0);

sprintf(send\_buf,"Content-Type: text/html\r\n");

send(cfd,send\_buf,strlen(send\_buf),0);

sprintf(send\_buf,"\r\n");

send(cfd,send\_buf,strlen(send\_buf),0);

free(send\_buf);

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : ATaskHttpServer

\* Description : ATaskHttpServer 任务

\* Parameters : none

\* Returns : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ATaskHttpServer( void \*pvParameters )

{

int iVariabSleExample = 0;

int fd = -1;

int NetTimeOut = 20000;

int ret;

int cfd = -1;

struct sockaddr\_in ServerAddr;

struct sockaddr ClientAddr;

socklen\_t ClientAddrlen = sizeof(struct sockaddr);

char \*Httpmsg;

char \*Sendmsg;

STATION\_STATUS StaStatus;

do

{

StaStatus = wifi\_station\_get\_connect\_status();

vTaskDelay(100);

}while(StaStatus != STATION\_GOT\_IP);

fd = socket(PF\_INET, SOCK\_STREAM, 0);

if(fd == -1)

{

printf("get socket failed \n");

vTaskDelete(NULL);

return;

}

setsockopt(fd, SOL\_SOCKET, SO\_RCVTIMEO, &NetTimeOut, sizeof(int));

memset(&ServerAddr,0,sizeof(ServerAddr));

ServerAddr.sin\_family = AF\_INET;

ServerAddr.sin\_addr.s\_addr = INADDR\_ANY;

ServerAddr.sin\_port = htons(SERVERPORT);

ServerAddr.sin\_len = sizeof(ServerAddr);

if(bind(fd, (struct sockaddr\*)&ServerAddr, ServerAddr.sin\_len) != 0)

{

printf("get bind failed \n");

vTaskDelete(NULL);

return;

}

if(listen(fd, 5) != 0)

{

printf("listen failed \n");

vTaskDelete(NULL);

return;

}

Httpmsg = (char\*)zalloc(sizeof(char)\*1000);

for(;;)

{

cfd = accept(fd, &ClientAddr, &ClientAddrlen);

if(cfd != -1)

{

printf("HttpClient accept\n");

ret = recv(cfd, Httpmsg, 1000,0);

if(ret > 0)

{

printf("HttpClient recv\n");

printf("HttpClient:%s\n",Httpmsg);

file\_ok(cfd,strlen(DefaultPage));

send(cfd, DefaultPage, strlen(DefaultPage),0);

}

else

{

printf("HttpClient data is no\n");

}

}

close(cfd);

}

vTaskDelete( NULL );

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FunctionName : HttpServer\_init

\* Description : HttpServer\_init 初始化

\* Parameters : none

\* Returns : none

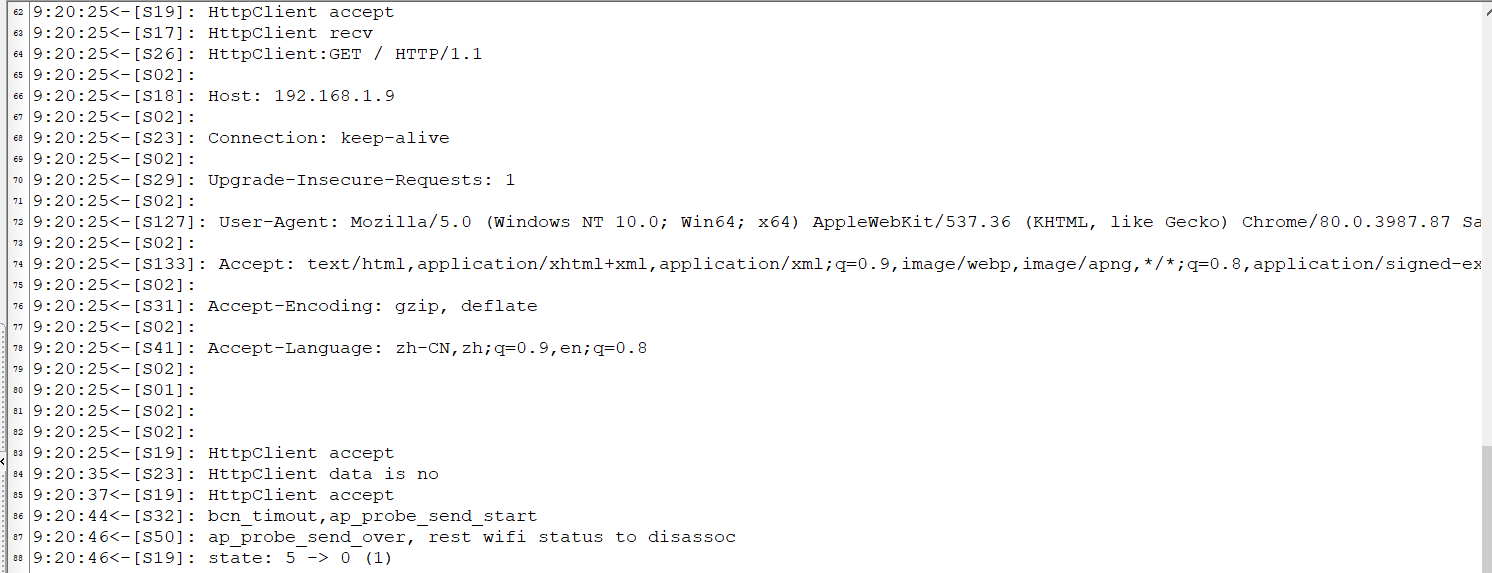
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

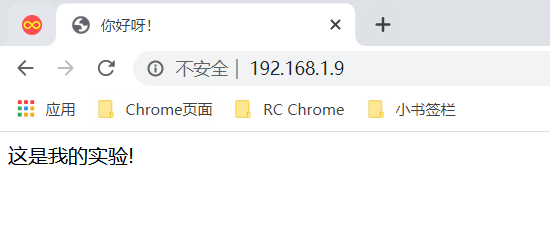
void HttpServer\_init( void )

{

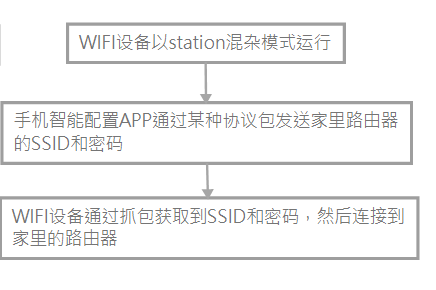
xTaskCreate(ATaskHttpServer, "HttpServer", 256, NULL, 4, NULL);

}





1. Smartconfig(用户端输入密码名称自己联网)
2. 一键配置网络，用户端输入密码名称后就可以一键联网。如智能插座在一键配置后就可以连接网络。
3. 原理：手机通过UDP广播，将AP的相关信息组帧发出，而WIFI模块一直处于UDP监听状态，获取到AP信息后，WIFI模块就可以直接加入AP。（AP是别人加入它，现在AP都有加密模式）



Airkiss是微信专门为物联网配网开发的，是对Smartconfig的进一步封装

1. 功能实现：

在user\_main.c文件的user\_init函数下，添加smart\_config任务

代码：

user\_init(void)

{

printf("SDK version:%s\n", system\_get\_sdk\_version());

//Led\_init();

wifi\_set\_opmode(STATION\_MODE);

xTaskCreate(smartconfig\_task, "smartconfig",256, NULL, 5, NULL);

/\*

struct station\_config \*config = (struct station\_config \*)\

zalloc(sizeof(struct station\_config));

//wifi\_softap\_get\_config( config);

sprintf(config->ssid, SSID);

sprintf(config->password, PASSWORLD);

//config->authmode = AUTH\_WPA\_WPA2\_PSK;

//config->max\_connection = 4;

wifi\_station\_set\_config(config);

free(config);

wifi\_station\_set\_auto\_connect(TRUE);

\*/

HttpServer\_init();

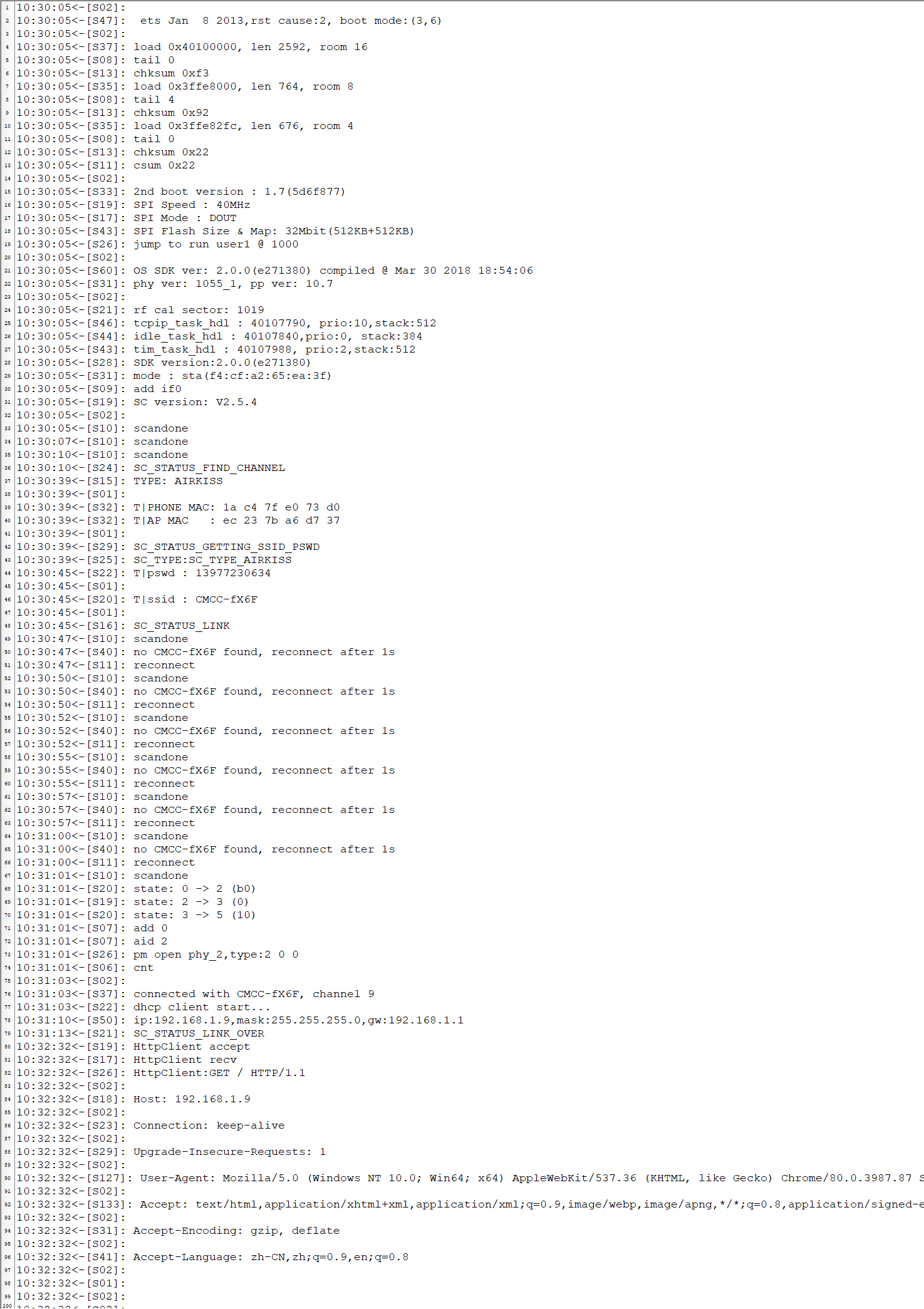
/\* need to set opmode before you set config \*/

//wifi\_set\_opmode(STATION\_MODE);

//xTaskCreate(smartconfig\_task, "smartconfig\_task", 256, NULL, 2, NULL);

//xTaskCreate(ATaskLed, "LED", 256, NULL, 2, NULL);

}



1. SNTP通信协议（实时时钟）
2. 简单的网络同步时钟协议，通过网络来获取时间，从而进行内部时钟的校正
3. NTP：是网络时间协议，用来同步网络设备的时间。
4. SNTP：
5. 是由NTP改编而来，大大简化了NTP协议，同时也能保证时间达到一定的精确度。
6. SNTP主要应用于同步因特网上计算机的时间。
7. SNTP协议采用客户端/服务器的工作方式，可以采用单播（点对点）或广播（一对多）模式操作。
8. 单播模式下，SNTP客户端能通过定期访问SNTP服务器获取准确时间，用于调整客户端自身所在系统的时间，达到同步时钟的目的。
9. 广播模式下，SNTP服务器周期性的发送消息给指定的IP广播地址或IP多播地址，SNTP客户端则通过监听这些地址来获取时间信息。
10. SNTP服务器通过接收GPS信号或自带的原子钟作为系统的时间基准。
11. NTP服务器列表：
12. 最常见的国内地址：cn.pool.ntp.org
13. windows系统自带的俩：time.windows和time.nist.gov
14. 阿里云的：ntp1.aliyun.com(1-7都可以用)
15. SNTP接口说明：
16. sntp\_setserver

功能：通过IP地址设置SNTP服务器，一共最多支持设置3个SNTP服务器

函数定义：void sntp\_setserver(unsigned char idx,ip\_addr\_t \*addr)

参数①：SNTP的服务器编号，最多支持三个SNTP服务器(0-2),0号为主服务器，1、2号为备用服务器。

参数②：IP地址，用户需要自行确保，传入的是合法SNTP服务器。

返回值：无

1. sntp\_setservername

功能：通过域名设置SNTP服务器，一共最对支持3个SNTP服务器

函数定义：void sntp\_setservername(unsigned char idx,char \*server)

参数①SNTP服务器编号，最多支持三个SNTP服务器(0-2),0号为主服务器，1、2号为备用服务器。

参数②：IP地址，用户需要自行确保，传入的是合法SNTP服务器。

返回值：无

（3）sntp\_init

功能：SNTP的初始化

函数定义：void sntp\_init(void)

参数：无

返回值：无

（4）sntp\_stop

功能：SNTP关闭

函数定义：void sntp\_stop(void)

参数：无

返回值：无

（5）sntp\_get\_current\_timestamp

功能：查询当前距离基准时间（1970.01.01 00:00:00 GMT+8）的时间戳，单位：秒

函数定义：uint32 sntp\_current\_timestamp()

参数：无

返回值：距离基准时间的时间戳

（6）sntp\_get\_real\_time

功能：查询实际时间（GMT+8）

函数定义：char\* sntp\_get\_real\_time(long t)

参数：与基准时间相距的时间戳

返回值：实际时间

1. FOTA远程固件升级（有bug时修复）