6.5 局域网通讯

一、实验目的

本节课主要学习 WiFi 模块的设置为客户端，电脑作为服务器在局域网的范围内远程控制点灯。

二、实验准备

在 2.WiFi 模块联网的基础上增加设置 WiFi 模块为客户端的功能。这样的好处是开机能够自动连接服务器，并且可以设置为透传模式。

三、实验原理

ESP8285 WiFi 模块是作为客户端接收电脑 TCP 服务器的信息，根据自定义的协议解析，并且根据数据内容实现不同的功能。相当于局域网内远程控制 K210 开发板。

四、实验过程

1.IO4 和 IO5 是 K210 开发板的 USB 串口引脚，使用的是串口 3，而 WiFi 模块的引脚是 IO13 和 IO14，使用的是串口 1。

```
void hardware_init(void)
{
    /* USB串口 */
    fpioa_set_function(PIN_UART_USB_RX, FUNC_UART_USB_RX);
    fpioa_set_function(PIN_UART_USB_TX, FUNC_UART_USB_TX);

    /* WIFI模块串口 */
    fpioa_set_function(PIN_UART_WIFI_RX, FUNC_UART_WIFI_RX);
    fpioa_set_function(PIN_UART_WIFI_TX, FUNC_UART_WIFI_TX);

    /* LED灯 */
    led_init(LED_ALL);
}
```

```
/*****************************HARDWARE-PIN*****************************
// 硬件IO口，与原理图对应
#define PIN_UART_USB_RX       (4)
#define PIN_UART_USB_TX       (5)

#define PIN_UART_WIFI_RX      (13)
#define PIN_UART_WIFI_TX      (14)

/*****************************SOFTWARE-GPIO*****************************
// 软件GPIO口，与程序对应
#define UART_USB_NUM           UART_DEVICE_3

#define UART_WIFI_NUM          UART_DEVICE_1

/*****************************FUNC-GPIO*****************************
// GPIO口的功能，绑定到硬件IO口
#define FUNC_UART_USB_RX       (FUNC_UART1_RX + UART_USB_NUM * 2)
#define FUNC_UART_USB_TX       (FUNC_UART1_TX + UART_USB_NUM * 2)

#define FUNC_UART_WIFI_RX      (FUNC_UART1_RX + UART_WIFI_NUM * 2)
#define FUNC_UART_WIFI_TX      (FUNC_UART1_TX + UART_WIFI_NUM * 2)
```

2. 初始化串口的配置，波特率设置为 115200，串口数据宽度为 8 位，停止位 1 位，不使用奇偶校验。

```
// 初始化USB串口，设置波特率为115200
uart_init(UART_USB_NUM);
uart_configure(UART_USB_NUM, 115200, UART_BITWIDTH_8BIT, UART_STOP_1, UART_PARITY_NONE);

/* 初始化WiFi模块的串口 */
uart_init(UART_WIFI_NUM);
uart_configure(UART_WIFI_NUM, 115200, UART_BITWIDTH_8BIT, UART_STOP_1, UART_PARITY_NONE);
```

3. 开机的时候发送“hello yahboom!”，提示已经开机完成。

```
/* 开机发送hello yahboom! */
char *hello = {"hello yahboom!\n"};
uart_send_data(UART_USB_NUM, hello, strlen(hello));
```

4. 自定义变量，主要用于记录和保存数据。

```
/* 接收和发送缓存的数据 */
char recv = 0, send = 0;
/* 接收WiFi模块数据标志 */
int rec_flag = 0;
/* 保存接收的数据 */
char recv_data[MAX_DATA] = {0};
/* recv_data下标 */
uint16_t index = 0;
```

5. 在接收到数据的时候进行判断，默认 rec_flag 为 0，如果接收到‘$’则表示开始接收数据，rec_flag=1，其中 MAX_DATA 可以自己设定大小，比协议数据内容的最大值大就可以了，这里设置为 10，表示最多能容纳 10 字符。

```
#define MAX_DATA        10
```

```
/* 接收WIFI模块的信息 */
if(uart_receive_data(UART_WIFI_NUM, &recv, 1))
{
    /* 发送接收到的数据到USB串口显示 */
    uart_send_data(UART_USB_NUM, &recv, 1);

    /* 判断是否符合数据要求 */
    switch(rec_flag)
    {
    case 0:
        /* 以‘$’符号为数据开始 */
        if(recv == '$')
        {
            rec_flag = 1;
            index = 0;
            for (int i = 0; i < MAX_DATA; i++)
                recv_data[i] = 0;
        }
        break;
```

6. 开始接收数据，首先是判断是否是‘#’结束符，如果是就结束，并且调用 parse_data 解析数据；如果超过了最大数据还没有接收到‘#’结束符，则表示异常，让 rec_flag 为 0；最后才是保存数据到 recv_data 中。

```
    case 1:
        if (recv == '#')
        {
            /* 以‘#’符号为数据结束 */
            rec_flag = 0;
            parse_data(recv_data);
        }
        else if (index >= MAX_DATA)
        {
            /* 超过最大数据没有接收到结束符‘#’，则清除 */
            rec_flag = 0;
            index = 0;
        }
        else
        {
            /* 保存数据到recv_data中 */
            recv_data[index++] = recv;
        }
        break;
    default:
        break;
    }
}
```

7. 解析数据，对比数据是否与设定的一致，如果一致则执行对应的内容。

```
void parse_data(char *data)
{
    // uart_send_data_dma(UART_USB_NUM, DMAC_CHANNEL0,
    /* 解析并对比发送过来的数据 */
    if (0 == memcmp(data, "led0_0", 6))
    {
        led0_state(LED_OFF);
    }
    else if (0 == memcmp(data, "led0_1", 6))
    {
        led0_state(LED_ON);
    }
    else if (0 == memcmp(data, "led1_0", 6))
    {
        led1_state(LED_OFF);
    }
    else if (0 == memcmp(data, "led1_1", 6))
    {
        led1_state(LED_ON);
    }
}
```

8.把接收的串口数据传输给 WiFi 模块。

```
/* 接收串口的信息，并发送给WiFi模块 */
if(uart_receive_data(UART_USB_NUM, &send, 1))
{
    uart_send_data(UART_WIFI_NUM, &send, 1);
}
```

9.编译调试，烧录运行

把本课程资料中的 wifi_module 复制到 SDK 中的 src 目录下，然后进入 build 目录，运行以下命令编译。

cmake .. -DPROJ=wifi_module -G "MinGW Makefiles"

make

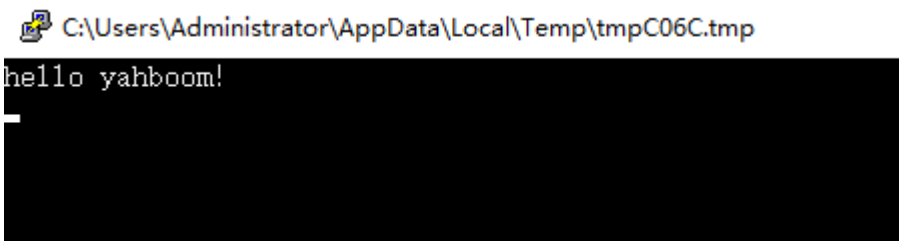
```
Generating .bin file ...
[100%] Built target wifi_module
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build>
```

编译完成后，在 build 文件夹下会生成 wifi_module.bin 文件。

使用 type-C 数据线连接电脑与 K210 开发板，打开 kflash，选择对应的设备，再将程序固件烧录到 K210 开发板上。

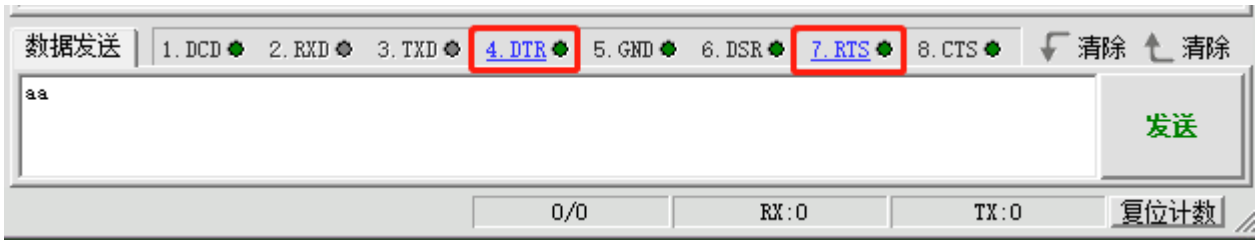
五、实验现象

1.烧录完成固件后，系统会弹出一个终端界面，如果没有弹出终端界面的可以打开串口助手显示调试内容。



2.打开电脑的串口助手，选择对应的 K210 开发板对应的串口号，波特率设置为 115200，然后点击打开串口助手。注意还需要设置一下串口助手的 DTR 和

RTS。在串口助手底部此时的 4.DTR 和 7.RTS 默认是红色的，点击 4.DTR 和 7.RTS，都设置为绿色，然后按一下 K210 开发板的复位键。

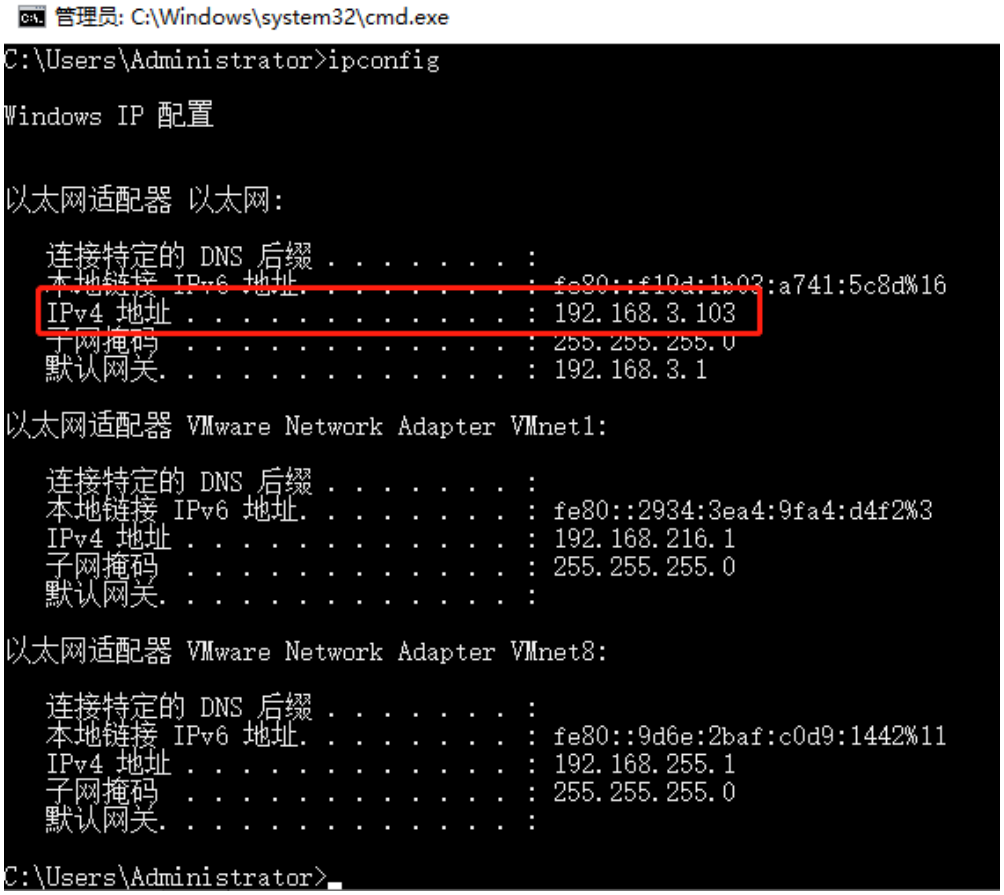


3.从串口助手，可以接收到 hello yahboom!的欢迎语。然后按一下 WiFi 模块的复位键，可以看到一大串乱码，这个不用管，只要看到 ready 字符则表示 WiFi 模块正常。由于上一节课已经连接好路由器，所以我们这次就不必重复连接。

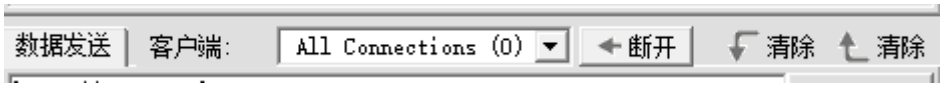
4.先打开网络调试助手 NetAssist，利用网络调试助手来搭建一个 TCP 服务

器。设置网络调试助手的参数，在左上角网络设置中，（1）协议类型选择 TCP Server；（2）远程主机地址选择本地（电脑）的 IP 地址，如果不清楚的可以查看下一步说明；（3）远程主机端口输入 8086。最后点击打开。这里必须要注意一点就是使用的电脑必须要与 WiFi 连接的是同个路由器，否则是无法连通的。

5.如何知道自己电脑的 IP 地址？以 win10x64 位系统为例，按 Win+R，在弹出的对话框输入 cmd 并按回车打开终端，输入 ipconfig 命令。



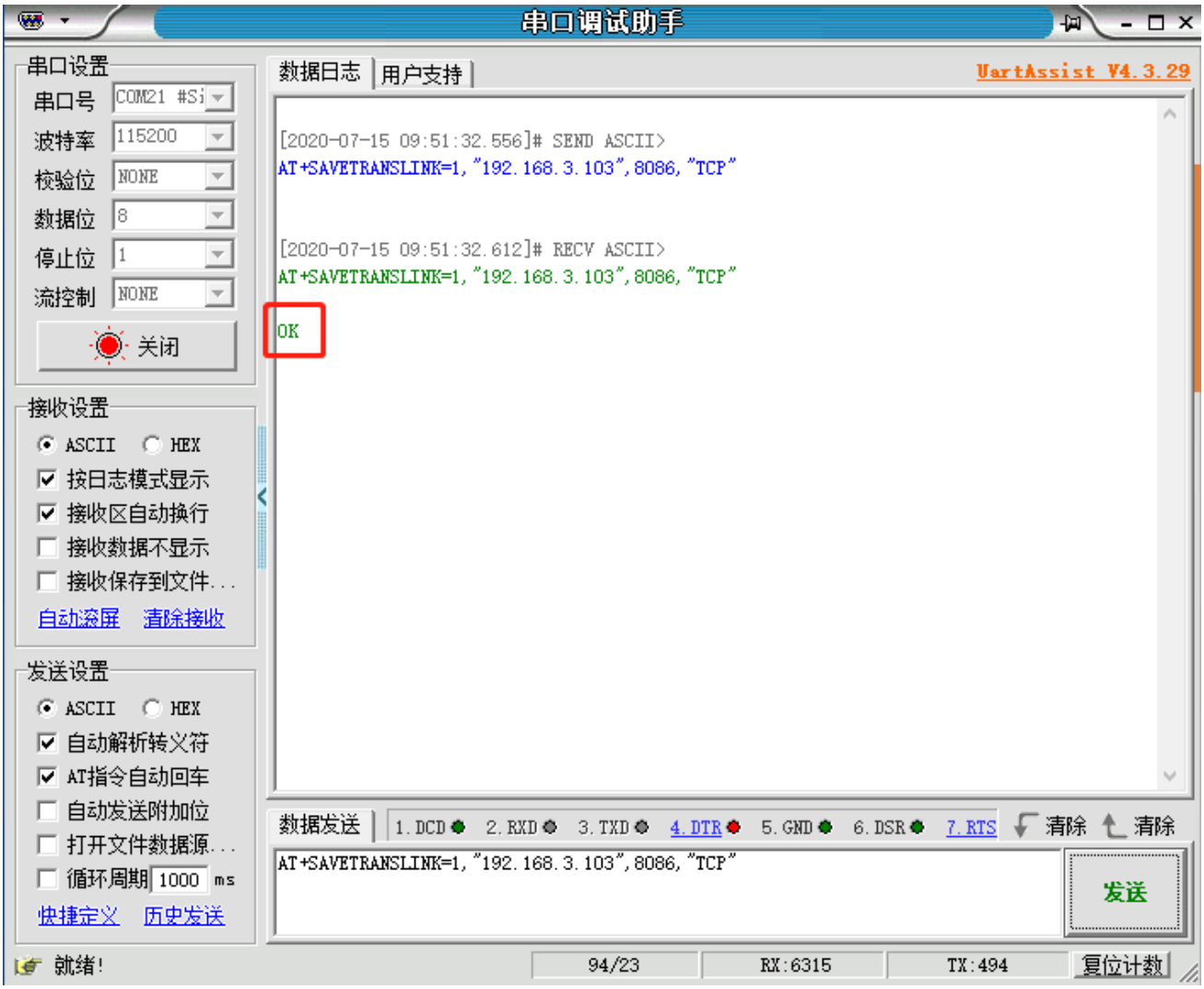
6.服务器打开完成之后，没有客户端连接的情况下会显示客户端为 0，每连接一个数值自动加 1。



7.接下来 WiFi 模块设置为客户端去连接服务器，只需要以下命令。

AT+SAVETRANSLINK=1,"服务器 IP",远程端口,"TCP"

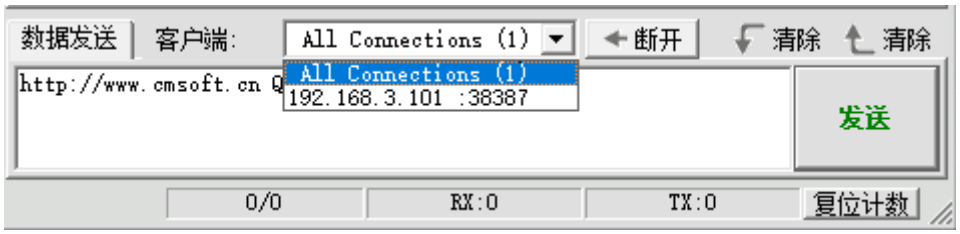
这里以连接电脑服务器 IP 地址为‘192.168.3.103’,远程端口为 8086 为例。



接收到 OK，就表示数据已经写入进去了，但是此时还不会连接到服务器，需要重启一下 WiFi 模块。输入 AT+RST 命令重启 WiFi 模块，接收到 ready，没有接收到 WiFi 连接的信息，此时表示已经成功。因为此命令会保存到 flash，开机会

自启动，并且进入透传模式，如果需要退出透传模式，请先取消‘AT 指令自动回车’的勾，然后发‘+++’就可以。

8. 此时服务器会显示一个已经连接的设备。



9. 此时从服务器发送数据给客户端，客户端也可以显示接收到的数据，或者从客户端发送数据给服务器，服务器也可以接收到数据。这就是透传模式的工作方式。我们送服务器发送点亮 LED0 的协议‘$led0_1#’，可以看到 LED0 红灯亮

起。


网络调试助手
网络设置
（1）协议类型
TCP Server
（2）本地主机地址
192.168.3.103
（3）本地主机端口
8086
关闭
接收设置
ASCII HEX
按日志模式显示
接收完自动换行
接收转向至文件...
暂停接收区显示
其他选项 清除接收
发送设置
ASCII HEX
自动解析转义符
AT指令自动回车
自动发送校验位
打开文件数据源...
循环周期 1000 ms
快捷定义 历史发送
数据日志 用户支持
野人家园出品 V4.3.13
[2020-07-15 10:56:49.738]# SEND ASCII TO ALL>
$led0_1#
数据发送 客户 断开 清除 清除
$led0_1#
发送
就绪 2/8 RX:42 TX:79 复位计数
串口调试助手
串口设置
串口号 COM21 #Si
波特率 115200
校验位 NONE
数据位 8
停止位 1
流控制 NONE
关闭
接收设置
ASCII HEX
按日志模式显示
接收区自动换行
接收数据不显示
接收保存到文件...
自动滚屏 清除接收
发送设置
ASCII HEX
自动解析转义符
AT指令自动回车
自动发送附加位
打开文件数据源...
循环周期 1000 ms
快捷定义 历史发送
数据日志 用户支持
UartAssist V4.3.29
[2020-07-15 10:56:49.847]# RECV ASCII>
$led0_1#
数据发送 1.DCD 2.RXD 清除 清除
发送
就绪 106/26 RX:6728 TX:544 复位计数


4K
LCD
DVP
BOOT

协议对应的功能：

协议	现象
$led0_1#	点亮 LED0
$led0_0#	熄灭 LED0
$led1_1#	点亮 LED1
$led1_0#	熄灭 LED0

六、实验总结

1.本次以 WiFi 模块作为客户端的工作方式为例，至于 WiFi 模块作为服务器的方式与 K210 的程序是一致的，所以就不再演示服务器的方式。

2.wifi_module 程序只是在 wifi_AT 的基础上增加了数据的判断和解析的功能。