第一章 ESP32 的 UDP 广播

1. 学习目的及目标

- 掌握 UDP 原理和工作过程
- 掌握乐鑫 ESP32 的 UDP 的程序设计
- 主要掌握 UDP 源码过程

2. UDP 科普（来自百度百科）

UDP 是 User Datagram Protocol 的简称， 中文名是用户数据报协议，是 OSI（Open System Interconnection，开放式系统互联） 参考模型中一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务，IETF RFC 768 是 UDP 的正式规范。UDP 在 IP 报文的协议号是 17。

UDP 协议全称是用户数据报协议，在网络中它与 TCP 协议一样用于处理数据包，是一种无连接的协议。在 OSI 模型中，在第四层—传输层，处于 IP 协议的上一层。UDP 有不提供数据包分组、组装和不能对数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。UDP 用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但是即使是在今天 UDP 仍然不失为一项非常实用和可行的网络传输层协议。

与所熟知的 TCP（传输控制协议）协议一样，UDP 协议直接位于 IP（网际协议）协议的顶层。根据 OSI（开放系统互连）参考模型，UDP 和 TCP 都属于传输层协议。UDP 协议的主要作用是将网络数据流量压缩成数据包的形式。一个典型的数据包就是一个二进制数据的传输单位。每一个数据包的前 8 个字节用来包含报头信息，剩余字节则用来包含具体的传输数据。

UDP 是 OSI 参考模型中一种无连接的传输层协议，它主要用于不要求分组顺序到达的传输中，分组传输顺序的检查与排序由应用层完成，提供面向事务的简单不可靠信息传送服务。UDP 协议基本上是 IP 协议与上层协议的接口。UDP 协议适用端口分别运行在同一台设备上的多个应用程序。

UDP 提供了无连接通信，且不对传送数据包进行可靠性保证，适合于一次传输少量数据，UDP 传输的可靠性由应用层负责。常用的 UDP 端口号有：

应用协议	端口号
DNS	53
TFTP	69
SNMP	161

UDP 报文没有可靠性保证、顺序保证和流量控制字段等，可靠性较差。但是正因为 UDP 协议的控制选项较少，在数据传输过程中延迟小、数据传输效率高，适合对可靠性要求不高的应用程序，或者可以保障可靠性的应用程序，如 DNS、TFTP、SNMP 等。

3. UDP 特点和流程

上面的原理很重要，但毕竟我们只是在 API 之上做应用。只需要了解特点和流程。知道特点可以做方案时候考量可行性，流程就是可行后的实施。

3.1. UDP 特点：

- 无连接的：发数据前不需要建立连接。
- 不可靠：尽最大努力交付，即不保证可靠交付。
- 支持一对一，一对多，多对一和多对多的交互通信
- 占用资源少，发送数据快。

3.2. UDP 流程： （本段来源）

UDP 编程的客户端一般步骤是：

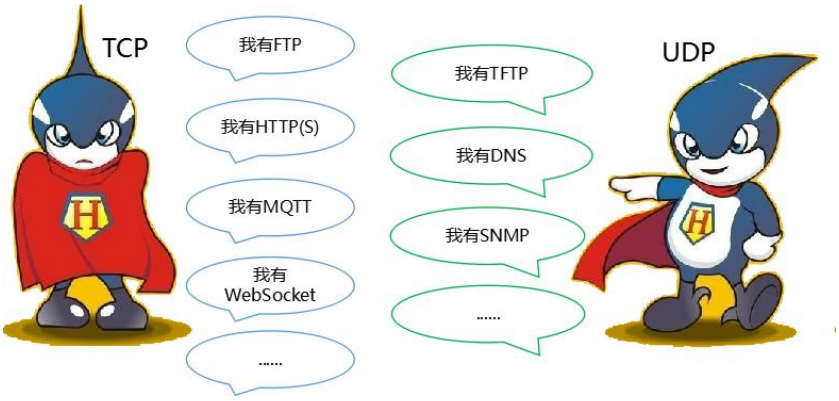
1.创建 UDP socket 套接字，用 socket()函数。

2.用 sendto()函数往指定的 IP，地址发送信息。

TCP 编程的服务器端一般步骤是：

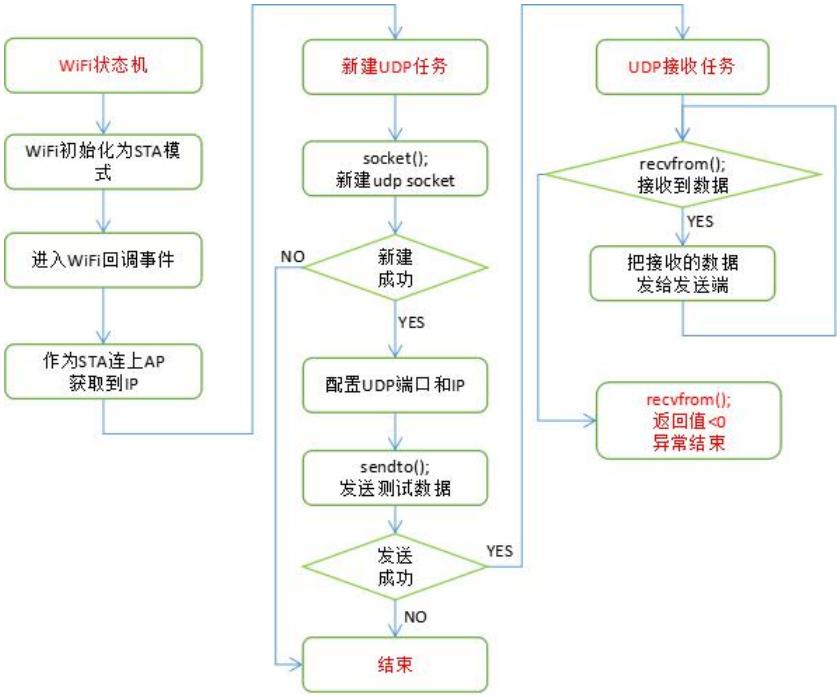
1. 创建 UDP socket 套接字，用 socket 函数。
2. 设置 socket 的属性，用 setsockopt（）函数，（可选）
3. 绑定包含 IP 信息，地址信息的（IPv4）结构体。用 bind（）函数
4. 循环接收消息，用 recvfrom（）函数
5. 关闭 socket 套接字

4. TCP 团伙和 UDP 团伙



5. 软件设计

5.1. ESP32 的 UDP 详细过程



5.2. ESP32 的 UDP Client 接口介绍

- 连接函数：connect();
- 关闭 socket 函数：close();
- 获取 socket 错误代码：getsocketopt();
- 接收数据函数：recvfrom();
- 发送数据函数：sendto();

更多更详细接口请参考官方指南。

5.3. ESP32 的 UDP 总结

初始化 wifi 配置后，程序会根据 WIFI 的实时状态，在回调函数中给出状态返回，所以只需要在回调中进行相关操作，STA 开始事件触发 UDP 工作，上后就可以进行数据的广播。

5.4. UDP 新建任务编写

```
esp_err_t create_udp_client()
{
    ESP_LOGI(TAG, "will connect gateway ssid : %s port:%d",
             UDP_ADRESS, UDP_PORT);
    //新建 socket
    connect_socket = socket(AF_INET, SOCK_DGRAM, 0);                    /*参数和 TCP 不同*/
    if (connect_socket < 0)
    {
        //打印报错信息
        show_socket_error_reason("create client", connect_socket);
        //新建失败后，关闭新建的 socket，等待下次新建
        close(connect_socket);
        return ESP_FAIL;
```

```
    }
    //配置连接服务器信息
    client_addr.sin_family = AF_INET;
    client_addr.sin_port = htons(UDP_PORT);
    client_addr.sin_addr.s_addr = inet_addr(UDP_ADRESS);

    int len = 0;           //长度
    char databuff[1024] = "Hello Server,Please ack!!";    //缓存
    //测试 udp server,返回发送成功的长度
     len = sendto(connect_socket, databuff, 1024, 0, (struct sockaddr *) &client_addr,
               sizeof(client_addr));
     if (len > 0) {
          ESP_LOGI(TAG, "Transfer data to %s:%u,ssucceed\n",
                    inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
     } else {
       show_socket_error_reason("recv_data", connect_socket);
          close(connect_socket);
          return ESP_FAIL;
     }
    return ESP_OK;
}
```

5.5. UDP 接收任务代码

```
void recv_data(void *pvParameters)
{
    int len = 0;           //长度
    char databuff[1024];    //缓存
    while (1)
    {
        //清空缓存
        memset(databuff, 0x00, sizeof(databuff));

        //读取接收数据
           len = recvfrom(connect_socket, databuff, sizeof(databuff), 0,
                     (struct sockaddr *) &client_addr, &socklen);
        if (len > 0)
        {
            //打印接收到的数组
            ESP_LOGI(TAG, "UDP Client recvData: %s", databuff);
            //接收数据回发
            sendto(connect_socket, databuff, strlen(databuff), 0,
                           (struct sockaddr *) &client_addr, sizeof(client_addr));
        }
        else
        {
            //打印错误信息
```

```
            show_socket_error_reason("UDP Client recv_data", connect_socket);
            break;
        }
    }
    close_socket();

    vTaskDelete(NULL);
}
```

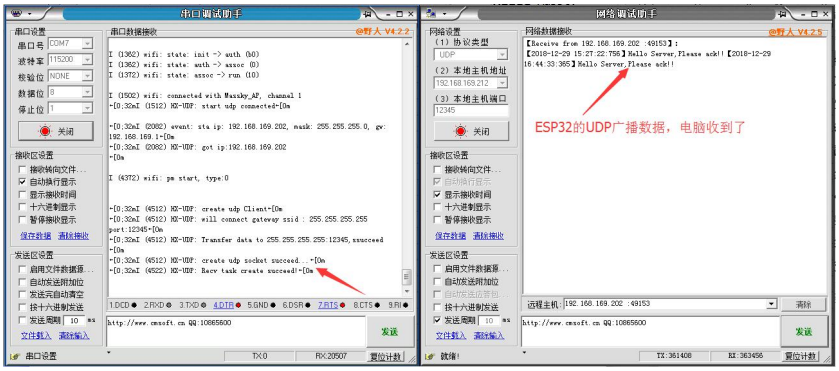
6. 测试流程和效果展示

6.1. 测试流程

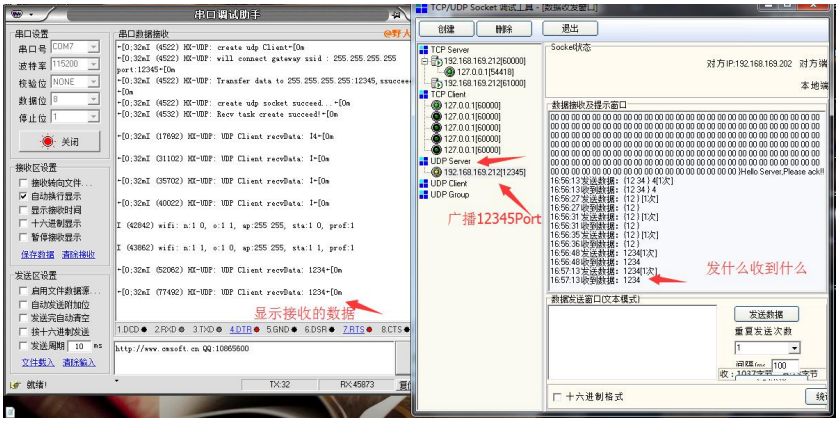
- 修改 AP 和 STA 的账号密码
- 修改 UDP Port
- 使用手机或者电脑使用助手工具进行 UDP 广播测试

6.2. 效果展示

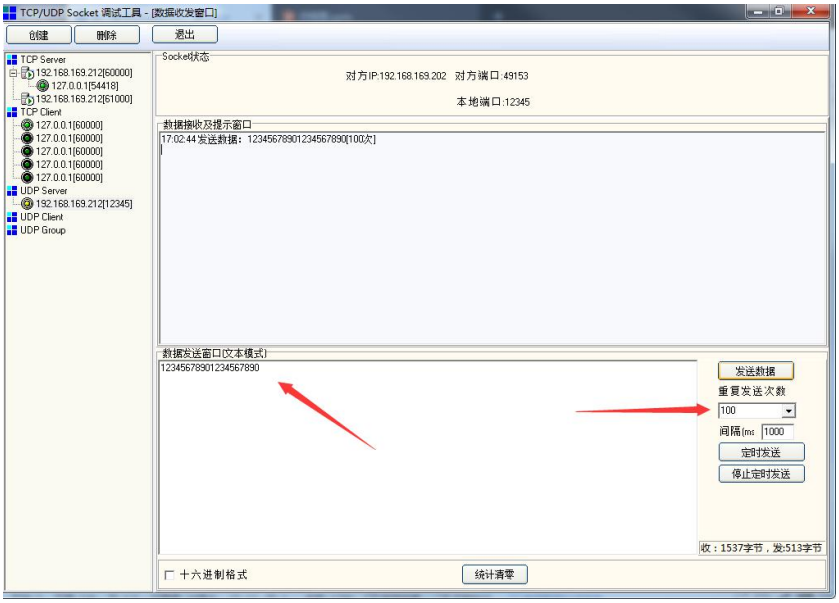
测试发送数据

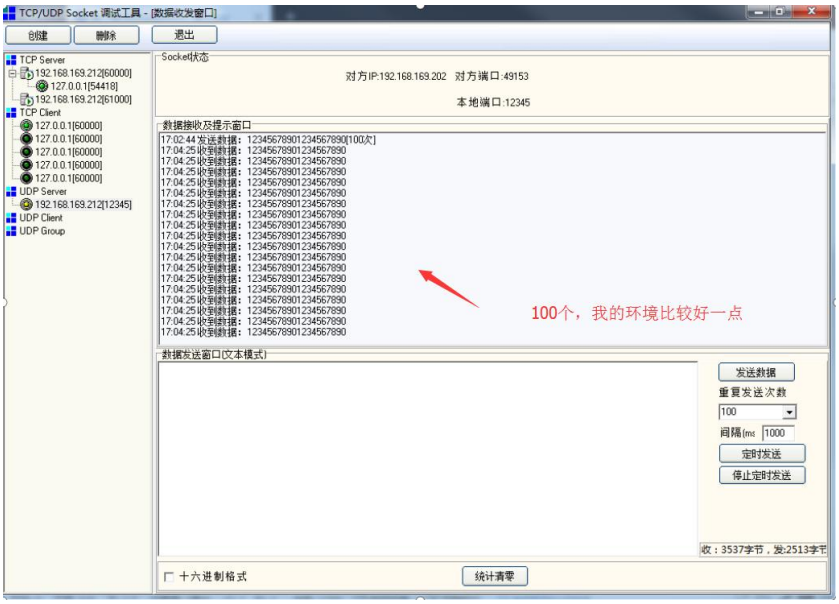


收发小测



发送 100 次，看看不可靠的程度





7. UDP 总结

- 底层重原理，应用中流程+接口。
- 此源码没有异常处理，自己移植需要适当修改，在接收任务中看返回值，决定是否重新新建 UDP Client，与 TCP 类似的操作。
- 源码地址：https://github.com/xiaolongba/wireless-tech