重　庆　大　学

微 电 子 与 通 信 工 程 学 院

课 程 设 计 报 告

课题名称：　通信系统综合设计与实践

年　　级：

专业班级：

学　　号：

学生姓名：

完成日期：　2023.7.12

成　　绩：

教师签名：

微电子与通信工程学院　制

二〇二三年七月

重庆大学本科学生课程设计任务书

<table>
<tr><td colspan="2">课程名称</td><td colspan="4">通信系统综合设计与实践</td></tr>
<tr><td>学院</td><td>微电子与通信工程</td><td>年级</td><td></td><td>专业、班</td><td></td></tr>
<tr><td>课题题目</td><td colspan="5">低速星型无线通信系统设计</td></tr>
<tr><td colspan="6">设计要求：
（1）深入学习 Arduino 开发技能，独立完成以 Arduino 为控制平台的无线传输设备设计，实现无线方式下的信息传输；
（2）深入理解无线网络知识，理解可靠通信原理，完成低速星型无线通信系统设计，实现网络控制与信息传递。
（3）团队分工合作，每位同学完成相应设计工作（必须含程序编写）。
基本要求：
（1）设计包含 1 个主节点，3 个终端节点的星型无线通信系统；
（2）PC 端基于 LabVIEW 设计可视化界面，实现数据采集、存储，曲线拟合，实现对无线通信系统的控制功能；
（3）整个系统之间的数据传输均采用统一的通信协议（可自定义）进行。
提高要求：
（1）实现 3 个终端节点的无线组网通信（通过主节点转发完成）；
（2）实现节点灵活入网以及网络拓扑维护，实现 PC 端对任意无线设备的信道参数的设置与读取，实时显示无线节点的在网状态；
（3）确保整个系统的可靠性和稳定性的前提下提高网络效率；
（4）完成组网设计方案编写，实现入网、网络维护、网络状态信息显示告警、网络参数设置在线调整（网络运行时可调整）等方案设计，完成其状态转换图、软件模块设计、结构化软件设计、伪代码设计。（此设计将作为最终设计完成考核依据，如某些假定设计无法最终完成可根据设计的思想以及伪代码等给予相应评价）；
（5）扩展发挥，设计实用环境完成可移动设备的远程控制、信息传输等。</td></tr>
<tr><td colspan="6">参考资料：
[1] 陈吕州.Arduino 程序设计基础（第 2 版）. 北京：北京航空航天大学出版社，2015.
[2] 沈金鑫.Arduino 与 LabVIEW 开发实战. 北京：机械工业出版社，2014.
[3] 李培毅.Arduino 实验案例指导手册（简版 0917）.</td></tr>
<tr><td colspan="6">课程设计工作计划：
1．2023 年 6 月 26 日 到 2022 年 6 月 29 日，深入理解掌握 Arduino 开发原理，学习完成无线设备基本设计，完成网络系统的设计方案。
2．2023 年 6 月 30 日 到 2022 年 7 月 7 日，完成各通信模块设计测试，完成基本组网设计。
3．2023 年 7 月 8 日 到 2022 年 7 月 14 日，完成整体系统程序编写与测试，完成报告编写。</td></tr>
<tr><td colspan="6">任务下达日期 2023 年 6 月 26 日 完成日期 2023 年 7 月 14 日
指导教师____
学生_____</td></tr>
</table>

目　录

低速星型无线通信系统设计

1.需求分析

1.1 设计目的

（1）深入学习 Arduino 开发技能，独立完成以 Arduino 为控制平台的无线传输设备设计，实现无线方式下的信息传输；

（2）深入理解无线网络知识，理解可靠通信原理，完成低速星型无线通信系统设计，实现网络控制与信息传递。

1.2 设计环境

（1）硬件：Arduino 板及 Lora 无线通信模块若干套，串口线多条 。

（2）软件：Arduino IDE 开发软件，Labview 开发软件。

1.3 设计要求

（1）设计包含 1 个主节点，3 个终端节点的星型无线通信系统。

（2）PC 端基于 LabVIEW 设计可视化界面，实现数据采集、存储，曲线拟合，实现对无线通信系统的控制功能。

（3）整个系统之间的数据传输均采用统一的通信协议进行。

（4）实现 3 个终端节点的无线组网通信（通过主节点转发完成）。

（5）实现节点灵活入网以及网络拓扑维护，实现 PC 端对任意无线设备的信道参数的设置与读取，实时显示无线节点的在网状态。

（6）确保整个系统的可靠性和稳定性的前提下提高网络效率；

（7）完成组网设计方案编写，实现入网、网络维护、网络状态信息显示告警、网络参数设置在线调整（网络运行时可调整）等方案设计，完成其状态转换图、软件模块设计、结构化软件设计、伪代码设计。

2.总体方案设计

2.1 整体架构

低速星型无线通信系统的设计参考了无线局域网协议 802.11 标准，它使用星型拓扑，无线局域网的中心叫做接入点 AP，它是无线局域网的基础设施，也是一个链路层的设备。所有在无线局域网中的站点，对网内网外的通信，都必须通过接入点 AP。现在的

无线局域网的接入点往往具有 100Mbit/s 或 1Gbit/s 的端口，用来连接到有线以太网。家庭使用的无线局域网接入点 AP，为了方便居民上网，就把 IP 层的路由器功能也嵌入进来。因此家用的接入点 AP 往往又称为无线路由器，但企业或机构使用的接入点 AP 还是和路由器分开的。

本系统设计模拟热点连接，如图 2-1 所示，为手机热点连接界面，记录了已连接设备的地址和名称，并可操纵已连接设备退网，被退网的设备即进入黑名单中被禁止使用本机热点。

图 2-1 手机热点连接界面

在本设计中，低速星型无线通信系统整体架构图如图 2-2 所示，系统物理层采用 Lora 模块，着重于通过链路层的设计实现系统整体功能，网络层及以上功能不采用。

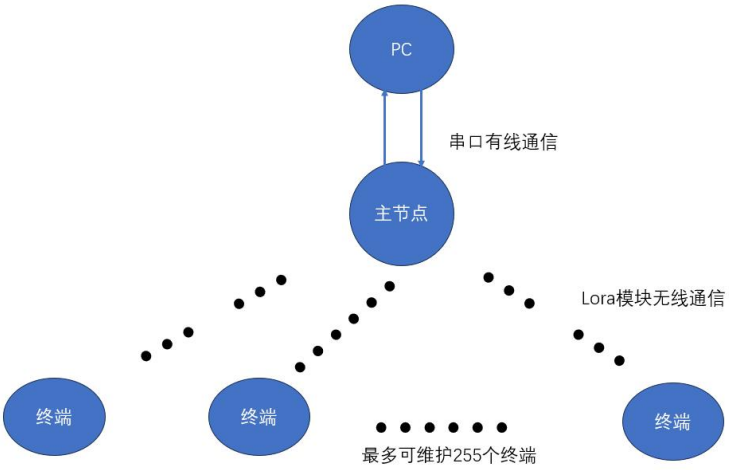

图 2-2 低速星型无线通信系统整体架构

（1）PC 端用户操作 LabVIEW 程序前面板实现对整个通信系统的控制及状态显示，其中控制命令有：

①允许指定终端入网

②命令指定终端退网

③网络参数设置在线调整

④显示在网终端

⑤显示黑名单（不被允许入网的终端）

⑦命令在网终端发送测量数据并将测得的数据做可视化显示

⑧网络状态信息显示告警等

（2）主节点实时维护和记录网络状态并执行 PC 端下发的命令。通过自定义 PC 端与主节点的串口通信协议，LabVIEW 程序将用户操作转化为对应帧命令下发给主节点，主节点通过 Lora 模块与各接入终端通信，最终实现用户命令以及网络维护

（3）终端节点之间的数据传递通过主节点转发完成。

2.2 网络维护

主节点为整个系统的核心部分，实现 PC 端和终端间的交互，并实时记录网络状态和维护网络运行。主节点工作过程如图 2-3 所示。

```
void loop()
{
  String Command = "";                    //定义接收主机命令的变量
  if(RecvFrPC(&Command))                  //判断是否有来自主机的命令
  {
    PC_Permit(Command);                   //入网允许命令
    PC_Logout(Command);                   //退网命令
    PC_ChangeFre(Command);                //改变网络参数命令
    PC_Link(Command);                     //更新Link终端命令
    PC_Update(Command);                   //更新网络状态表命令
    PC_BlackList(Command);                //更新黑名单命令
  }
  Broadcast();                            //广播通知附近设备可接入
  KeepContact();                          //与已接入设备保持联系
  AskExist();                             //再次联系可能断开连接的设备
  if(Node_Link != NULL) GetLinkData();    //对Link终端采集数据
  delay(100);
}
```

图 2-3 主节点工作过程

主节点通过两个链表实时记录网络状态并对网络进行维护，如图 2-4 和如 2-5 所示，分别为接入链表和黑名单链表链接的结构体，分别记录在网设备的信息和进入黑名单设备的信息。

```
//网络状态链表 ================================================================================
typedef struct node NODE;
typedef struct node{
  NODE* Next;        //下一节点
  byte Addr;         //地址
  char* Name;        //名字
  byte State;        //退网时掉线 切换频率时掉线 建立连接时掉线 断开连接时掉线 传输数据时掉线 保持联系时掉线
  byte Time;         //超时重传次数
}NODE;

NODE* NodeList = NULL;      //链表表头
byte StateMask[6] = {0x01,0x02,0x04,0x08,0x10,0x20};
```

图 2-4 接入链表

```
//黑名单链表=========================================
typedef struct BlackNode BNODE;
typedef struct BlackNode{
  BNODE* Next;        //下一节点
  byte Addr;         //地址
  char* Name;       //名字
}BNODE;

BNODE* BlackList = NULL;                //链表表头
```

图 2-5 黑名单链表

（1）设备入网

终端与主节点建立关联的方法有两种，一种是主动扫描，一种是被动扫描，或者可以同时进行主动扫描和被动扫描，这样可以更加迅速的建立关联。本系统设计采用被动扫描，其可以节省终端处的电源功率消耗，其过程如图 2-6 所示：

①主节点周期性调用函数 Broadcast( )发出广播帧，告知附近终端可接入。

②附近终端收到广播帧，可向主节点发出接入请求。

③主节点收到接入请求，向主节点发送接入确认帧。

这样主节点和终端节点间的关联就建立了，成功入网的终端则被加入接入链表。

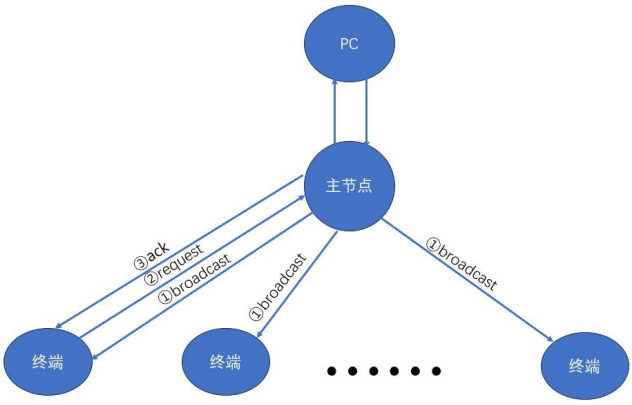

图 2-6 入网过程

（2）退网过程

类似于热点连接，退网有两种情况。

①终端用户可主动退网。主动退网的设备不再响应主节点的消息，主节点发现该终端掉线后则从接入链表中删除该终端节点。

②PC 端要求指定终端退网。主节点则调用函数 PC_Logout( )将 PC 端指定退网的终端移除出接入链表并记录到主节点的黑名单中，主节点不响应黑名单中终端的接入请求，若该终端想再次接入网络，必须由 PC 端给主节点发送入网允许命令，主节点则调用函数 PC_Permit( )将指定终端从黑名单中删除。

（3）网络维护

①已经接入网络的终端节点会维持一个“保活计时器”，每次收到主节点的消息（非广播帧）都会更新“保活计时器”从头计时，当该计数器减到 0 时就认为主节点故障并和主节点所在网络断开连接。

②主节点周期性地调用函数 KeepContact( )，向已在网终端发送一个空数据帧以告知各在网终端主节点正常工作中。

由于该种命令帧不需要携带数据，因此我们将终端间通信的功能也结合到这一部分中来，终端有数据发送时，先将数据存储到缓冲区，在下次对主节点的保持联系命令的确认中将要发送的数据写入到 ACK 帧的数据部分，主节点收到 ACK 帧后发现携带数据就将数据及目的地址解析出来转发给目标终端。

③对于可能故障的终端节点（主节点超时未收到该终端的反馈帧），主节点会对其进行记录，并在周期性的调用函数 AskExist( )对该终端再次访问，调用 Time_Timeout 次该函数仍未收到反馈，则认为该终端掉线，并从接入链表中删除。

2.3 PC 端命令

除 2.2 网络维护部分外，主节点还可以实现来自端的其他命令，如调用函数 PC_ChangeFre( )修改系统工作频率，调用函数 PC_Update( )向 PC 端反馈最新的接入链表，调用函数 PC_BlackList( )向 PC 端反馈最新的黑名单链表，调用函数 PC_Link( )更新 Link 终端（注：PC 端指定要求上报数据的终端，一个系统中同时只有一个 Link 终端），同时主节点周期性的采集 Link 终端的数据上报给 PC 端，PC 端通过 LabVIEW 程序实时绘制出数据曲线。

3.Arduino 功能模块设计

主节点与端节点间通信的帧结构：

3-1 主节点与端节点间通信的帧结构

主节点与 PCLabView 间串口通信帧结构：

```
Serial.write(CHAR_SYNC);//同步位
Serial.write(Function);//功能位
Serial.print(MsgData);//信息位
Serial.print(FCS);//校验位
Serial.write(CHAR_END);//终止位
```

3-2 主节点与 PC 端间串口通信的帧结构

主节点与端节点之间的功能位对应的功能有：

0：退网功能

1：切换信道频率

2：建立连接状态

3：断开连接

4：采集数据

5：保持联系与转发

6：中心节点广播

7：端节点申请接入

8：端节点接入成功

首先中心节点读取串口数据，中心节点读取来自 PC 发送的串口数据，若数据符合与主机间的报文格式约定则接收该报文，并执行报文功能位对应的功能。

3.1 退网功能

主机要求终端 i 退出网络。首先从 PC 发来的帧中解析出要求退网的目的地址，根据目的地址首先判断该终端是否在网，若本来就不在网就在串口输出显示“addr(目的地址) off net(erro)”,然后中心节点将功能位为“0”的通信帧发给目的端节点，并同时开启一个定时器，若主机在规定时间内收到了来自终端 i 的确 认报文，则将终端 i 退出接入网络列表，并加入黑名单链表，并在串口输出显示“addr (地址): successfully logout”；若在规定时间内没有收到终端 i 的确认报文，则将网络状态链表中的 State 的第 0 位置 1 表示退网功能失败，并在串口中打印输出“addr（目的地址）: logout-response time out(erro)”。端节点收到退网命令，则按上图执行相应操作，即将表示在网状态的 ACS 置 0 和表示连接状态的 Link 置 0，同时给中心节点返回一个 ACK 确认帧。

```
if(msg_r.Function == '0')                    //接到要求退网命令
{
  Serial.println("接到要求退网命令");
  Acs = false;
  Link = false;
  msg_s.Function = (byte)'0';
  (msg_s.MsgData) = CHAR_ACK;
  SendToCN(&msg_s);
}
```

图 3-3 退网功能设计

3.2 切换频率功能

主机要求切换信道频率，中心节点在将功能位为 1 的通信帧发给要求切换频率的端节点，同时开启一个定时器，若中心节点在规定时间内收到了来自目标端节点的返回确认信息，则在串口打印输出“Addr OK”,若未能收到返回去确认帧则将该目的端节点的 State 的第 1 位置 1，表示切换频率失败，完成一个端的换频后将链表指针指向地址指向下一位地址，重复上述操作，直至将所有信道频率切换完毕。

```
else if(msg_r.Function == '1')            //接收到更改频率命令
{
  Timer_start(MAX_WAIT_TIME_CONTACT);
  Serial.println("接收到更改频率命令");
  msg_s.Function = '1';
  msg_s.MsgData = CHAR_ACK;
  SendToCN(&msg_s);
  LoRa.setFrequency(msg_r.MsgData.toInt() * 1E6);
}
```

图 3-4 切换频率功能设计

端节点收到更改频率命令后，将功能位为“1”的 ACK 确认帧返回给中心节点。

3.3 建立连接功能

主机要与目的地址的端节点建立连接，中心节点首先通过 Find_Node 函数在网络状态链表中寻找对应地址的节点，若未找到说明目的端节点还未进入网络，则输出显示“addr :off net (erro)”，并返回一个功能位为 2 的帧给主机，否则则将功能位为“2”的通信帧发给目的端接点，并开启一个定时器，若在规定时间内未收到返回确认帧，则将目的端节点的网络状态链表中的 State 的第 2 位置 1 表示建立连接失败，如果收到返回确认帧，则输出显示“addr: successfully linked”。

```
else if(msg_r.Function == '2')            //收到建立连接命令
{
  Timer_start(MAX_WAIT_TIME_CONTACT);
  Serial.println("收到建立连接命令");
  Link = true;
  msg_s.Function = '2';
  (msg_s.MsgData) = CHAR_ACK;
  SendToCN(&msg_s);
}
```

图 3-5 建立连接功能设计

端节点收到功能位为“2”的建立连接通信帧后，重置定时，并将 Link 置 1 表示建立连接，并返回一个同样功能位的 ACK 确认帧给中心节点。

3.4 断开连接功能

主机要求目的地址的端节点断开连接，中心节点将功能位为“3”的通信帧发给目的端接点，并开启一个定时器，若在规定时间内未收到返回确认帧，则将目的端节点的网络状态链表中的 State 的第 3 位置 1 表示断开连接失败，如果收到返回确认帧，则输出显示“addr : successfully broke the old link”。

```
else if(msg_r.Function == '3' && Link)          //收到断开连接命令
{
  Timer_start(MAX_WAIT_TIME_CONTACT);
  Serial.println("收到断开连接命令");
  Link = false;
  msg_s.Function = '3';
  msg_s.MsgData = CHAR_ACK;
  SendToCN(&msg_s);
}
```

图 3-6 断开连接功能设计

端节点收到功能位为“3”的断开连接通信帧后，重置定时，并将 Link 置 0 表示断开连接，并返回一个同样功能位的 ACK 确认帧给中心节点。

3.5 采集数据功能

主机命令采集已经建立联系 Link 为 1 的目的端节点，中心节点发送一个功能位为“4”的通信帧给目的端节点，并开启一个定时器，若在规定时间内未收到端节点返回的确认帧，则将目的端节点在网络状态表中的 State 的第 4 位置 1 表示采集数据失败，若收到确认帧则不进行其他操作。

```
else if(msg_r.Function == '4' && Link)          //收到发送数据命令
{
  Timer_start(MAX_WAIT_TIME_CONTACT);
  Serial.println("收到发送数据命令");
  msg_s.Function = '4';
  msg_s.MsgData = String(GetData());
  SendToCN(&msg_s);
}
```

图 3-7 采集数据功能设计

端节点收到采集数据命令后，通过 GetData 函数将一个随机产生的 0—255 之间的整数发送给中心节点。

3.6 保持联系及转发功能

（1）在完成组网和建立连接后，中心节点向已建立联系的端节点发送功能位为 5 的报文，并打开一个定时器，若在规定时间内未能收到端节点返回的确认报文，则将该端节点的 State 的第 5 位置 1 表示保持联系失败。

（2）端节点收到保持联系报文后，将该端节点从串口输入的要发送数据通信的下一端节点的地址作为返回确认报文的数据，返回确认报文功能位为 5。

（3）中心节点收到该返回报文后，将报文中的地址作为转发报文的目的地址，转发报文的信息为返回确认报文的源地址，将该转发报文转发出去，由此完成了两终端节点间的通信。

（4）同时如果保持联系失败，则进行询问是否掉线的操作，若连续询问三次后，

仍未得到端节点的回复，则将该端节点从网络状态链表中删除。

3.7 组网功能

（1）中心节点广播功能

中心节点向公共广播地址发送一个功能位为“6”的广播帧，表示端节点可申请接，同时开启一个定时器，若未在规定时间内收到来自任一端节点的返回确认帧，则进行超时重传，端节点收到广播帧后即进行申请接入。

（2）端节点申请接入功能

端节点向中心节点发送一个功能位为“7”的请求接入的通信帧给中心节点，同时打开一个定时器，若在规定时间内未收到来自中心节点的确认接入成功的 ACK 确认帧，则在串口打印输出“申请接入失败”，并将 ACS 置 0 表示未接入网络。若成功收到确认报文，则重置最大保持联系时钟，并将 ACS 置 1 表示接入网络。

（3）端节点接入成功功能

在完成前两步后，若中心节点收到端节点发来的正确的申请接入报文后，将返回给该端节点一个功能位为“8”的确认帧，表示端点接入网络成功，并将该节点加入网络状态链表。

4.Labview 功能模块设计

LabVIEW 端使用 VISA 插件实现 LabVIEW 串口编程。VISA 是虚拟仪器软件体系结构（Virtual Instruments Software Architecture）的缩写，实质上是一个 I/O 口软件库及其规范的总称，应用于应用于仪器编程的标准 I/O 应用程序接口。

（1）VISA 配置串口函数：将指定串口根据特定设置初始化。

（2）VISA 写入函数：将 PC 端发送的字符串帧写入指定串口。

（3）VISA 读取函数：将 PC 端收到的字符串帧从指定串口中读取出来。

(1)VISA 配置串口函数　　(2)VISA 写入函数　　(3)VISA 读取函数

图 4-1 LabVIEW 的基本串口通信工具

4.1 串口写入模块

串口写入模块整体采用了事件结构函数，当某按键布尔值改变（即按下按钮）时，模块会自动执行此按键所对应的功能分支，完成该功能的字符串帧写入（即 PC 端向中心节点发送数据）。

为了实现 PC 端与中心节点间发送与接收的帧功能彼此对应，我们将事件结构的超时时间设置为 1ms。

（1）允许入网模块

PC 端通过该模块将允许入网帧发送给中心节点，告知中心节点哪些终端被允许入网。

当用户按下允许入网按键后，PC 端将帧头‘#’、功能位‘/’、允许入网地址和帧尾‘*’分别以字符串的形式连接成一个新的字符串，写入指定串口。

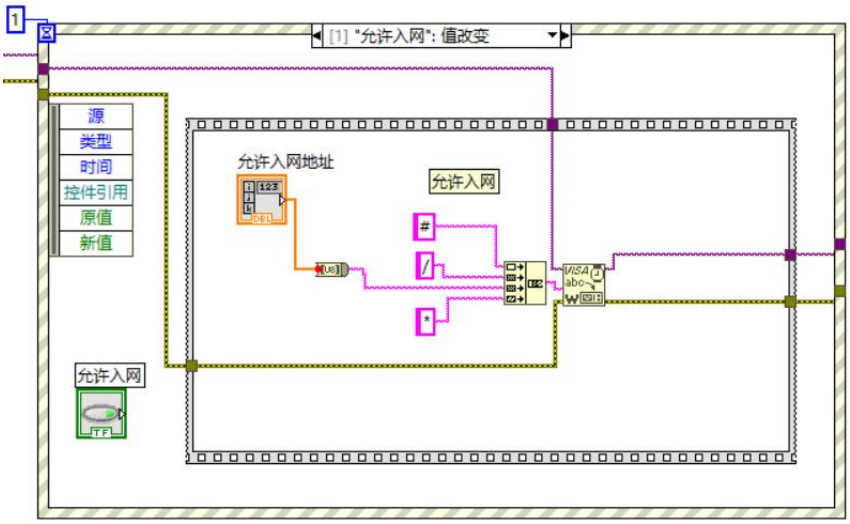

图 4-2 允许入网模块

（2）禁止入网模块

PC 端通过该模块将禁止入网帧发送给中心节点，告知中心节点哪些终端被禁止入网。

当用户按下禁止入网按键后，PC 端将帧头‘#’、功能位‘0’、禁止入网地址和帧尾‘*’分别以字符串的形式连接成一个新的字符串，写入指定串口。

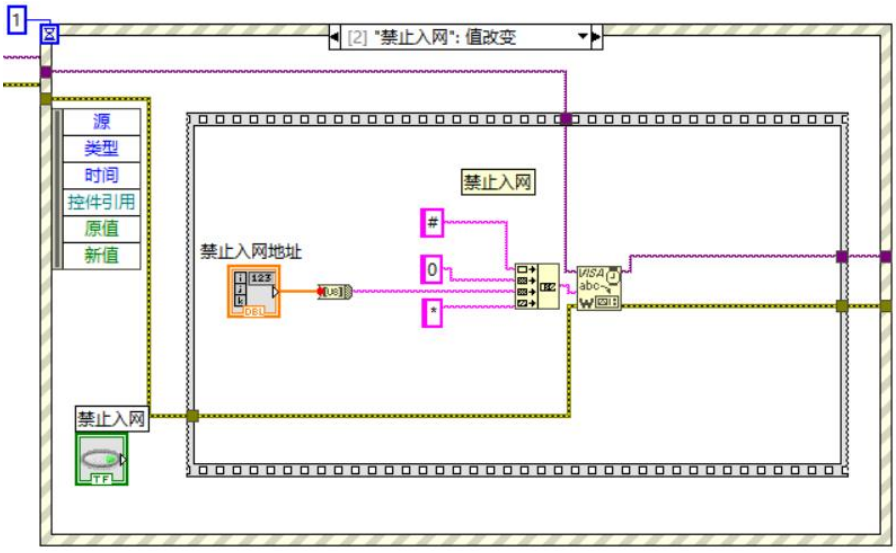

图 4-3 禁止入网模块

（3）信道频率设置模块

PC 端通过该模块将信道频率设置帧发送给中心节点，告知中心节点所设置的信道频率。

当用户按下设置频率按键后，PC 端将帧头‘#’、功能位‘1’、信道频率和帧尾‘*’分别以字符串的形式连接成一个新的字符串，写入指定串口。

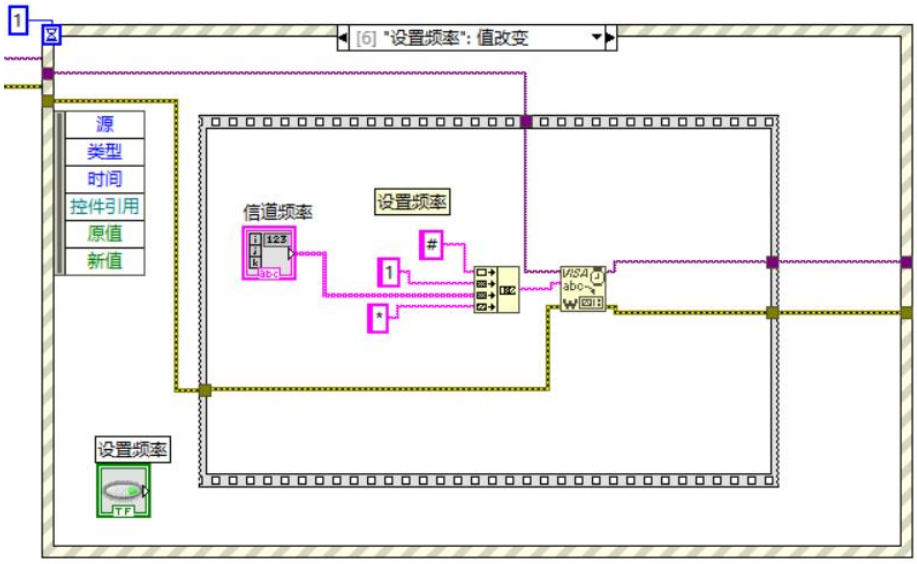

图 4-4 信道频率设置模块

（4）采集数据模块

PC 端通过该模块采集数据帧发送给中心节点，告知中心节点 Link 终端的地址。（Link 终端：被采集数据的终端）

当用户按下采集数据按键后，PC 端将帧头‘#’、功能位‘2’、Link 终端地址和帧尾‘*’分别以字符串的形式连接成一个新的字符串，写入指定串口。

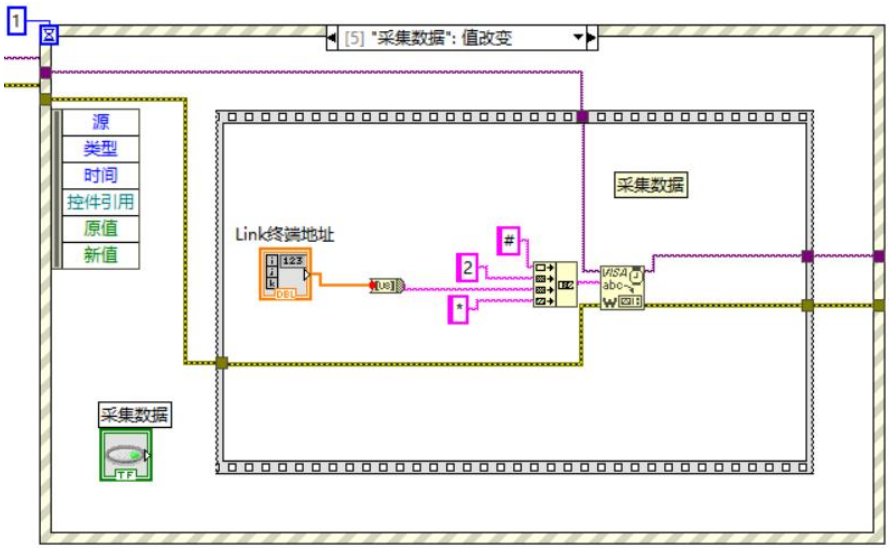

图 4-5 采集数据模块

（5）在网状态表更新模块

PC 端通过该模块将在网状态表查询帧发送给中心节点，请求中心节点反馈最新的在网状态表。

当用户按下更新在网状态表按键后，PC 端将帧头‘#’、功能位‘5’、和帧尾‘*’分别以字符串的形式连接成一个新的字符串，写入指定串口。

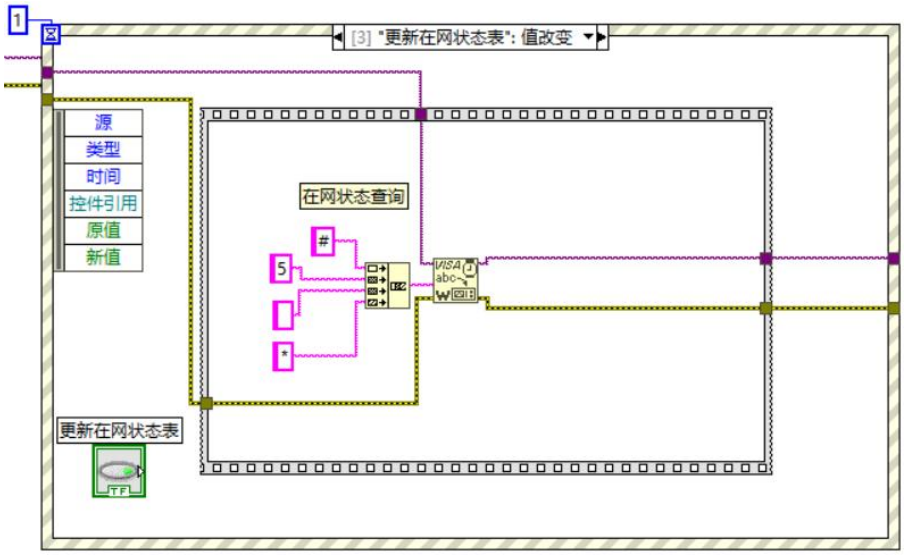

图 4-6 在网状态表更新模块

（6）黑名单更新模块

PC 端通过该模块将黑名单查询帧发送给中心节点，请求中心节点反馈最新的黑名单。

当用户按下更新黑名单按键后，PC 端将帧头‘#’、功能位‘6’、和帧尾‘*’分别以字符串的形式连接成一个新的字符串，写入指定串口。

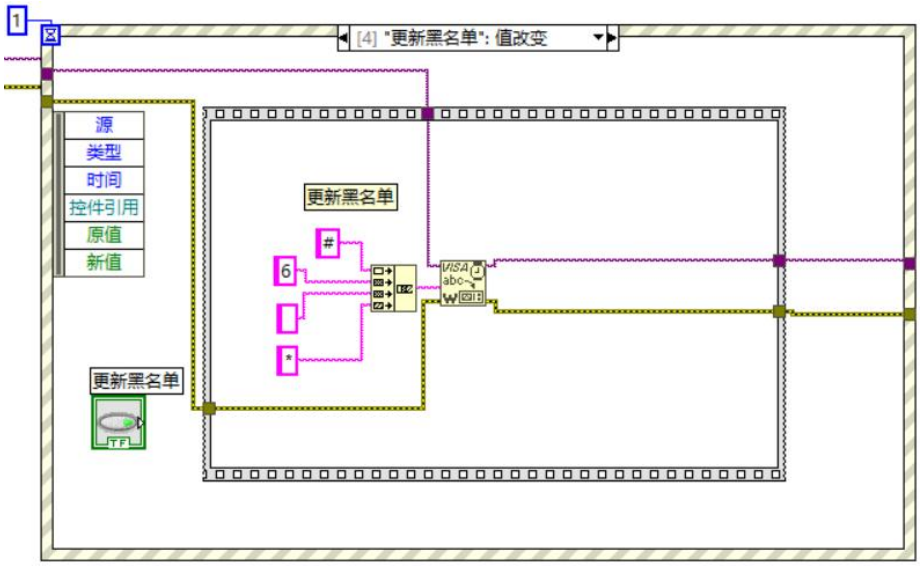

图 4-7 黑名单更新模块

4.2 串口读取模块

首先采用图 3-8 所示模块判断串口是否有数据输入。若有，则进入串口读取模块。

如图 3-9 所示，串口读取函数中心节点对指定串口中的数据（即中心节点发送给 PC 端的反馈帧）进行读取，再判断帧首和帧尾是否正确。若正确，则将剔除掉帧头帧尾的字符串传递给数据接收模块。

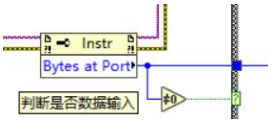

图 4-8 判断是否有数据输入

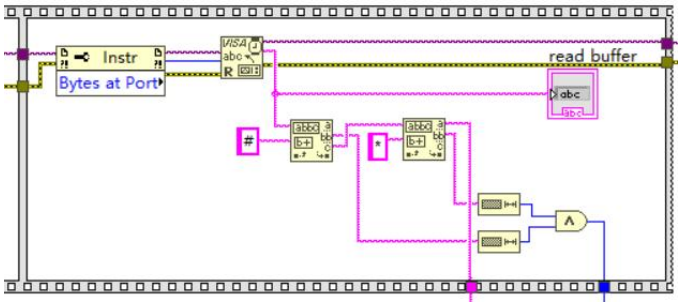

图 4-9 串口数据读取

数据接收模块根据接收帧中功能位的不同，将剔除掉功能位的字符串移交给相应功能的接收框图进行处理。

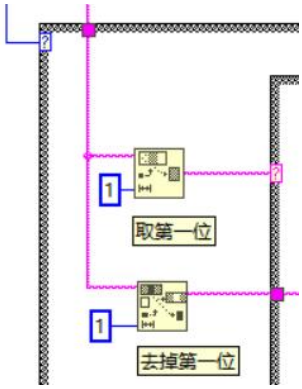

图 4-10 功能位判断

数据接收模块根据反馈帧功能位的不同共设置了 6 个接收框图。

（1）允许入网反馈帧接收模块

当反馈帧中功能位为‘/’时，PC 端将收到的信息打印在“提示信息”字符串中，告知用户终端的入网情况。

入网情况有两种：

①若终端已被允许入网，则显示“addr X :permitted already(erro)”

②若终端成功被允许入网，则显示“addr X :permitted successfully(erro)”

图 4-11 允许入网反馈帧接收模块

（2）禁止入网反馈帧接收模块

当反馈帧中功能位为‘0’时，PC 端将收到的信息打印在“提示信息”字符串中，告知用户终端的退网情况。

退网情况有三种：

①若终端不在网，则显示“addr X :off net(erro)”

②若禁止入网功能实现超时，则显示“addr X: logout-response timed out (erro)”

③若终端成功被禁止入网，则显示“addr X: successfully logout”

图 4-12 禁止入网反馈帧接收模块

（3）Link 终端连接建立反馈帧接收模块

当反馈帧中功能位为‘2’时，PC 端将接收到的信息打印在“提示信息”字符串中，告知用户 PC 端与 Link 终端连接情况。

连接情况有三种：

①若 PC 端与 Link 终端连接成功，则显示“addr X :successfully linked”

②若连接建立功能实现超时，则显示“addr X: Link-response timed out(erro)”

③若 Link 终端不在网，则显示“addr X :off net(erro)”

图 4-13 Link 终端建立连接反馈帧接收模块

（4）采集数据反馈帧接收模块

当反馈帧中功能位为‘3’时，PC 端将接收到的数据信息以曲线的形式呈现给用户。

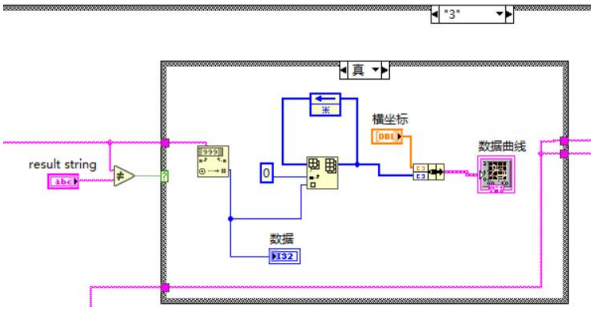

图 4-14 采集数据反馈帧接收模块

（5）在网状态表接收模块

当反馈帧中功能位为‘5’时，PC 端将接收到的在网终端地址以及每个终端对应的用户以行的形式依次加入到表格中。

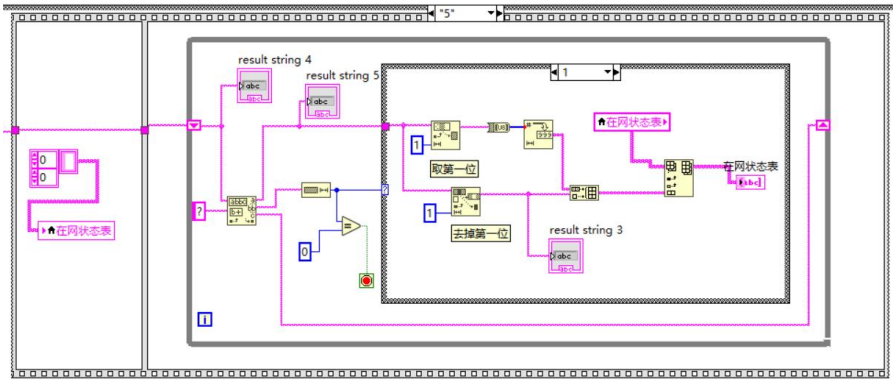

图 4-15 在网状态表接收模块

（6）黑名单接收模块

当反馈帧中功能位为‘6’时，PC 端将接收到的黑名单地址以行的形式依次加入到表格中。

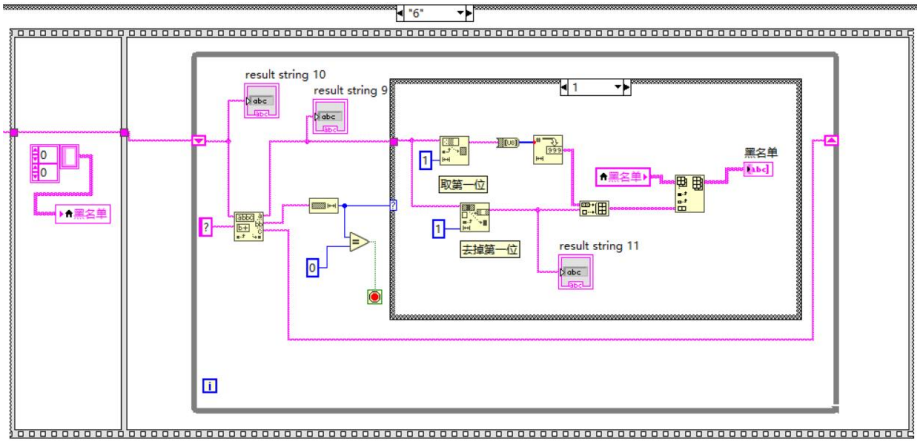

图 4-16 黑名单接收模块

4.3 前面板设计

本次的前面板设计以简洁和规整两个主题为设计目标。

（1）简洁：尽量采用浅色的方块图标以给予用户干净清澈的感官体验。每个图标间都预留了适当的空间，使得整个前面板在视觉上宽敞大方，且各个功能的按键分布直观显眼。

（2）规整：相同类型的图标皆设置为等高等宽，每个图标间的列间距和行间距都保持着绝对一致。将功能按键、数据曲线、表格、信息栏用明显的细线分隔开，使得前面板各个板块分工明确，便捷了用户对所需功能的寻找。

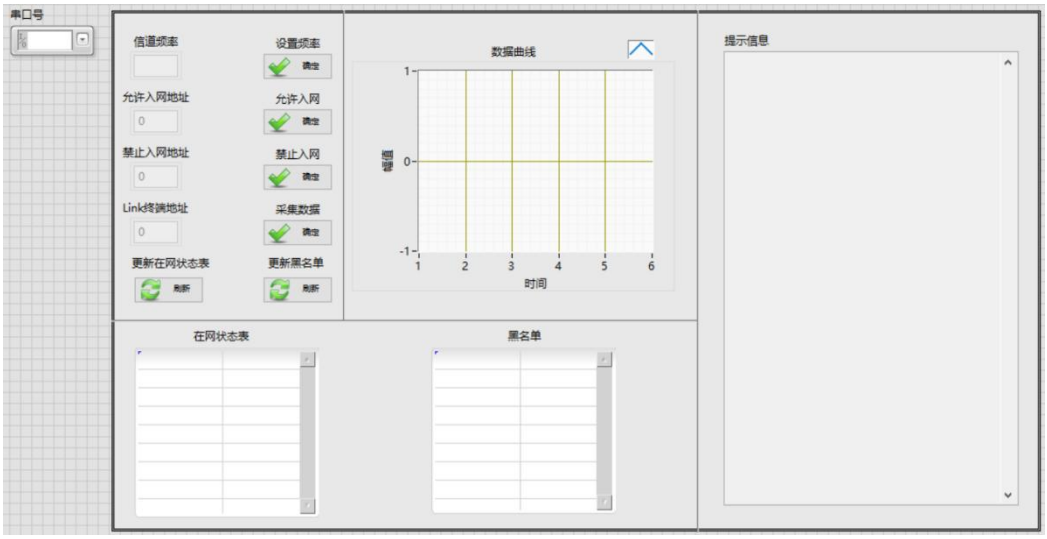

图 4-17 前面板设计

5.联合调试与总体测试

5.1 入网调试

如图 5-1 所示是无用户接入网络时的控制面板图，此时在网状态表、黑名单均未显示。控制面板模块具有的功能包含设置通信频率、入网控制、退网控制、采集终端发送的数据、更新在网终端状态以及不允许某个终端入网的功能，右侧的旁白可以显示提示信息，是对目前的状态的说明。采集终端数据时可以将其收到的数据进行存储，自动绘制其数据曲线。

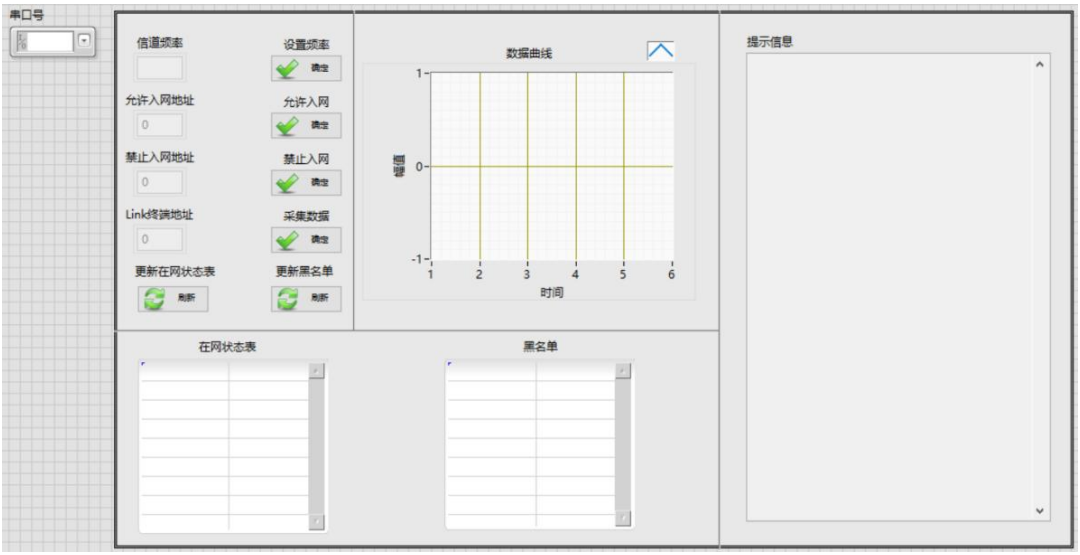

图 5-1 无用户入网的控制面板图

当有某一个终端要进行入网时，用户通过串口向终端发送“connect”，如图 5-2 和图 5-3 所示，则终端进入广播帧监听状态，当收到广播帧就向该广播帧的源地址发送入网请求，主节点收到入网请求后，确定终端不在黑名单则发送 ACK 帧，终端节点也成功收到 ACK 帧则一次接入过程成功。如果终端节点收到了广播帧并发送了入网请求却超时未收到 ACK 帧，则终端也会通过串口及时显示接入失败情况，此时有两种可能，一是入网请求帧丢失或 ACK 帧丢失，二是该终端已被加入黑名单。则用户可多次发送连接命令，若多次显示接入失败则可确定自己是被加入了黑名单。

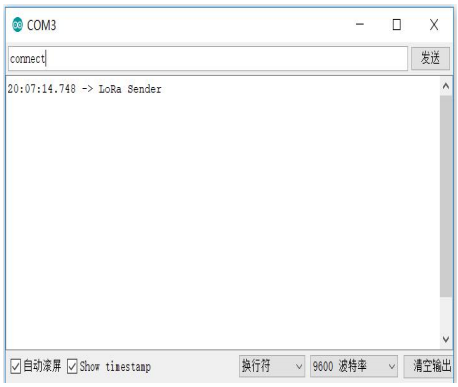

图 5-2 用户端口发送入网请求

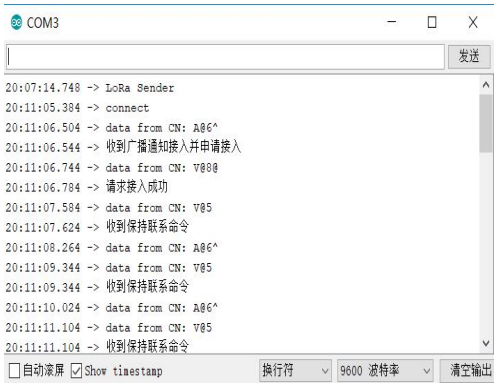

图 5-3 用户端口请求入网成功

此时的控制面板模块的在网状态表会自动更新在网的终端，点击刷新按钮会显示目前在网的终端地址以及用户名，如图 5-4 所示。当多个设备请求入网时，在网状态表也能自动检查状态，点击刷新在网状态表便自动更新，如图 5-5 所示，五个终端同时入网时的状态，表示五个终端都已经接入网络。

图 5-4 一个终端入网

图 5-5 五个终端同时入网

5.2 退网调试

当终端主动要求退网时，用户在串口输入“disconnect”，则终端不再响应主节点的命令了，如图 5-6 所示。主节点多次访问该终端都未收到反馈则认为该终端退网，从接入链表中删除（需要一定时间，不会立刻删除），随后点击控制面板的刷新会显示目前在线的终端，其余便已经退网，如 5-7 所示，地址为 88、89、80 的终端都已退网，在线状态表只有地址为 80、85 的终端在线。

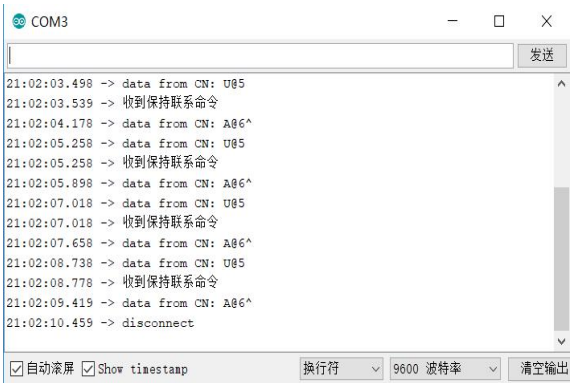

图 5-6 终端请求退网

图 5-7 目前在线的终端

当 PC 端要求某个终端退网时，可在控制面板的退网地址中输入要求退网的终端地址，然后按下退网按钮，这个地址的终端就自动退网，并加入“黑名单”，就是说没有主节点给与此终端节点权限，此终端节点将不能在接入网络；点击在网状态表和“黑名单”的刷新按钮，页面会显示结果，如图 5-8 所示。

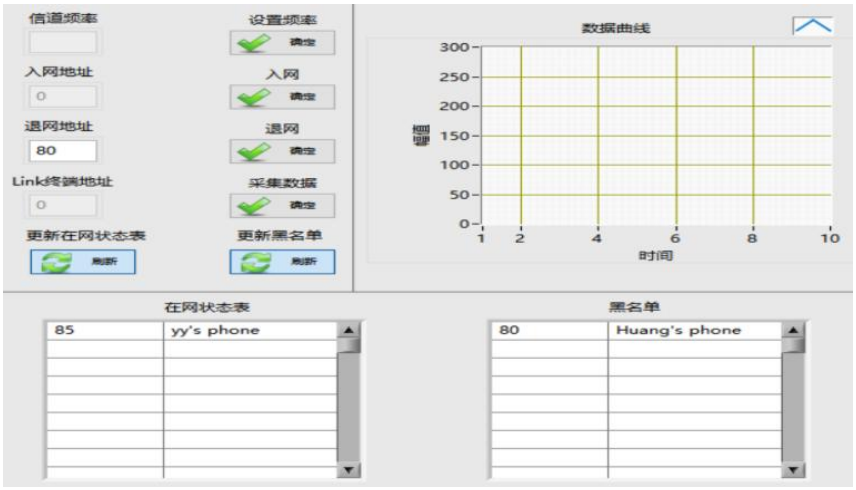

图 5-8 某个终端节点被退网之后的状态

如图 5-8 所示，地址为 80 的终端节点被要求退网时，被自动放入黑名单，更新状态后的状态表如图。

当主节点允许某个节点重新入网时，跟上述方法类似，输入允许入网地址并按下按钮，此地址的终端节点便有权限连接网络，从“黑名单”中移除，终端节点发送“connect”重新进行入网，如图 5-9 所示，地址为 80 的地址被重新允许入网，从黑名单里面移除。如 5-10 所示旁白第一条“addr 80:successfully logout”表示刚才将地址为 80 的节点退网，第二条“addr 80:permited successfully”表示将其重新允许入网。

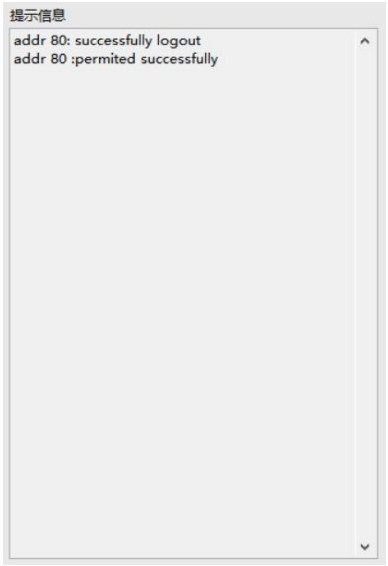

图 5-9 某个终端节点被允许入网

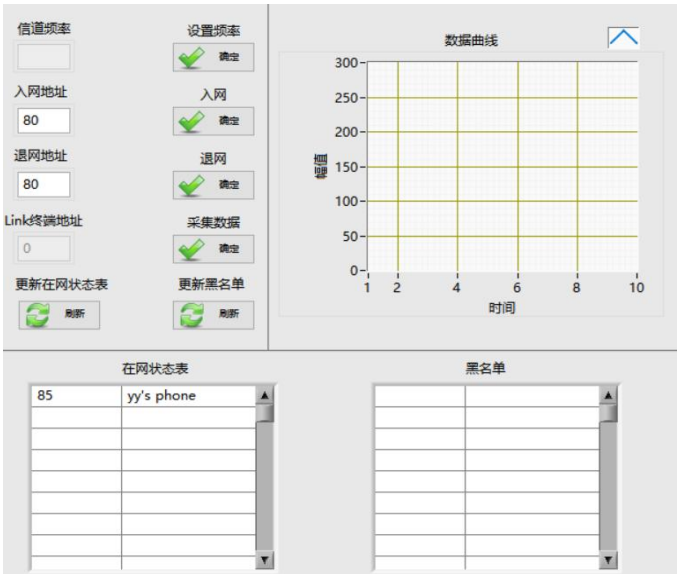

图 5-10 旁白信息

5.3 数据采集

如果想采集某个终端节点发来的数据，可以在控制面板的的 Link 终端的地址内输入其地址，并点击采集数据的按钮，主节点数据将被自动存储，一旁的数据曲线便可自动进行绘，旁白将出现“addr xx :successfully linked”；若某个终端节点未连上网，要

去采集其数据，旁白将出现“addr xx:off net(erro)”，如图 5-11 和图 5-12 所示。

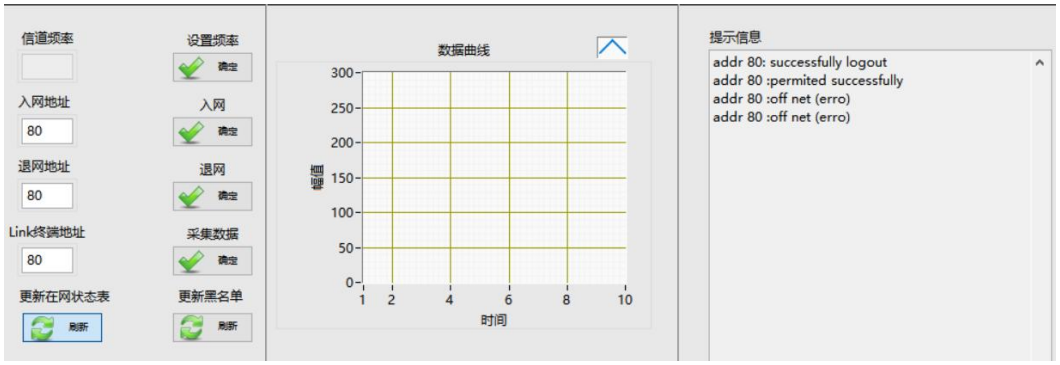

图 5-11 地址为 80 的节点不能检测数据

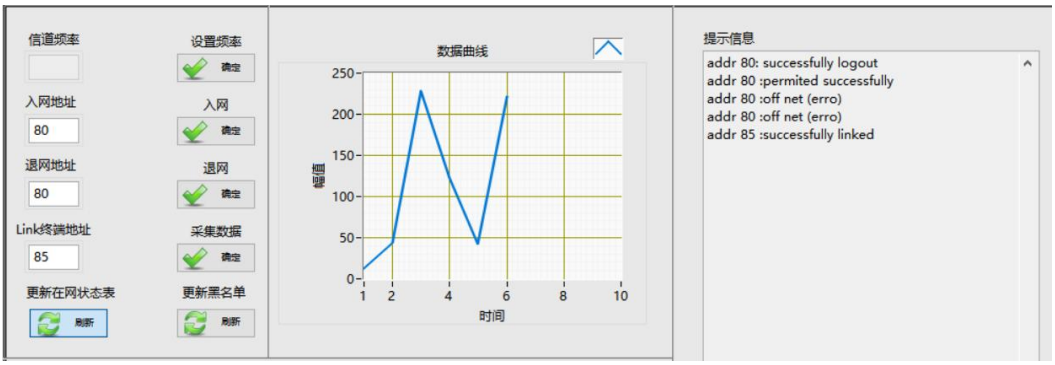

图 5-12 地址为 85 的节点能检测数据

图 5-13 此时的在网状态表

因为此时只有地址为 85 的终端节点在网，因此只有检测此节点的数据。

5.4 更改信道频率

在控制面板上的信道频率内输入想改的目的频率，点击其设置频率，便可实现更改信道频率的功能。

图 5-14 信道频率的更改

心得体会

实验总结一
持续三周的通信系统综合设计与实践课程设计结束了，这是我的第一个以小组形式进行的课程设计，我的工作主要是编写各个通信功能模块的程序代码。首先我们组先在一起讨论出了总体的方案设计，有了整体的框架后，就开始具体的进行功能模块的实现。 首先要先组建一个星型无线网络，我们采取了中心节点向一个公共地址广播的形式，端节点收到则采用退避算法向中心节点发送接入申请，中心节点收到后再返回一个确认报文。编写代码前要保证思路清晰，而本次课程设计的程序体量还是很大的，因此只有思路清晰，知道每一步要干什么，程序编写起来才会不混乱。入网后，我设置了一个 Link 布尔型变量来表示是否建立了联系，只有建立了联系才能采集数据和实现中心节点转发端节点发来的数据，与入网和建立联系对应的又编程实现了退网和入网功能。为了实现信道频率可调，我又编写的更改频率功能的子功能模块，建立联系后的采集数据部分也是一个单独的功能模块，一共有八个功能模块，通过通信报文中的功能位加以区分。 值得一提的是，在本次课程设计的代码编写中，我们大量使用了链表这一链式存储类型。因为在网络动态维护中，需要不停的插入和删除一些端节点，而使用链表进行插入和删除元素的效率很高。同时各种通信帧结构可以通过结构体链表的形式封装起来，提升了程序的可读性和简洁性。本次课程设计让我受益匪浅的还有一个很重要的点是增强了我与他人之间合作交流的能力，以前的课程设计都是个人单独的任务，思路想法都是自己的，而本次课程设计不仅需要有自己的思路，还需要了解队友的思路，就像我在编写程序时也需要连接做前端 Labview 的同学准备怎么进行帧解析，我在编写程序的时候就要尽量采用与之对应的格式。 作为一名通信工程专业的学生，亲手搭建一个规模较大的通信系统是一件很有成就感的事情，通过本次课程实践让我直观领略到了通信的魅力，同时更加真实的感受了通信协议的重要性，将以前学过的许多理论知识很好的在实践中运用了起来。

实验总结二
作为一名工科学院的学生，每一次实验和课程设计都要认真对待，因为只学习理论课是不足够的，必须将学习的知识运用到实践之中，才能有更多的体会，总的来说，这次课程设计让我学到了很多东西。 由于上学期寒假有做过数字接口课设和通信系统课设的基础，因此本次课程设计的内容并没有引入新的软件，但是很多东西都已忘记，所以要复习以前的内容，并且提升对软件的熟悉度。本次课程设计大致可以分为三个部分：通信协议设计和顶层框架设计，各个模块的设计以及模块间的联合调试。和小组一起协议设计、以及完成中心节点和终端算法的编程实现。首先，第一个问题就是如何设计数据帧结构，通过搜集各种资料有了一定的了解后，随后就是如何对中心节点以及端节点如何编程实现对应的功能，最后便是如何与 Labview 如何配合使用，难度都是不小的。 在是实现过程中遇到过两个影响非常严重的问题。一、在不存在其它板子干扰的情况下，进行中心节点和终端联合调试的过程中总是出现终端板发送报文消息但是中心节点收不到相应报文。起初以为是板子的问题，更换板子之后问题也间歇性出现。通过查阅相关资料，发现 LoRa 接收和发送存在一定的转换时间，所以这个看似很严重的问题只需要在接收和发送之间加一个很短的延时即可解决。二、在联合调试时总是发现 Labview 端的数据更新很慢，并且各种指令的执行都很慢，最初以为程序写得过于冗余以及向串口打印了过多的提示字符串导致。优化程序结构已经完成相应自认为可以提高执行效率的方法之后也不见好转，问题最终解决是在调整扩展因子时将其调小，发现速度提升了不少。 课程设计对我来说是一次小测验，开头的时候确实是比较困难，但是只要自己有信心解决这些问题，和小组共同探讨，也增进了同学之间的交流，也完成了老师给定的任务，即使最后的结果不是完美的，但是要享受过程，这是最令人印象深刻的。在过程中不断找到自己问题，不断解决，自己能看到自己能力的不足，并渐渐提高，这并不是什么坏事。 总之，本次课程设计收获颇多，也谢谢老师们的指导指正。

实验总结三
在本次课程设计中，我负责用 LabVIEW 实现 PC 端与主节点间的串口通信以及课设报告的整合与排版。 此次任务的完成要点是串口处数据的写入与读取。首先是写入部分，我需要

与 Arduino 代码编写的同学确定好数据帧的格式，再按照我们确定好的格式将数据采用字符串或者数字转字符串的形式写入到串口，然后主节点再按照我们规定的格式依次解析便可以得出我写入数据帧的各个部分。

写入部分的完成较为轻松，难点更多在于读取部分的设计。

由于功能的多样性导致了 PC 端读取的数据帧中数据部分长度与内容各不相同，因此如何用同样的一个模块从不同形式的数据帧提取出数据部分（我们所需要的部分）是首要难题。最早的方案采用的是字符串转字节之后通过拆分再拼接的方式提取出数据部分，较为繁琐，但在我们小组的整体课设方案改进后，提取数据部分便只需要剔除掉帧头帧尾以及功能位就够了。由此可以得出，项目整体架构的设计对其中各模块的难易实现有很大的影响。

第二个难点在于对各个功能的实现。我们小组设计了 6 种功能，因此在 LabVIEW 上我需要设计 6 种功能的实现。印象最深的一个障碍是数据曲线的绘制不具有实时性。我的解决办法是将采集数据与横坐标进行数组捆绑，再结合一起作为 XY 图的输入。

除了程序框图实现的难点之外，我在美观设计也投入了很多心思。

以“在网状态表”与“黑名单”为例。起初的设计是直接采用字符串进行显示，但前面板的输出现实情况较为拥挤且识别不易，因此后续更改为了在表格中显示，大幅度提升了数据的直观性与可分辨性。

总结，本次课程设计是一次团队任务，不仅需要个体的技术支持，还需要组员间的配合与搭建。首先我们应当搭建好整个项目的框架，再将架构拆分成各个模块交给不同的成员设计与实现，在过程中我们应当多交流以防止各个模块间的功能不互通，最后再将各个成员的模块联合起来进行局部或者整体调试，从而完成项目。

实验总结四

在本次实践中，我主要负责整体方案的设计、Arduino 代码编写以及参与 Labview 设计和整体系统调试等。在这个过程中，我遇到了许多问题，但在老师的指导下以及不懈的努力下问题最终都一一解决。

在设计方案上，我们想适应不定多个终端随机接入的情况，而不是只维护固定多个终端的系统，固定终端的情况在主节点中可以采用数组穷举各个节点的状态参量，在 Labview 设计的前面板上也可以穷举各个节点的状态。但我们设想的情况是终端地址不定，终端接入个数不定，以上方案不再合适，所以我们在主节点中用链表记录接入的终端的信息，在前面板上采用用户输入地址的方式查询任意终端的信息。

前期我们没有仔细思考系统最终呈现出的效果，因此在设计的过程中涌现出很多种设计方案。

1）组网过程不受 PC 控制的方案，由主节点广播可接入信号，附近终端节点收到广播帧，发送请求接入帧，帧中数据为接入密码，主节点判断密码正确，即将该终端地址加入接入链表，接入成功，主节点实时维护网络拓扑记录。PC 可通过下发命令控制相应终端退网（从接入链表中删除）

问题：PC 下发退网命令后，相应终端成功退网，但下次广播帧该终端又立刻接入了进来，整个过程中，该终端可能退网只有几十毫秒，不符合预期效果。

2）采用组网过程完全受 PC 控制的方案，PC 端下发命令广播可接入信号，更新一次网络拓扑，可解决 1）中的问题。

问题：不符合应用场景。

3）采用组网过程不受 PC 控制的方案，在 1）的基础上，设计一个黑名单链表，将主机要求退网的终端加入黑名单，主节点在接收请求接入帧时，不处理黑名单中终端的帧。可解决 1）中的问题。

问题：进入黑名单的终端每次收到广播帧都会发送接入请求，而主节点不接受，这样就不仅使终端耗电量增大，还加大了建立关联时碰撞的可能。

4）在 1）的基础上，使终端自身维持一个 bool 型变量 Acs_Permit，收到退网命令后，Acs_Permit 置 0，则以后收到主节点的广播帧都不再发送入网请求。收到主节点的允许入网命令后 Acs_Permit 置 1，则可发送入网请求。

问题：实用场景中，主节点入网允许命令如果丢失，那么主节点认为自己已经邀请了该终端，而该终端仍认为自己不被允许，这就导致该终端永远也不会再加入该网络了。

经过反思，我们最终认为是因为整个系统的实用场景一直没有得到准确的确

定，所以设计方案一直模棱两可，因此结合 3）和 4）的方案，我们最后确定出了最终的版本，即模拟热点接入的方案。既然模拟热点接入的方案，那主节点除了知道已接入设备的地址，同时还应该知道接入设备的名称。因此我们在链表中增加了成员变量 Name。超时重传部分我们没有采用一直连续超时重传的方式，我们的系统中终端数目不定，如果出现同时若干个终端掉线，那么在主节点在一次 loop 中就要耗费很长的时间，不仅不能及时响应 PC 端的命令，另一方面也可能使其他在网终端不能及时收到“保持联系命令”，所以我们采用每次 loop 中对一个终端顶多超时重传一次的方式，使一次 loop 的时间尽量稳定。

在单片机编程上，通过了解 IDE 提供哪些库函数可以大大简化编程过程，比如 Arduino 针对字符串的操作提供了非常丰富且方便调用的函数，因此在组装帧和解析帧的过程中我们都采用操作字符串的方法。我们以字节为单位，组装和解析帧，组装的过程中，可以选用字符操作和字节操作，因此这个过程中一定要清晰的了解什么时候用字符函数什么时候用字节函数，以及两者之间的转换关系，如 print 函数和 write 函数的区别，另外，起初查阅资料的过程中，了解到 char 类型和 byte 类型的数据的区别只是一个为带符号八位数，一个为不带符号的八位数，但 Aduino 的一些函数对 char 类型和 byte 类型的参数调用结果却有极大的不同，导致程序无法运行出理想结果。并且字符串中无法表示一个十六进制为 0x00 的字符，因此我们想要发送 0x00，必须独立出字符串单独发送。Labview 设计中同样需要注意字节和字符的区别，且应注意两边协议的同步。

另外，由于 Arduino 软件不提供调试功能，所以在程序调试的过程中极大的考验了我们的耐心，我们采用的方法是在不同的位置调用 Serial.print( )函数确定程序运行到了哪里，比如一次我们在 if 语句中判断接收字符串是否为 CHAR_ACK（define 为‘@’），但是发现即使接收字符串打印结果为‘@’仍然无法进入 if 语句，最后意识到原因是 String 类型和 char 类型不应该进行比较。一次在 if 语句判断中同时使用&和==操作符，但由于没加括号导致程序无法进入 if 语句，最后发现 &与运算符的优先级低于==相等运算符。还有一次因为在 if 语句中将==写成=，导致程序无法运行出理想效果，最后也耐心的发现并解决了问题。还有一些编程语言基础问题，如定义了野指针后未指向任何变量就去使用，花了很长时间找 bug。

在 LabVIEW 设计上，由于前期和代码这边没有对接好，后来对接的时候发现了许多地方都要重新修改，并且调试过程中发现目标效果和实际差距甚远，如在读取串口时，我们前期通过判断串口有无数据触发读取串口，若有数据则立刻读取，发现每次读取到的数据都比发的数据少几个字符，第二次读取才能读取完串口，后来我们在读取前延时 50ms，等待主节点发送的数据全部到达串口后再

一次性读取，同时主节点这边的代码也应该保证发送帧的时间间隔不能太短。

通过这次课设我收获到了许多，也发现了自身的许多问题，如做事太盲目，缺乏大局观，细节把控不够到位等。缺乏实践能力以及工程上的应用能力，想当然地设计方案而不去了解设备参数及其实际应用场景等。另外虽然前期多次接触代码编写，阅读过许多优秀的程序，但通过本次课设我有机会将编程能力应用到实际中并实现自己的想法，同时也意识到自己的不足并实现了超越自己的进步。

实验总结五
本次课程设计我们设计了一个低速星型无线通信系统，可以实现终端节点灵活控制、实时数据灵活监测、系统各部分间可靠通信、信道参数灵活设置，同时在不需要监测相关数据时，可以控制节点出网并停止采集信息，减少能耗，可以用于低功耗的监测系统。 由于此前我的 LabVIEW 设计基础较为薄弱，本次课程设计中我遇到了很多困难。以前在做通信原理实验和课程设计时，我对 LabVIEW 的理解其实并不深入，很多功能都不太会使用，LabVTEW 使用的一些技巧也没有学握。为了能够尽快熟练使用 LabVIEW，一开始我参考网上的资源如 CSDN 学习，但学得较慢，有些模块的使用还不是通透。后来我查看了 LabVIEW 的帮助和例程，里面讲得很详细全面，而且有着丰富的实例帮助理解，大大加快了我学习理解的速度。同时对于一些帮助文档里讲得比较复杂不大清晰的模块。 而在本次课程设计中的最难点在于 PC 端、中心节点以及三个终端节点间的联合调试，可以说一半时间在设计程序框图，一半时间联调修改。联调修改的往往是逻辑错误，也就是 LabVIEW 运行得了但结果不符合预期。 总的来说，通过这次课程设计，我对 LabVIEW 的使用更加熟练，学会了使用 LabVIEW 进行程序化设计，在这个过程中我我到了尽快熟悉一款软件的方法，系统联合调试的经验也更加丰富，最终完成了课程设计的任务。

参 考 文 献

[1] 陈吕州.Arduino 程序设计基础（第 2 版）. 北京：北京航空航天大学出版社，2015.

[2] 沈金鑫.Arduino 与 LabVIEW 开发实战. 北京：机械工业出版社，2014.

[3] 李培毅.Arduino 实验案例指导手册（简版 0917）.