SoftAP wifi config demo reference design

Version: 1.0

Release date: 16 Nov. 2016

Document Revision History

Revision	Date	Description
1.0	16 Nov. 2016	MTK SoftAP wifi config, based on SDK3.3.1.

Table of contents

1. Introduction

1.1. Objective

The purpose of this document is to describe the design of MTK SoftAP wifi config.

In this document, you can configure the wifi sta on the MT76x7, to connect the specific Router. Because the device based on MT76x7 have not the UI screen,the end user need to operate a app, user can input the ssid & psk string,then the data will be transfer to device.Finally,the device will connect the specific Router.

In this document, SW design and SW coding should be introduced for reader. And reader could build the SoftAP demo in a short time.

2. Software design

2.1. Component

MCU: MediaTek MT7687 or MT7697

WIFI: embedded in MediaTek MT7687 or MT7697 (WIFI SOC)

Router:Wifi Router

PC: Run the config app, to input the ssid & psk

2.2. Introduction

a.Device launch wifi by AP mode(ssid: MTK_DEMO_SOFT_AP), and set open auth;

b.User scan & connect the wifi AP by PC;

c.Start the PC app to input the SSID & PSK, the data will be transferred to device;

d.Device switch the opmode from AP to STA, then connect to the Router.

2.3. Architecture & Flow

Figure 1. architecture & flow

3. Software

3.1. MT76x7 device code

Device launch wifi by AP mode(ssid: MTK_DEMO_SOFT_AP), and set open auth.DHCPD will start.

```
int mtk_demo_run(void)
{
   char ssid[32 + 1] = {0};
   char passwd[64 + 1] = {0};
   char bssid[6];
   char auth;
   char encry;
   char channel;

   int ret;

   printf("mtk_demo_run() start... \n");

   if(1)
   {
      //printf("mtk demo wireless setup timeout!\n");

      if(mtk_demo_config_softap(ssid, passwd) < 0){
         printf("mtk demo softap timeout!\n");
         ret = -1;
         goto out;
      }
   }

   printf("get the ssid is:%s, passwd is:%s\n", ssid, passwd);//get ssid & passwd OK;

 #if 1
   if(mtk_demo_connect_ap((unsigned char *)ssid, (unsigned char *)passwd) < 0)
            {
      printf("%s connect failed\n", ssid);
      ret = -1;
      goto out;
   }
 #else
   xTaskCreate(mtk_demo_connect_ap,"mtk_demo_connect_ap",1024,NULL,1,NULL);
   printf("task create->mtk_demo_connect_ap. \n");
 #endif

   printf("wifi connected\n");
```

```
    ret = 0;
out:

    return ret;
}

int mtk_demo_config_softap(char *ssid_buf, char *passwd_buf)
{
    int ret = 0;
            /* prepare and setup softap */
            mtk_demo_softap_setup();

            /* tcp server to get ssid & passwd */
            ret = mtk_demo_softap_tcp_server();

    strncpy(ssid_buf, g_mtk_demo_softap_ssid, strlen(g_mtk_demo_softap_ssid));
    ssid_buf[strlen(g_mtk_demo_softap_ssid)] = 0;
    strncpy(passwd_buf, g_mtk_demo_softap_passwd, strlen(g_mtk_demo_softap_passwd));
    passwd_buf[strlen(g_mtk_demo_softap_passwd)] = 0;

    mtk_demo_softap_exit();

            return ret;
}

void mtk_demo_softap_setup(void)
{
            /*
             * wilress params: 11BGN
             * channel: auto, or 1, 6, 11
             * authentication: OPEN
             * encryption: NONE
             * gatewayip: 172.31.254.250, netmask: 255.255.255.0
             * DNS server: 172.31.254.250.       IMPORTANT!!!  ios depend on it!
             * DHCP: enable
             * SSID: 32 ascii char at most
             * softap timeout: 5min
             */

            char ssid[WIFI_MAX_LENGTH_OF_SSID + 1] = {0};
            //ssid: max 32Bytes(excluding '\0')
            strncpy(ssid, "MTK_DEMO_SOFT_AP", sizeof(ssid));
            printf("[mtk demo]: set_ap_config() ssid_name =%s\n", ssid);

            while(!g_supplicant_ready){
                        vTaskDelay(20);
            }
```

```
  printf("[mtk demo]: g_supplicant_ready is true, Ready. \n");

        wifi_config_set_opmode(WIFI_MODE_AP_ONLY);
        wifi_config_set_security_mode(WIFI_PORT_AP, WIFI_AUTH_MODE_OPEN,
WIFI_ENCRYPT_TYPE_WEP_DISABLED);
        wifi_config_set_channel(WIFI_PORT_AP, 6);
        wifi_config_set_ssid(WIFI_PORT_AP, (uint8_t *)ssid, strlen(ssid));
        wifi_config_reload_setting();

        printf("[mtk demo]: Set softAP done. \n");

        while(!g_supplicant_ready){
                vTaskDelay(20);
        }

        //set AP gateway IP:
        {
                char ip_buf[] = "172.31.254.250";
                char mask_buf[] = "255.255.255.0";
                char start_ip[] = "172.31.254.251";
                char end_ip[] = "172.31.254.254";
                char primary_dns[] = "172.31.254.250";
                char secondary_dns[] = "8.8.4.4";
                struct ip4_addr addr;
                struct netif *sta_if;
                struct netif *ap_if;
                ap_if = netif_find_by_type(NETIF_TYPE_AP);
                sta_if = netif_find_by_type(NETIF_TYPE_STA);

                netif_set_status_callback(ap_if, NULL);

                inet_aton(mask_buf, &addr);
                netif_set_netmask(ap_if, &addr);
                inet_aton(ip_buf, &addr);
                netif_set_ipaddr(ap_if, &addr);
                netif_set_gw(ap_if, &addr);//gatewayip: 172.31.254.250

                dhcp_stop(sta_if);
                netif_set_link_up(ap_if);
                netif_set_default(ap_if);
                dhcpd_settings_t dhcpd_settings;
                memset(&dhcpd_settings, 0, sizeof(dhcpd_settings_t));
                strcpy((char *)dhcpd_settings.dhcpd_server_address, ip_buf);
                strcpy((char *)dhcpd_settings.dhcpd_netmask, mask_buf);
                strcpy((char *)dhcpd_settings.dhcpd_gateway, ip_buf);
                strcpy((char *)dhcpd_settings.dhcpd_primary_dns, primary_dns);
```

```
                strcpy((char *)dhcpd_settings.dhcpd_secondary_dns, secondary_dns);
                strcpy((char *)dhcpd_settings.dhcpd_ip_pool_start, start_ip);
                strcpy((char *)dhcpd_settings.dhcpd_ip_pool_end, end_ip);
                dhcpd_start(&dhcpd_settings);

                printf("start dhcpd, stop dhcp. g_supplicant_ready:%d\n", g_supplicant_ready);
        }
}
```

Setup a softap TCP server, to accept the connect req & recv data:

```
//setup softap TCP server:
int mtk_demo_softap_tcp_server(void)
{
        struct sockaddr_in server, client;
        socklen_t socklen = sizeof(client);
        int fd = -1, connfd, len, ret;
        char *buf, *msg;
        int opt = 1, buf_size = 512, msg_size = 512;

        printf("[mtk demo]: setup softap & tcp-server\n");

        buf = pvPortMalloc(buf_size);
        msg = pvPortMalloc(msg_size);
        assert(fd && msg);

        fd = socket(AF_INET, SOCK_STREAM, 0);
        assert(fd >= 0);

        memset(&server, 0, sizeof(server));
        server.sin_family = AF_INET;
        server.sin_addr.s_addr = htonl(INADDR_ANY);//SOFTAP_GATEWAY_IP, 0xFAFE1FAC;
        server.sin_port = htons(SOFTAP_TCP_SERVER_PORT);

        ret = setsockopt(fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
        assert(!ret);

        ret = bind(fd, (struct sockaddr *)&server, sizeof(server));
        assert(!ret);

        ret = listen(fd, 10);
        assert(!ret);

        printf("[mtk demo]: server %x %d created\n", ntohl(server.sin_addr.s_addr),
                        ntohs(server.sin_port));
```

```
        connfd = accept(fd, (struct sockaddr *)&client, &socklen);
        assert(connfd > 0);
        printf("[mtk demo]: client %x %d connected!\n", ntohl(client.sin_addr.s_addr),
                        ntohs(client.sin_port));


        len = recvfrom(connfd, buf, buf_size, 0,
                        (struct sockaddr *)&client, &socklen);
        assert(len >= 0);

        buf[len] = 0;
        printf("[mtk demo]: softap tcp server recv: %s\n", buf);

        ret = mtk_demo_get_ssid_and_passwd(buf);
        if (!ret)
        {
                snprintf(msg, buf_size,"ok");
        }
        else
                snprintf(msg, buf_size,"fail");

        len = sendto(connfd, msg, strlen(msg), 0,
                        (struct sockaddr *)&client, socklen);
        assert(len >= 0);
        printf("[mtk demo]: ack %s, len %d\n", msg, len);

        close(connfd);
        close(fd);

        vPortFree(buf);
        vPortFree(msg);

        return 0;
}
```

Receive the SSID & PSK,and parse it:

```
/* string info parser */
//recv string format: ssid:ACS9_IOT,passwd:123iot456.
int mtk_demo_get_ssid_and_passwd(char *msg)
{
        char *ptr, *end, *name;
        int len;

        //ssid:
        name = "ssid:\0";
```

```
		ptr = strstr(msg, name);
		if (!ptr) {
			printf("%s not found!\n", name);
			goto exit;
		}
		ptr += strlen(name);
#if 0
		end = ptr;
		while (*end++ == ',');  /* eating the beginning " */
#else
  end = strstr(msg, ",");
#endif
		len = end - ptr;

		assert(len < sizeof(g_mtk_demo_softap_ssid));
		strncpy(g_mtk_demo_softap_ssid, ptr, len);
		g_mtk_demo_softap_ssid[len] = '\0';

		//passwd:
		name = "passwd:\0";
		ptr = strstr(msg, name);
		if (!ptr) {
			printf("%s not found!\n", name);
			goto exit;
		}

		ptr += strlen(name);
#if 0
		end = ptr;
		while (*ptr++ == '.');/* eating the beginning " */
#else
  end = strstr(msg, ".");
#endif
		len = end - ptr;

		assert(len < sizeof(g_mtk_demo_softap_passwd));
		strncpy(g_mtk_demo_softap_passwd, ptr, len);
		g_mtk_demo_softap_passwd[len] = '\0';

#if 0
		//bssid-mac
		name = "\"bssid\":";
		ptr = strstr(msg, name);
		if (!ptr) {
			info("%s not found!\n", name);
			goto exit;
		}
```

```
        ptr += strlen(name);
        while (*ptr++ == ' ');/* eating the beginning " */
        end = strchr(ptr, '"');
        len = end - ptr;
#endif//0

        return 0;
exit:
        return -1;
}
```

The device switch the opmode from AP to STA, then connect to the Router:

```
void mtk_demo_softap_exit(void)
{
   struct netif *sta_if;
   struct netif *ap_if;
   ap_if = netif_find_by_type(NETIF_TYPE_AP);
   dhcpd_stop();
   netif_set_link_down(ap_if);

   sta_if = netif_find_by_type(NETIF_TYPE_STA);
   netif_set_default(sta_if);
   // netif_set_status_callback(sta_if, ip_ready_callback);
   dhcp_start(sta_if);
}

int32_t mtk_demo_connect_ap(unsigned char *get_ssid, unsigned char *get_passwd)
{
        LOG_I(common, "Enter connection init WPA2: mtk_demo_connect_ap(). \n");
        uint8_t opmode  = WIFI_MODE_STA_ONLY;
        uint8_t port = WIFI_PORT_STA;
        uint8_t *ssid = (uint8_t *)get_ssid;
        wifi_auth_mode_t auth = WIFI_AUTH_MODE_WPA_PSK_WPA2_PSK;
        wifi_encrypt_type_t encrypt = WIFI_ENCRYPT_TYPE_TKIP_AES_MIX;
        uint8_t *password = (uint8_t *)get_passwd;
        uint8_t nv_opmode;

        //if (1)
        {
                wifi_config_get_opmode(&nv_opmode);
                if (nv_opmode != opmode) {
                        //wifi_config_set_opmode(opmode);
                        wifi_set_opmode(opmode);
                }
```

```
                wifi_config_set_ssid(port, ssid ,strlen(ssid));
                wifi_config_set_security_mode(port, auth, encrypt);
                wifi_config_set_wpa_psk_key(port, password, strlen(password));
                wifi_config_reload_setting();

                network_dhcp_start(opmode);
        }

  #if 1
        return 0;
  #else
        vTaskDelete(NULL);
  #endif
}
```

3.2. App in PC side

There is a sample app for user can input the **ssid & psk** string, then the data will be transfer to device.

modification as follow:

Cause that MT7687 work as TCP Server & PC App work as TCP Client, so server port should be set to the same as TCP server side.

Should be modified, as the following shown:

```
#define SERVER_PORT 8000
```

MT7687 work as TCP Server . Set the TCP server IP is 172.31.254.250, so the PC App work as TCP Client must connect to the server IP. The IP address is little-endian.

User input the ssid & psk by PC App work as TCP Client,then the data will be transfer to MT7687 work as TCP Server.

```
int main(int argc, char* argv[])
{
        SOCKET sock;
        struct sockaddr_in server;

        char buf_scanf[512];
        WSAData wsa;
        int retval;

        WSAStartup(0x101,&wsa);

        sock = socket(AF_INET,SOCK_STREAM,0);
        if(sock == INVALID_SOCKET)
```

```
	{
		printf("Creating socket error\n");
		return 0;
	}

	server.sin_family = AF_INET;
	server.sin_addr.S_un.S_addr = inet_addr("172.31.254.250");
	server.sin_port = htons(SERVER_PORT);

	if(connect(sock,(sockaddr*)&server,sizeof(server)) < 0)
	{
		retval = WSAGetLastError();
		printf("Conneting to server error\n");
		return 0;
	}

	retval = 1;
	printf("connect to server\n");


	while(1)
	{
		memset(buf_scanf, 0x00, 100);

		//scan the keboard until input string is "exit"
		printf("Send = ");
		scanf("%s",buf_scanf);

		if(strcmp(buf_scanf,"exit") == 0)
			break;


send(sock,buf_scanf,100,0);

		memset(buf_scanf, 0x00, 100);
		retval = recv(sock,buf_scanf,512,0);
		if(retval > 0){

			printf("recv echo data:%s\n",buf_scanf);
		}else
			break;
	}


	closesocket(sock);
	WSACleanup();
```

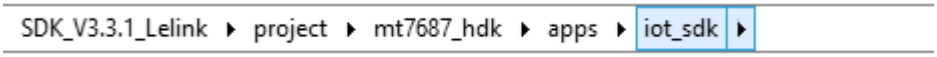
```
        printf("passed\n");

        return 0;
}
```

4. Test & debug

4.1. Software merge

Copy the demo project to the following path:



And then run the build command:

./build.sh mt7687_hdk iot_sdk

4.2. Device

Download the software BIN into MT7687, for how to download, refer to “LinkIt for RTOS get started”

Then power on the MT7687 HDK.

Firstly,set the wifi AP mode. The ssid is MTK_DEMO_SOFT_AP and the auth mode is Open.

```
[T: 9226 M: minisupp C: ERROR F: hostapd_setup_bss L: 562]: Using interface ra1
with hwaddr 00:0c:43:76:62:12 and ssid 'MTK_DEMO_SOFT_AP'
[mtk demo]: Set softAP done.
[DHCPD:DBG]dhcpd_start [0][0]
[DHCPD:DBG]DHCPD preparing
start dhcpd, stop dhcp. g_supplicant_ready:1
[mtk demo]: setup softap & tcp-server
[mtk demo]: server 0 8000 created
[T: 11212 M: inband C: WARNING F: inband_queue_evt_handler L: 772]: u2PacketType
(0xe000), ucEID(0x30), ucSeqNum(0x0) not handled!
[DHCPD:DBG]DHCPD started
[DHCPD:DBG][Server IP]172.31.254.250
[DHCPD:DBG][Netmask]255.255.255.0
[DHCPD:DBG][Gateway]172.31.254.250
[DHCPD:DBG][DNS1]172.31.254.250
[DHCPD:DBG][DNS2]8.8.4.4
[DHCPD:DBG][Start IP]172.31.254.251
[DHCPD:DBG][End IP]172.31.254.254
[DHCPD:DBG]dhcpd task entry:1
[DHCPD:DBG]Wait for UDP
```

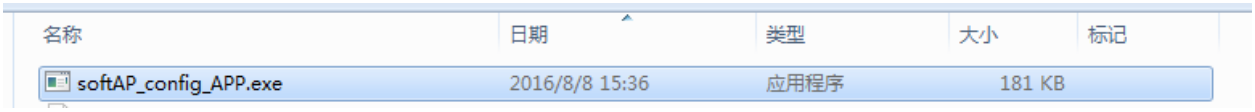
The TCP Server will be setup.The Server IP is 172.31.254.250.

4.3. PC Side

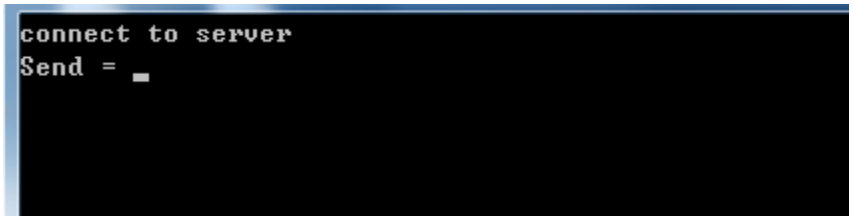
Turn on the wifi radio, then scan the wifi AP(MTK_DEMO_SOFT_AP), then connect to it.

4.4. Operation flow

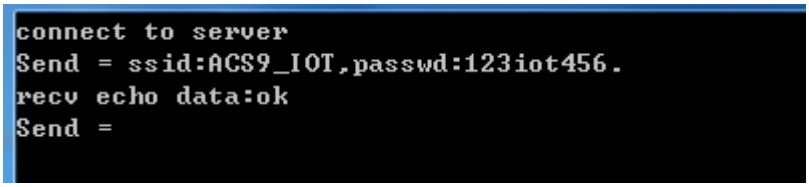
Then you can launch the APP.Doule click the exe program.



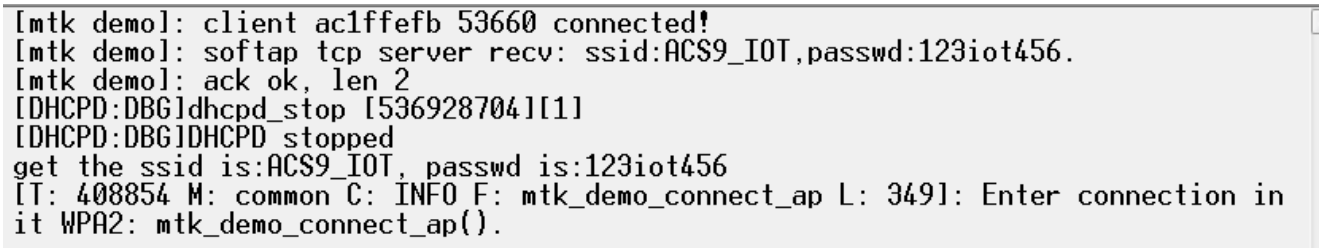
the PC App work as TCP Client will connect to the server IP.



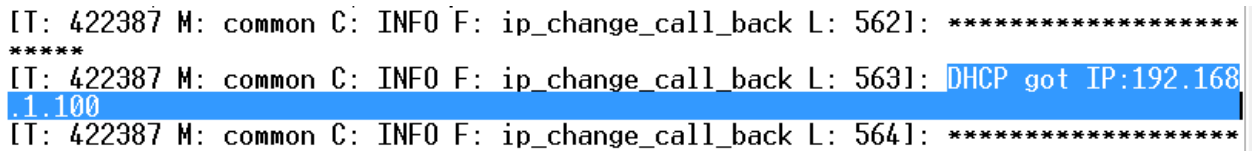
User input the **ssid & psk** by PC App work as TCP Client,then the data will be transfer to MT7687 work as TCP Server.



MT7687 work as TCP Server.Then receive the data, and parse the ssid & psk. The MT7687 will switch the wifi mode from AP to STA, then connect the specific Router.



From the DHCP success, user can check the result of SoftAP wifi configuration.



5. Limitation

5.1. Data Format

User input the **ssid & psk** by PC App. The data format is "**ssid is:xxx,passwd:xxx.**",it is a only demo by MTK.

Customer can modify the data format.

5.2. App

App on PC is a demo app based on windows in this design. Customer can develop a app based on Android or IOS on SmartPhone.