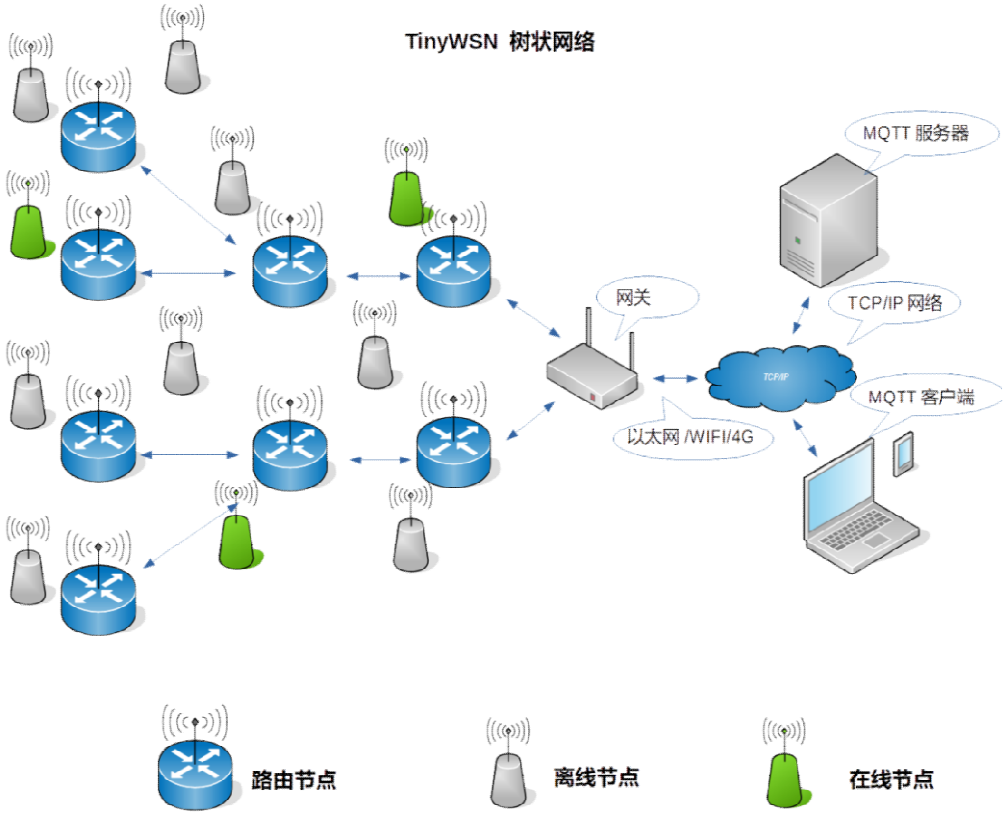
TinyWSN 快速入门指南

文档作者 tzhong
2023/5/1

目录

产品概述

TinyWSN（Tiny Wireless Sensor Network）是一套 Sub-1G 的无线传感器网络产品，提供一种低功耗，远距离，大容量，安全可靠的通信方式，支持星型，链型和树型等多种组网方式。下图就是它的一个典型树型网络组网：

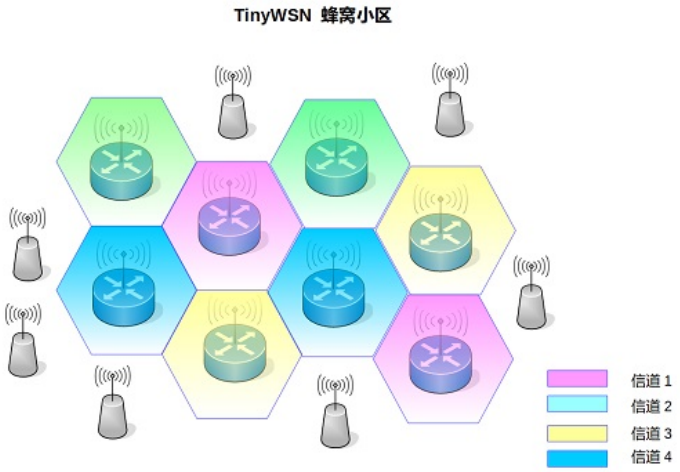


主要由节点和网关组成，节点有路由和终端两种功能角色，终端又有在线和离线两种工作模式，但它们都是由相同的节点模块经过配置而成，网关根据实际需要进行选配，具体可参见**《TinyWSN 网关使用手册》**，本文档使用一个简单的网络管理软件 TinyWSN 来代替网关，简化教程环境搭建，降低学习难度。

节点功能表

功能角色	工作模式	说明
路由节点	RTT	只处理上行无线，在穿越障碍的应用中
	RTB	只处理下行无线，例如在网关中的模块
	RTU	处理上下行无线，典型的路由应用中
终端节点	在线	工作在同步模式，在分配的时隙中进行收发，空闲时进入睡眠，降低系统功耗，优点是速度快,空闲功耗小于 2uA。
	离线	工作在异步模式，周期侦听无线唤醒帧，可以由网络侧唤醒，也可以由传感器的告警唤醒，或者外部命令输入唤醒，或者自定义周期唤醒，转入在线状态，优点是功耗低,平均功耗小于 3uA。

终端节点在两种工作模式下，都支持双向通信，可以由节点发起或者网络发起。每个网络预先分配一组频率资源，路由节点根据扫描信道的结果，动态优选干净信道，最后形成一个蜂窝小区：



通常传感器网络具有以下的特点：

- ✓ 节点数量多，无线容易冲突和干扰
- ✓ 节点流量低，例如几个小时发一条消息
- ✓ 节点功耗低，电池需要支持十几年以上

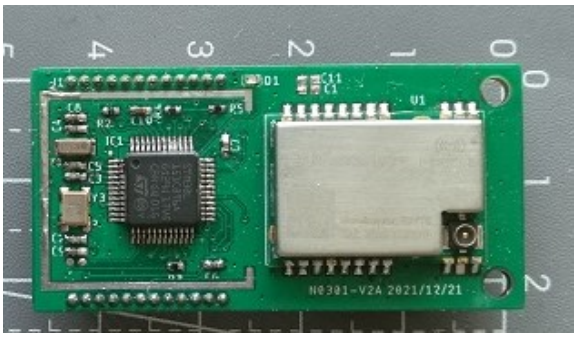
TinyWSN 就是为这类应用而设计的无线传感器网络，其中在线节点工作在同步模式中，只在分配的时隙进行收发，相互之间避免了冲突和干扰，而大部分终端节点是离线节点，这样可以节省无线资源，使得能够支持更多的节点，也使得节点功耗更低。

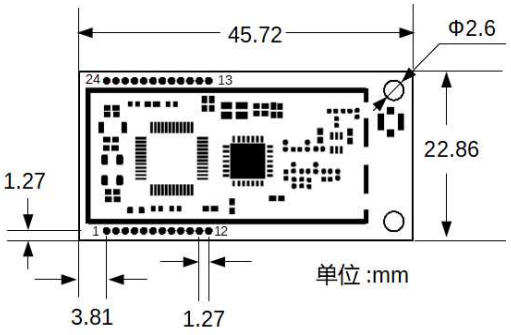
产品特性

名称	描述
频率范围	Sub-1G
频率复用	时分频分
工作模式	在线模式，离线模式
网络拓扑	树型，星型，链型
级联数目	255
小区节点	120
网络节点	65535
通信距离	500 米~几公里，取决于节点型号
收发功耗	24mA~180mA，取决于节点型号
空闲功耗	2uA
离线功耗	3uA
功率控制	动态控制
信道分配	动态分配
无线唤醒	支持
用户接口	交互命令，系统报文，嵌入模式
升级方式	串口升级，无线升级
安全机制	接入密码，会话密码，动态加密，TLS/SSL
网关接口	以太网，WIFI，4G
工作温度	-40 度~+85 度

节点模块

节点模块由处理器和射频模块组成，如下图所示

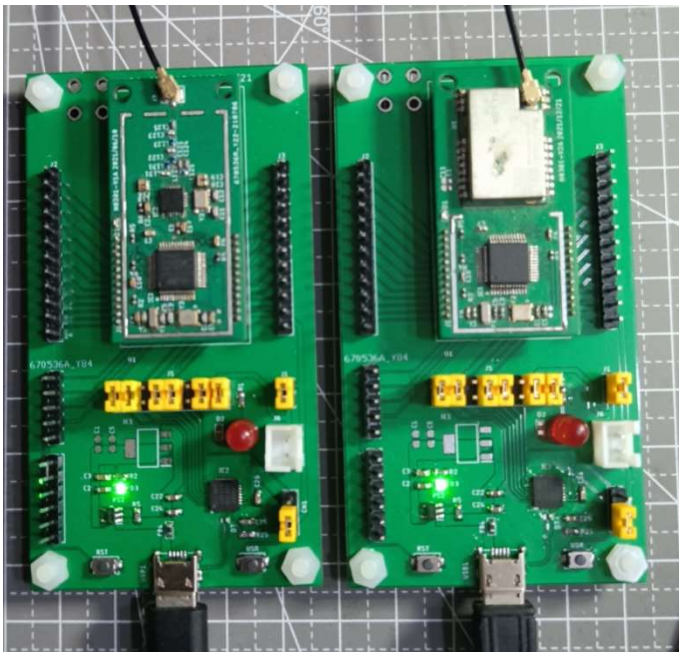




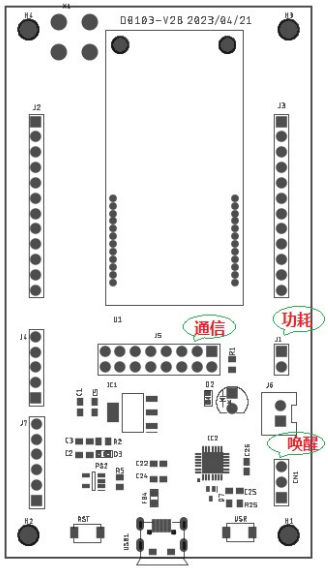
处理器实现 TinyWSN 的无线协议部分，对外留有两排间距 1.27 引脚。

编号	名称	类别	方向	描述
1	GND	电源	输入	电源
2	VCC	电源	输入	电源
3	RXD	系统	输入	UART 数据接受
4	TXD	系统	输出	UART 数据发送
5	STA_IND	系统	输出	内部状态输出
6	USR_PKT	系统	输入	接口选择
7	PB10	用户	可配	用户自定义
8	PB11	用户	可配	用户自定义
9	PB12	用户	可配	用户自定义
10	PB13	用户	可配	用户自定义
11	PB14	用户	可配	用户自定义
12	PB15	用户	可配	用户自定义
13	SWDIO	系统	双向	下载调试接口
14	SWCLK	系统	输入	下载调试接口
15	NRST	系统	输入	下载调试接口
16	STA_INP	系统	输入	外部状态输入
17	NET_IND	系统	输出	网络状态指示
18	WKU_IND	系统	输出	睡眠唤醒输出
19	WKU_INP	系统	输入	睡眠唤醒输入
20	PB6	用户	可配	用户自定义
21	PB7	用户	可配	用户自定义
22	PB8	用户	可配	用户自定义
23	PB9	用户	可配	用户自定义
24	TBD	系统	输入	未定义

除了模块供电引脚， 一部分是系统定义的，另外一部分是在内嵌应用中由用户自定义的。下图是模块的开发板，方便用户对模块的功能和功耗进行评估，



开发板中 USB 转串口芯片 CP2104，连接到模块的 UART，还利用它的流控信号来控制模块



和读取状态，同时还提供功耗测试接口 J1。

J5 用于通信控制

组号	模块信号	底板信号	默认安装	描述
1	PB7	LED	否	用户 LED 控制
2	模块电源	系统电源	是	模块供电
3	TBD	RTS	否	待定义
4	UART_TX	RX	是	串口发送
5	UART_RX	TX	是	串口接受
6	STA_IND	DSR	是	内部状态输出,默认是接受队列的状态，置位

				时表示接受队列无待发送的消息。
7	USR_PKT	DTR	是	串口接口选择，置位时表示选择系统报文，复位时则选择交互命令
8	NET_IND	DCD	是	网络的连接状态，置位时表示正在接入，复位表示连网成功。

CN1 用于睡眠控制

管脚连接	默认安装	描述
1，2	否	禁止睡眠，用于调试模式
2，3	否	RTS 控制， 可由应用控制，复位时禁止睡眠，置位时允许睡眠，主要用串口数据输入。当串口输入命令时，需提前 5mS 唤醒模块，数据输入结束后，再允许睡眠
悬空	是	允许睡眠，内嵌模式时使用，或者功耗测量时，当串口没有打开时，RTS 默认信号会禁止模块进入睡眠

目前许多串口终端软件，例如 RealTerm，串口助手 Xcom32 等等，都可以直接设置和读取流控信号，这样就可以利用它，设置模块的工作模式和读取它的工作状态，方便调试和开发。

为了满足不同用户需要，节点模块提供了三种使用方式

方式	硬件接口	描述
交互命令	串口	这是最简单的一种模式，提供类似 AT 命令，可以手工输入命令，或者由程序控制，简单易学
系统报文	串口	输入有帧结构的系统报文，报文有帧校验，功能是交互命令的超集，安全可靠，详见**《TinyWSN 系统报文手册》**
内嵌模式	内存	内嵌模式的应用程序和系统程序共享处理器和内存，应用程序独立编译，链接和下载，通过共享内存和系统程序进行交互，更加高效，更低成本。详见**《TinyWSN 内嵌编程手册》**，下面是链接中提供的应用的开发环境，应用模板以及设备驱动，可供参考 https://gitee.com/tinywsn/fw-stm32l1-wbed-usr

本文档重点在交互命令的使用方式上，其他两种使用方式参见相应的手册。

节点型号

型号	主控	射频	调制
N0301	STM32L151	CC1101	GMSK
N0302	STM32L151	SI4463	GMSK
N0304	STM32L151	SX126X	LORA

注意：

- ✓ 串口电平是负电平，置位时为低电平，复位时为高电平，当连接到用户处理器需要注意这个问题
- ✓ 串口助手 SSCOM32 的 RTS 和 DTS 控制，启动后需要手工设置和清除一次，默认状态和显示不一致

交互命令

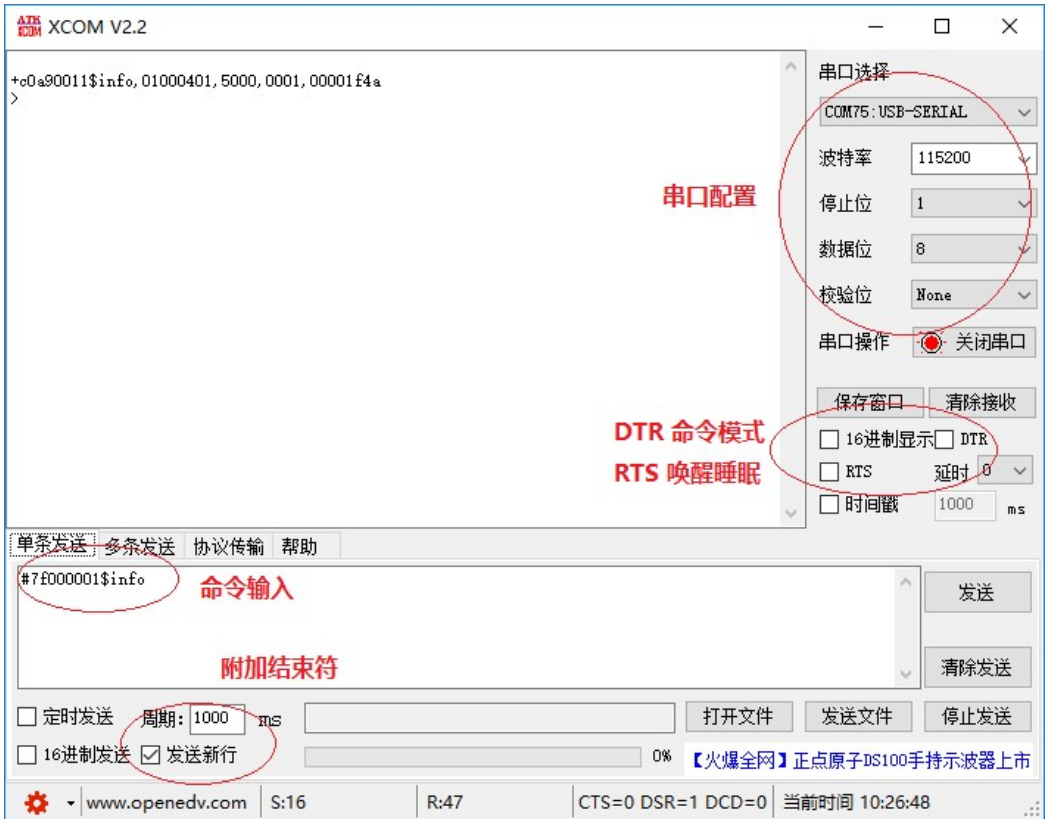
节点模块通过串口提供交互命令接口，串口设置为 115200，8N1，用户通过串口软件就可以手工输入命令，命令执行后模块会输出结果，例如下面就是一个交互过程，

```
>#7f000001$info
+c0a90012$info,01000411,4000,0001,00000175
>
```

其中“>”示符，绿色表示输入命令，黑色是模块的输出响应，

- ✓ 模块对输入没有回显，如果需要可以打开串口软件的本地回显
- ✓ 模块不支持输入行编辑，可以使用串口软件本身的命令行编辑

下图是串口软件 XCom 配置，输入命令，点击发送，就可以看到结果



每个模块出厂时，都烧录有唯一的 32 位地址，系统保留几个特殊地址

地址	说明
0x7f000001	对端地址，用在串口连接时
0x00000001	网关地址，数据路由至外网
0xffffffff	广播地址，所以节点都接受

除了直接使用地址寻址单一模块，还支持小区广播和子网广播

寻址方式	地址	说明
单一地址	模块地址	具有特定地址的单一模块接受数据
小区广播	路由模块	路由模块收到数据后，在它的小区广播，子节点均可接受到
子网广播	路由模块	同上，当子节点有路由模块时，数据继续传播，所有后代节点均可接受到，路由模块是根节点时，子网广播就变成了全网广播

命令输入的语法格式定义如下:

#addr$type,data/option

其中#,$,/是命令的分隔字符,

对象	说明
Addr	接受者地址，可以是模块地址或特殊地址
Type	命令的类型，支持的命令见后面描述
data	命令的数据，格式由具体的命令确定
option	命令的选项，对命令进行额外的控制

命令选项包含字符：bfpdnxu， 说明如下

字符	说明
B	广播消息，如果路由模块接受数据，它会在小区广播
F	继续传播，如果路由模块接受数据，它会传递到子网
P	消息应答，数据接受者会发一个应答报文给发送者
d,n,x	接受对象，用在指点广播消息中接受者类别， d 表示路由模块可接受 n 表示终端模块可接受 x 表示所有模块可接受
u	消息类别，默认消息传输是可靠传输，会进行确认和重传，u 表示无需重发，新消息还会覆盖队列中待发送消息，主要应用周期测量的消息中

在上面示例中#7f000001$info 就是一个标准命令，其中 7f000001 表示对端地址，info 表示信息查询命令，这条命令没有数据和选项

命令响应或者接受数据的语法格式：

+addr$type,data

这里的 addr 是消息发送者的地址，其他和上面是类似的， 例如网关接受的消息，它是节点 c0a90011 和 c0a90012 上报的 LED 指示灯状态的消息

```
>
 +c0a90011$data,val,led,0
 +c0a90012$data,val,led,0
```

下面是系统目前支持的命令列表,其中命令选项均被省略

功能	格式	说明
数据发送	#addr$data,payload	发送用户的数据，最长 41 字节
连通测试	#addr$ping	测试节点是否在线
唤醒节点	#addr$wkup,node[,token]	唤醒离线节点入网
通知节点	#addr$ntfy,node,payload	发送用户数据给离线节点
设置功能	#addr$setf,bits	使能位图中的系统功能
清除功能	#addr$rstf,bits	禁止位图中的系统功能
重载功能	#addr$load	恢复默认的系统功能
查询状态	#addr$info	查询系统和网络状态
复位网络	#addr$rnet	复位网络的所有节点模块
复位模块	#addr$rsys	复位串口对端的节点模块
网络暂停	#addr$halt	暂停网络，禁止节点接入
网络恢复	#addr$cont	恢复网络，允许节点接入

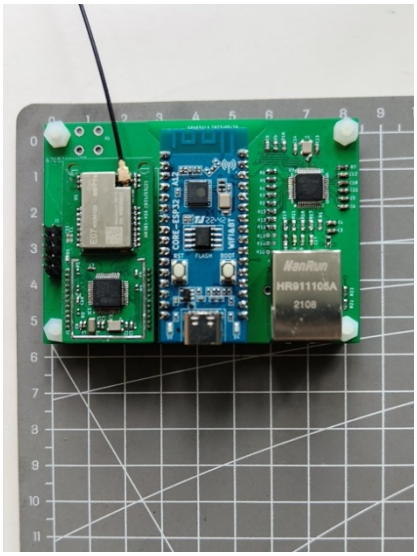
对于每条命令详解参加**《TinyWSN 交互命令手册》**。在后面介绍的网络管理 TinyWSN 软件，就是基于这个命令接口开发的，它会把所有交互的命令记录一个窗口中，可以观察到发送的命令和响应。

网关模块

目前 TinyWSN 的网关分两类

类型	操心系统	说明
MQTT	Linux	网关支持 wifi，以太网以及 4G，网关连接到 MQTT 服务器，可以直接使用 MQTT 客户端对网络进行管理，而参考的网关软件 TinyGW 使用 python 开发，它使用的系统报文接口，源码链接 https://gitee.com/tinywsn/tinygws，可以自行修改和定制。
透传型	freertos	无需开发，只需简单配置，就把网关的命令接口透传至远端

有关 TinyWSN 支持的各种网关的具体信息，可以参加下面中的介绍:
http://www.tinywsn.net/wordpress/index.php/gateway
在本教程中会使用到一种迷你的透传型网关 G0B03，它是基于 ESP32C3 开发的，160M 主频的 RISC-V 处理器，支持 WIFI，以太网以及 BLE，支持 USB 接口（串口，JTAG），有很高性价比。



它可以完成以下透传类型

类型	说明
USB	网关命令接口透传至 USB 的串口，可以无法拔下节点模块，直接对节点配置
网络	网关命令接口透传 TCP 服务器，可以连接到网络管理程序 TinyNMS
BLE	网关命令接口透传远端的 BLE 的模块，连接到网络管理程序 TinyNMS

它有一个 USB 接口，插上 PC 后，支持 USB 串口，它内嵌一个命令接口，如下所示:

> id?
TinyGW - Tiny gateway for TinyWSN, Autorun
Built Jun 11 2023 - 20:43:57 by www.tinywsn.net
>
它完成 WIFI,ETH 和 BLE 的配置和连接，以及为节点模块建立透明通道，下表就是常见命令的简要说明，具体的命令说明可参见**《TinyWSN 网关模块说明》**

命令格式	说明
nvm -c get -k key	读取配置项，系统支持的配置项参见后面的配置说明表
nvm –c set –k key –v name	写入配置项
wsn –c init –v usrpkt	无线节点接口初始化
tsh –c init –v text	网关命令接口的配置
eth –c init –o drv	初始 eth 驱动
eth –c stat –o drv	显示 eth 状态，包括 link 状态，dhcp 结果等待
eth –c conn –o cli	连接到 TCP 服务器,在配置项中定义了服务器信息
eth –c disc –o cli	断开服务器的连接
eth –c stat –o cli	显示目前连接状态
eth –c send –o cli –v data	发送数据
eth –c skpa –o cli	设置心跳周期
eth –c qkpa –o cli	查询心跳周期
wifi –c init –o drv	初始 wifi 驱动，在配置项中定义接入点的信息
wifi –c scan –o drv	开始 wifi 扫描
wifi –c stat –o drv	显示 wifi 状态
wifi –c conn –o cli	连接到 TCP 服务器
wifi –c disc –o cli	断开服务器的连接
wifi –c stat –o cli	显示目前连接状态
wifi –c send –o cli –v data	发送数据
ble –c init –o srv	初始化 BLE 服务器
ble –c stat –o srv	显示 BLE 目前状态
ble –c send –o srv –v data	发送数据
ble –c init –o cli	初始化 BLE 客户端
ble –c scan –o cli	启动 BLE 的扫描
ble –c conn –o cli	连接到 BLE 服务器
ble –c disc –o cli	断开服务器的连接
ble –c stat –o cli	显示目前连接状态

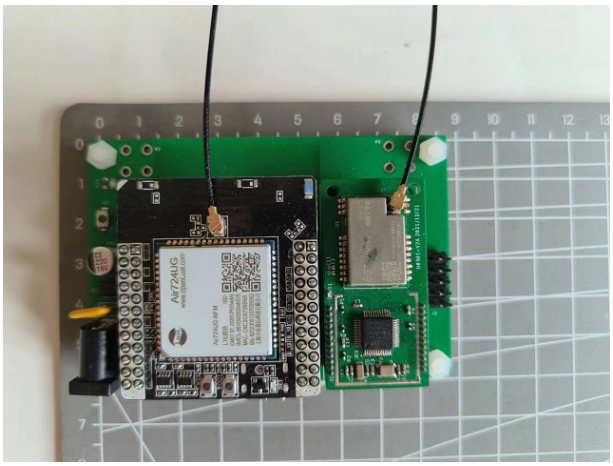
ble –c send –o cli –v data	发送数据	
ipc –c conn –s src -d dst	当上面的各个设备建立好连各自接后，就可以使用这个ipc命令在它们之间建个一个管道pipe，完成透明传输,其中 src 和 dst 可以分别选择下面对象:	
	tsh	网关命令通道
	wsn	无线节点通道
	wifi	无线连接通道
	eth	eth 连接通道
	ble	ble 连接通道
	例如要把无线节点的命令透明传输到 TinyNMS 的 TCP 服务器,先初始化 wifi 并且和 TCP 服务器建立连接，然后再运行下面: ipc -c conn -s wsn -d wifi	

配置说明

名称	示例	说明	
ssid	TP-LINK_C4XXXX	wifi 接入点的 ssid	
pass	welcome	wifi 接入点的密码	
host	192.168.3.109	TCP 服务器地址	
port	9904	TCP 服务器端口	
ctmr	8	连接建立的等待定时器	
ktmr	120	连接保持的心跳周期	
bsrv	a9:ed:12:98:00:8f	绑定的 BLE 服务器地址	
autorun	wsn2wifi	定义上电自动运行模式，在平常调试过程中，我们可以使用上面命令进行各种配置和连接，但是在现场安装后，就可以使用这个配置，指定上电自动运行的透传模式:	
		wsn2wifi	无线节点透传到 WIFI
		wsn2eth	无线节点透传到 ETH
		wsn2ble	无线节点透传到 BLE 客户端
		tsh2ble	网关命令透传到 BLE 服务器

在后面的教程中，有各种透传的具体配置实例。

下面再介绍一个 LTE 透传型网关 G0C03，它是基于合宙的 air724 模块，利用它内嵌 lua 开发的通透型网关，可以节省一个外部 MCU，系统架构简洁高效，如下图所示。



输入+5V~12V，底板上有一个 USB 转串口，它也内嵌一个命令行接口，用于系统配置：

Welcome to Luat Console

>print(nvm.get("host"))

45.32.85.175

>

它的命令格式其实就是 lua 语句，目前主要用于 参数配置

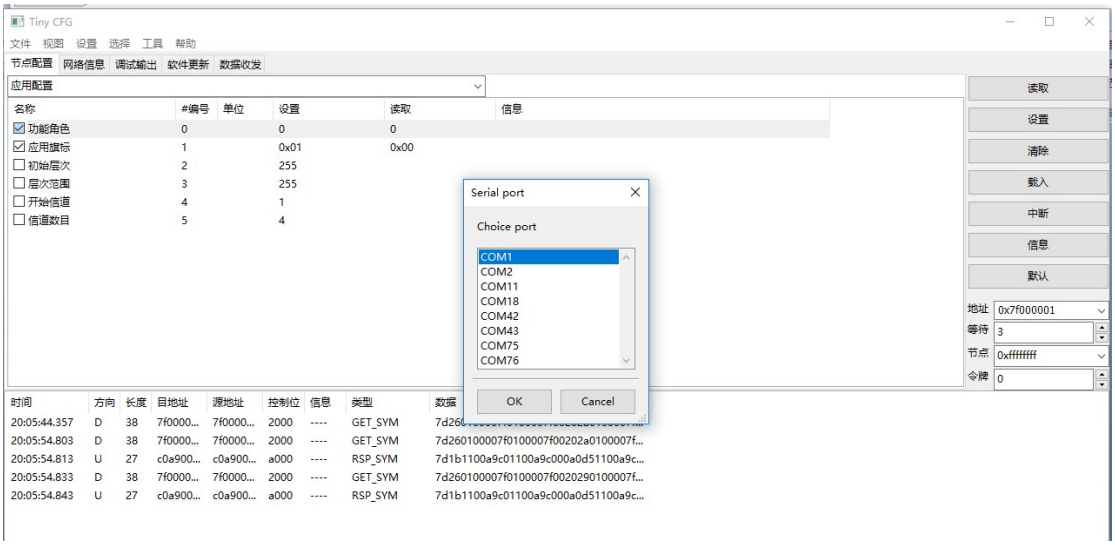
命令格式	说明
nvm.set("name", "val")	设置参数
print(nvm.get("name"))	获取参数

目前支持的配置参数

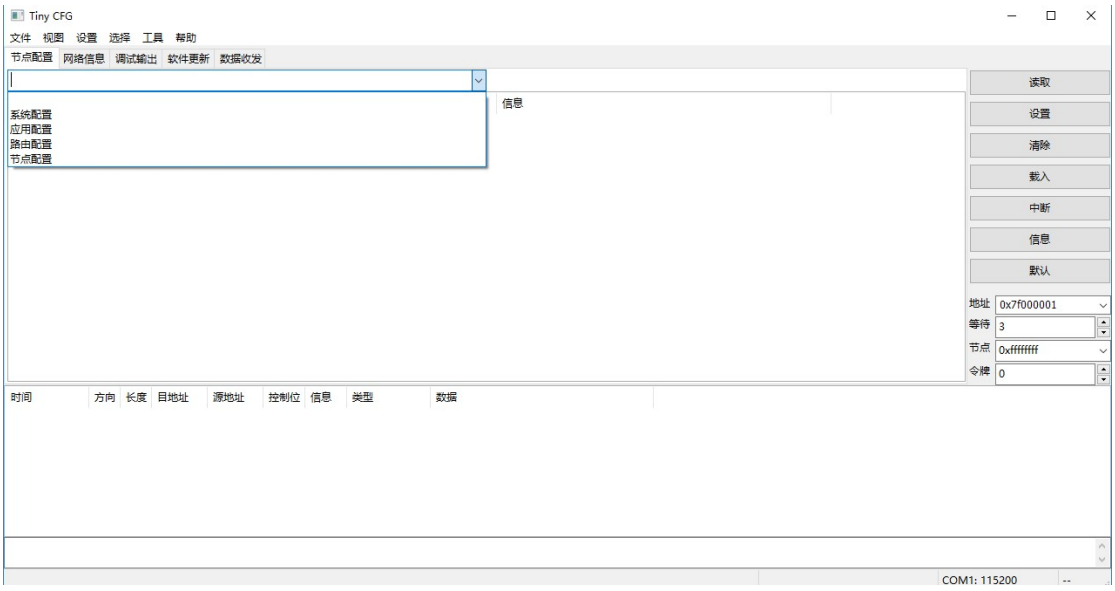
名称	示例	说明
host	"35.32.85.115"	TCP 服务器地址
port	"9999"	TCP 服务器端口
ktmr	120	连接保持的心跳周期

节点配置-TinyCFG

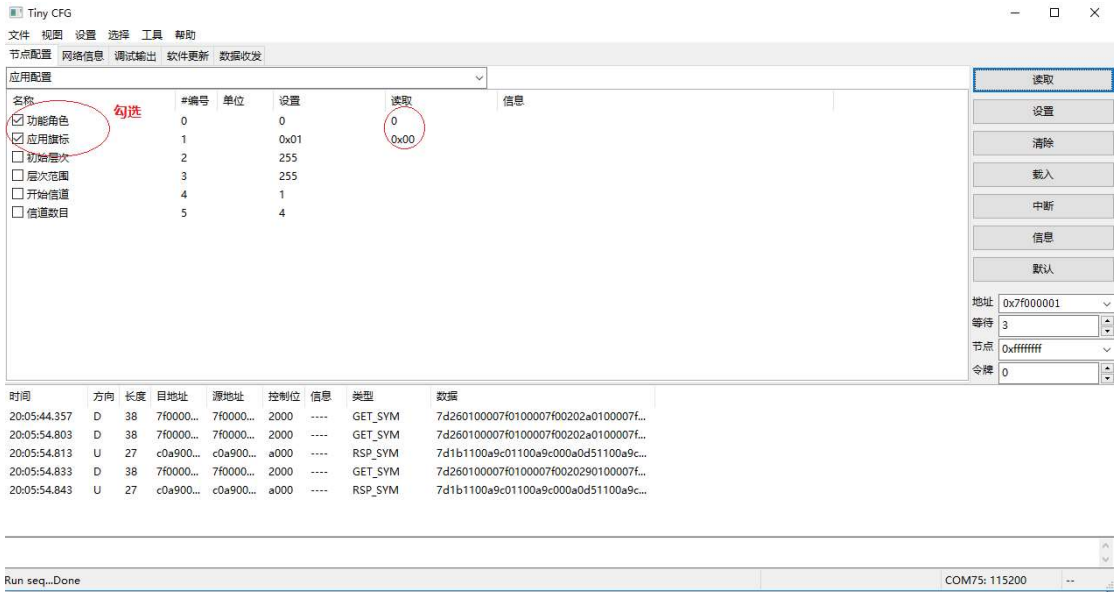
TinyCFG 是节点的配置软件，它通过串口和模块相连，它可以读取和设置模块的参数。
启动后首先选择串口



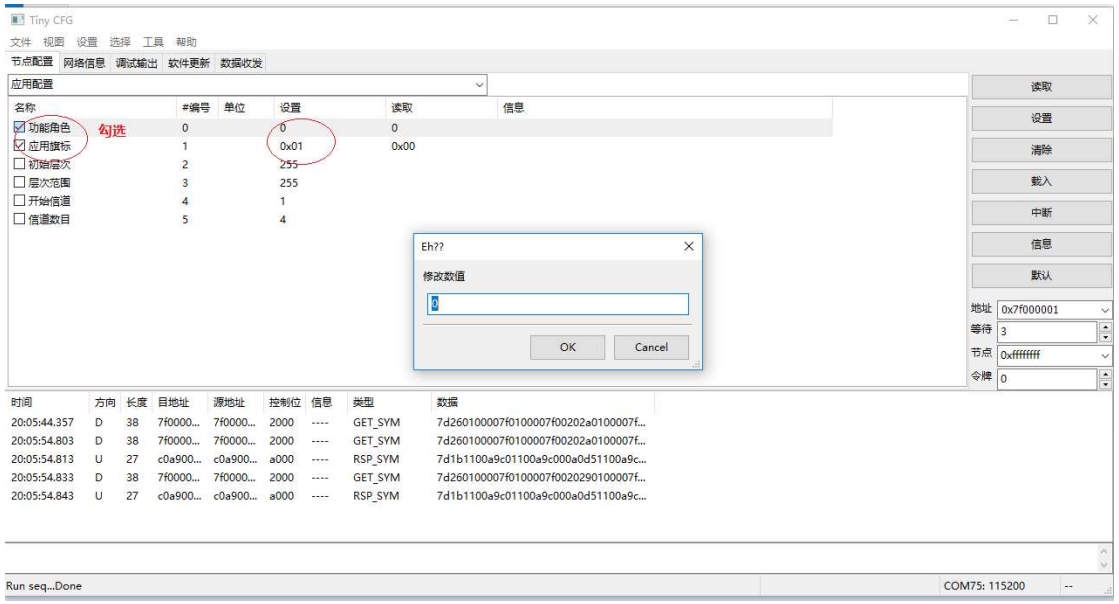
首先在左上角的下拉框中然后选取参数类别



读取参数-勾选需要读取的参数，点击读取按键

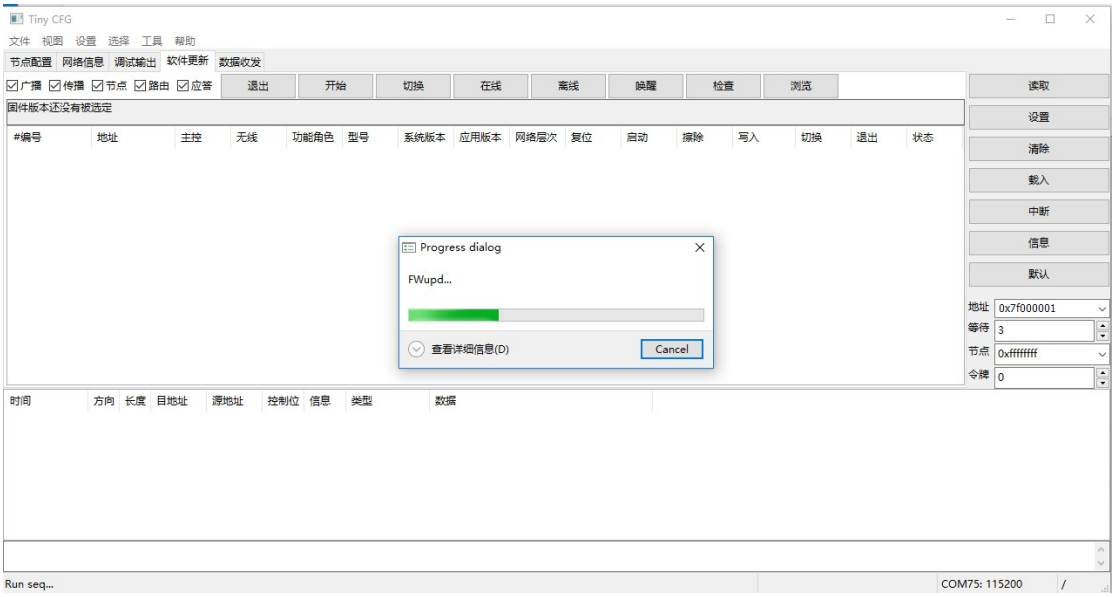


设置参数-勾选需要设置的参数，分别双击相应参数行，弹出修改对话框，修改完毕后，点击设置按钮



固件更新-这里只介绍通过串口更新固件，无线固件更新的操作参见**《TinyWSN 配置工具手册》**，固件可以是系统程序或者应用程序，点击工具菜单->更新，选择更新的文件

跳出下载进度条，等待模块重新复位，进度更新直到下载完成



参数说明

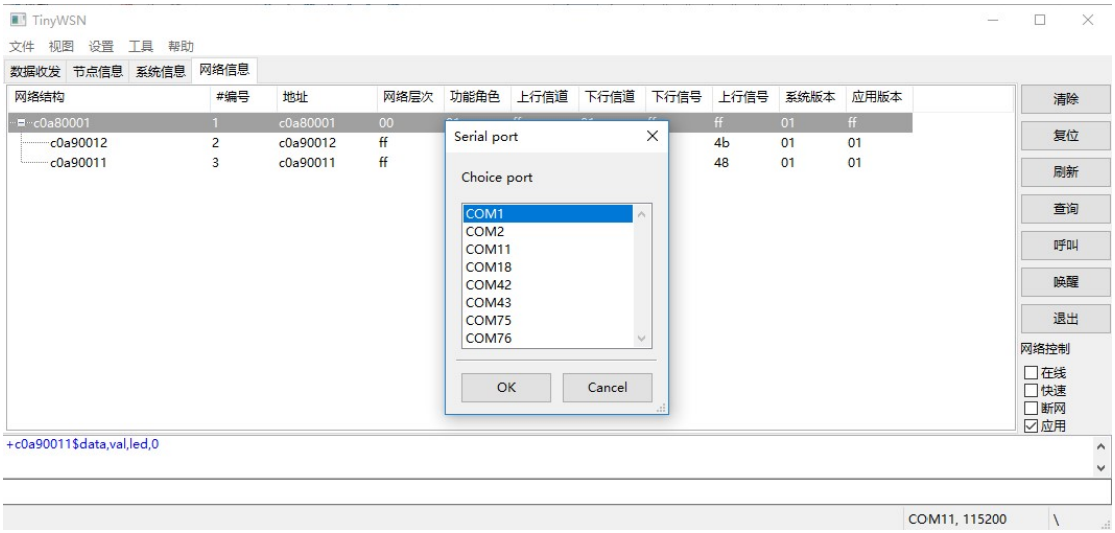
类别	名称	默认值	说明
系统配置	系统主控	3	
	无线系统	1	
	功能角色	3	
	硬件型号	1	
	无线地址	0xc0a80001	
应用配置	功能角色	0	0 终端节点 1 RTB 节点 2 RTT 节点 3 RTU 节点
	应用旗标	0x01	0x00 使用固定信道 0x01 动态扫描信道
	初始层次	255	主要用来决定路由节点在网络中的层次，0 表示根节点，节点模块只能接入层次比自己小的上级节点
	层次范围	255	用来节点控制可接入范围，只有符合（上级层次+层次范围 >= 自己层次），默认值表示可以接入任何层次
	开始信道	1	网络分配的频率资源
	信道数目	4	网络分配的频率资源
	扫描间隔	9	
	扫描门限	3	
	选定门限	6	
	扫描退出	64	
路由配置	路由旗标	0x00	
	下行信道	1	
	慢速时隙	0x323a	慢速时隙配置
	快速时隙	0x2228	快速时隙配置
	网络标识	0x3721	
	网络密码	0xbeef	
	广播信息	0x0401	
节点配置	节点旗标	0x01	0x00 离线节点 0x01 在线节点
	上行信道	0	
	网络密码	0xbeef	
	接入标识	0	
	网络标识	0x3721	

网络管理-TinyNMS

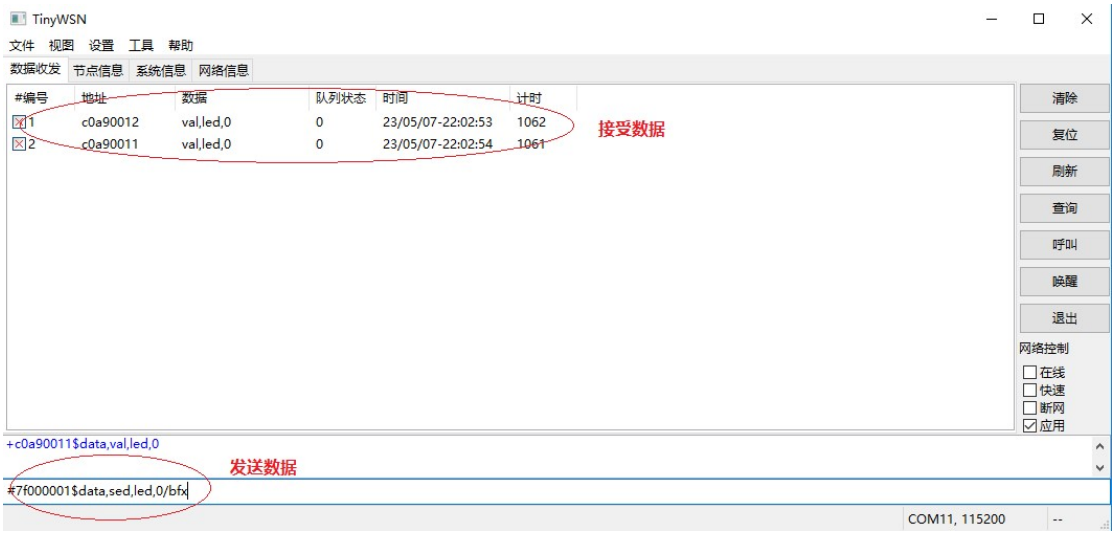
TinyNMS 是一个基于命令接口的简单的网络管理程序，它一般通过串口和网关或者根节点连接，可以读取网络信息，收发数据，唤醒节点等功能。
程序启动后，首先选择通信接口，目前它支持三种模式

- ✓ 串口通信
- ✓ TCP 客户端
- ✓ TCP 服务器

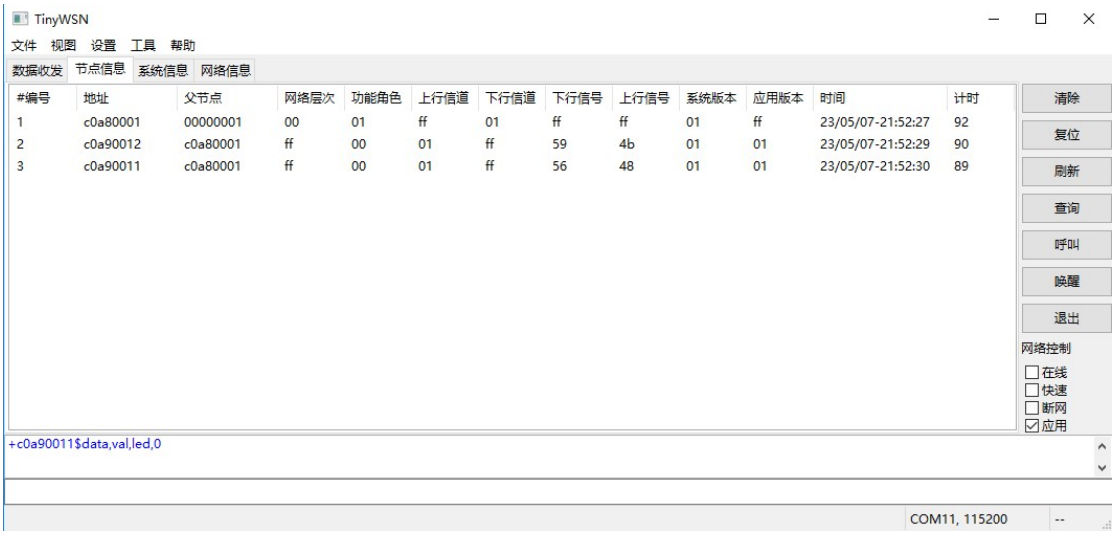
如果连接到开发板，只需要如下图打开串口：



数据收发-显示接收的数据以及接收时间，在下面的输入框中可以输入命令

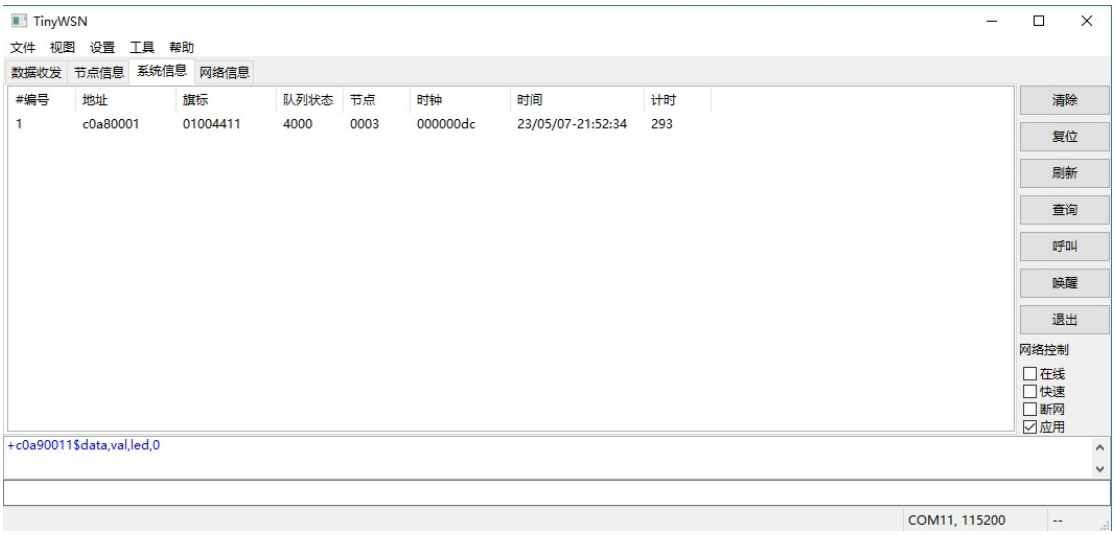


节点信息-为了收集节点信息，首先点击呼叫按钮，它会发出全网 ping 命令，所以在线节点会直接信息返回，从中可以看到各个节点的层次关系，分配的无线信道等等



网络信息-在节点信息收集完毕后，点击刷新按钮，它会重构网络的拓扑结构

系统信息-点击查询按钮，会返回系统信息，例如在线的节点数目，系统的功能设置，并且会更新右下角网络控制中的信息。

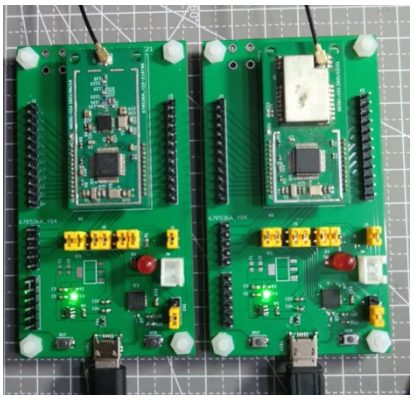


在右下角有网络控制选项，通过勾选可以控制网络功能

名称	说明
在线	当离线节点唤醒后，数据收发完毕后，仍然停留在在线状态，直至选项清除
快速	启动快速时隙配置，一般用作参数设置或者软件下载过程中。
断网	暂停网络，禁止节点接入
应用	启动内嵌的应用程序

实战教程-环境准备

硬件：开发板+节点模块数套+网关模块



软件：

名称	链接	提取码
TinyCFG	https://pan.baidu.com/s/1W9js3vFRQwyFjpzq5WfcYg	f8xp
TinyNMS	https://pan.baidu.com/s/1HcrRLq9yzkrJqSwcOb4McQ	8iqd
串口软件	https://pan.baidu.com/s/1hPyJzewDozY1TKnbxTODTA	9b13
USB 驱动	https://pan.baidu.com/s/1Axf7M-htnnFC3mS7F1MAzQ	uid1
最新固件	https://pan.baidu.com/s/102XFb7BztratVTEkWRJKKg	wx6s

下载和安装上面的软件，节点默认都是配置成终端节点，需要至少需要一个路由节点，充当根节点，其他层次的路由节点根据需要添加，可以使用 TinyCFG 进行配置，只修改下面三个参数：

路由节点	功能角色	网络层次	层次范围
根节点	0（RTB）	0	1
第一层	3（RTU）	1	1
第二层	3（RTU）	2	1
…			

后续教程为了更加直观方便，节点模块将使用的内嵌模式的一个演示程序，程序源码 https://gitee.com/tinywsn/fw-stm32l1-wbed-usr/apps/tinysh
可以自行编译，支持 gcc/iar/keil 环境，或者从最新固件目录下载编译好的文件，参见前面 TinyCFG 的串口烧录教程，演示程序完成以下几个功能

	在线状态	离线状态
LED 状态上报	按分配时隙的周期上报	默认 5 分钟自动唤醒，上报后重新进入离线状态
LED 控制命令	接受网络侧命令	接受网络侧通知
用户按键触发	翻转 LED 状态，按分配的周期上报	唤醒模块的睡眠，翻转 LED 状态，上报后重新进入离线状态

它定义了几条简单的应用层命令

命令	说明
set,led,stat	设置 LED 的状态
qry,led	查询 LED 的状态
val,led,stat	上报 LED 的状态

例如可以使用下面命令发送到节点 c0a80017，其中应用层的命令作为透明数据进行传输

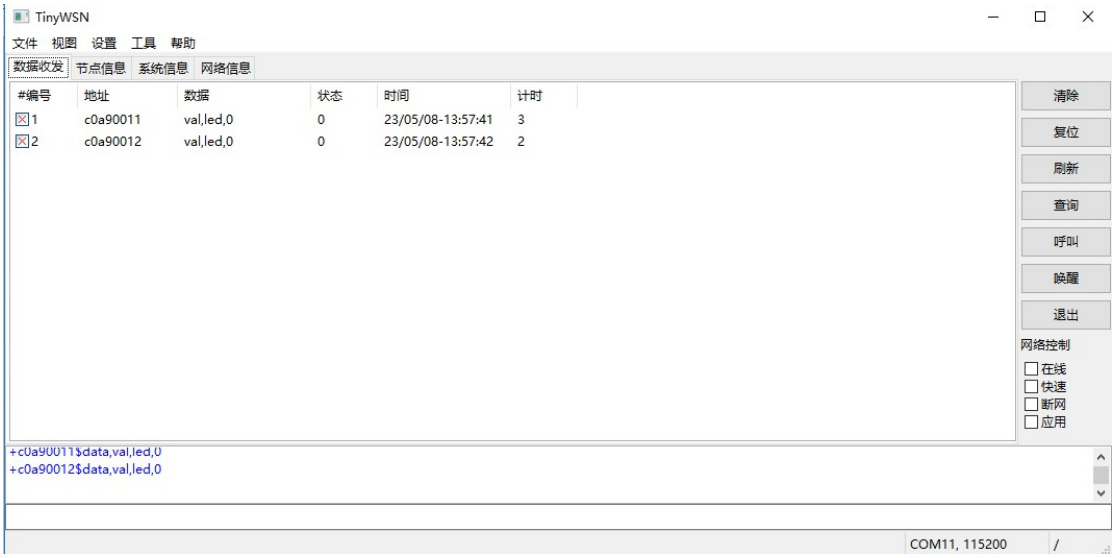
#c0a80017$data,set,led,1

当使用嵌入应用模式时，数据传送到嵌入应用，由嵌入应用进行解析，否则作为通知消息发送到串口，由外部应用进行解析，例如在交互命令模式时，会收到下面命令

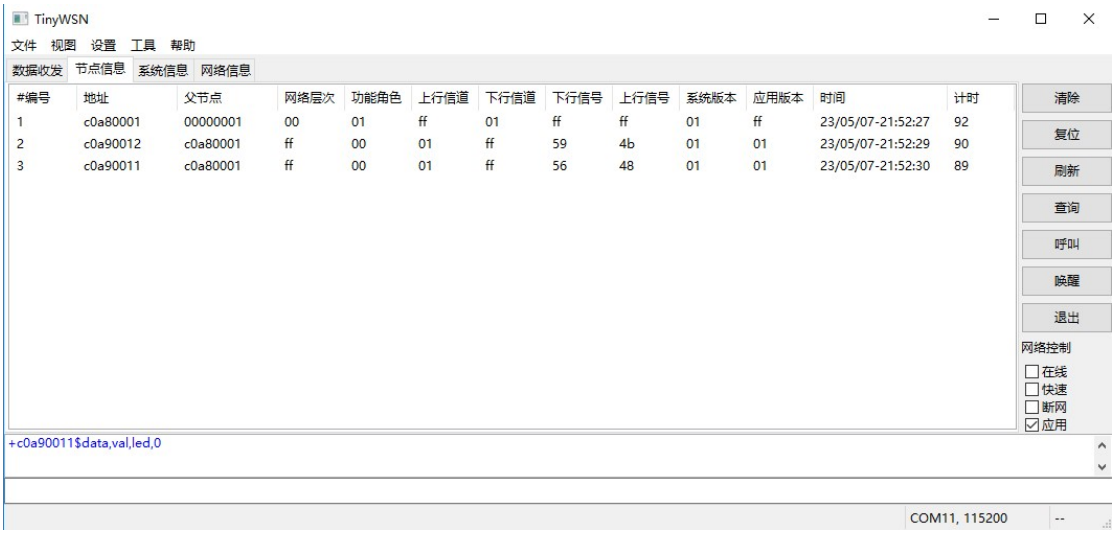
^c0a80001$data,set,led,1

实战教程-网络拓扑

利用 TinyNMS 连接到根节点串口上，各个模块自行上电，组网完成后，可以观察到各个节点上报的 LED 状态，在记录窗口中还上报的消息。



为了获取节点的其他信息，点击呼叫按钮，可以观察到各个节点上报的信息，这些信息包括无线信道，信号强度，以及网络层次等信息。



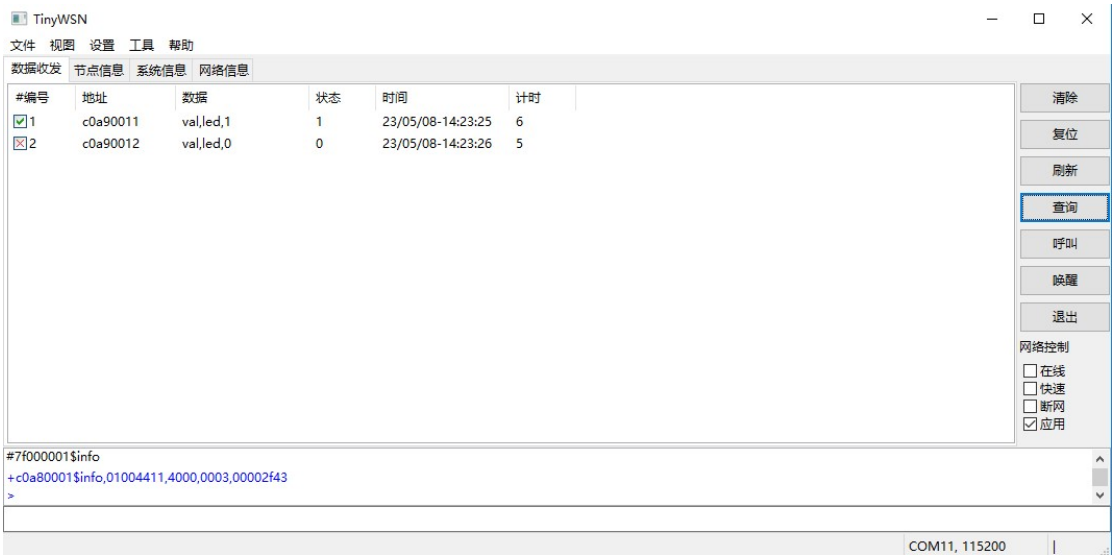
这时点击刷新按钮，根据接收到的节点信息，构成网络的树状的拓扑结构

实战教程-数据收发

在上面的教程中，我们看到了节点的上行数据，我们可以输入下面命令，点亮节点的 LED

#c0a90011$data,set,led,1

等待几秒后，可以观察节点的 LED 已经被点亮，同时看到新 LED 状态上报，也可以按开发板按键翻转 LED 状态，然后再观察到新上报的 LED 状态

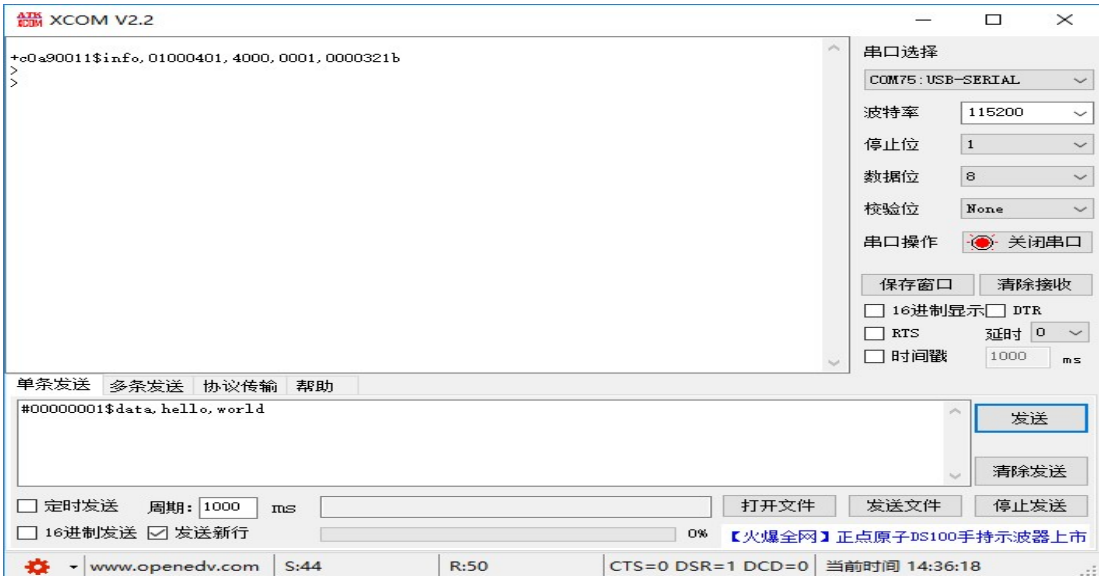


如果想尝试终端节点从串口发数据，首先清除勾选的网络控制-应用，这时所有节点应用程序被关闭，打开串口软件 XCom 连接到一个终端节点，输入下面命令

#00000001$data,hello,world

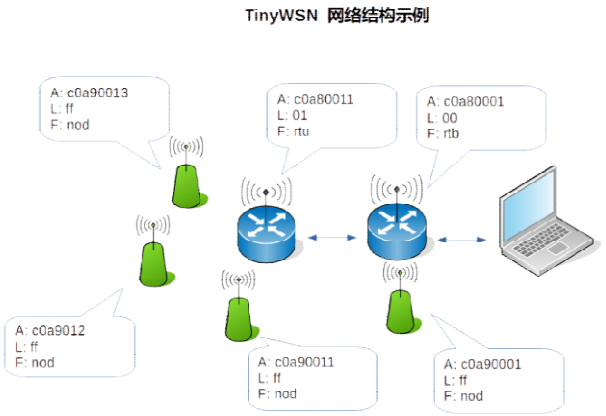
可在连接根节点的 TinyNMS 的记录窗口中看到这条上行消息

+c0a90011$data,hello,word



实战教程-小区广播

为了直观地看到各种寻址方式的差别，配置一一个简单的两级网络，如下图所示



在下面每条命令前，执行下面命令，关闭全网所有 LED

#7f000001$data,set,led,0/bfx

然后依次试验下面项目，观察结果

试验项目	命令	LED 点亮
小区广播	#c0a80001$data,set,led,1/bn	c0a90001
小区广播	#c0a80011$data,set,led,1/bn	c0a90011 c0a90012 c0a90013
子网广播	#c0a80001$data,set,led,1/bfn	c0a90001 c0a90011 c0a90012 c0a90013
子网广播	#c0a80001$data,set,led,1/bfx	c0a80001 c0a80011 c0a90001 c0a90011 c0a90012 c0a90013

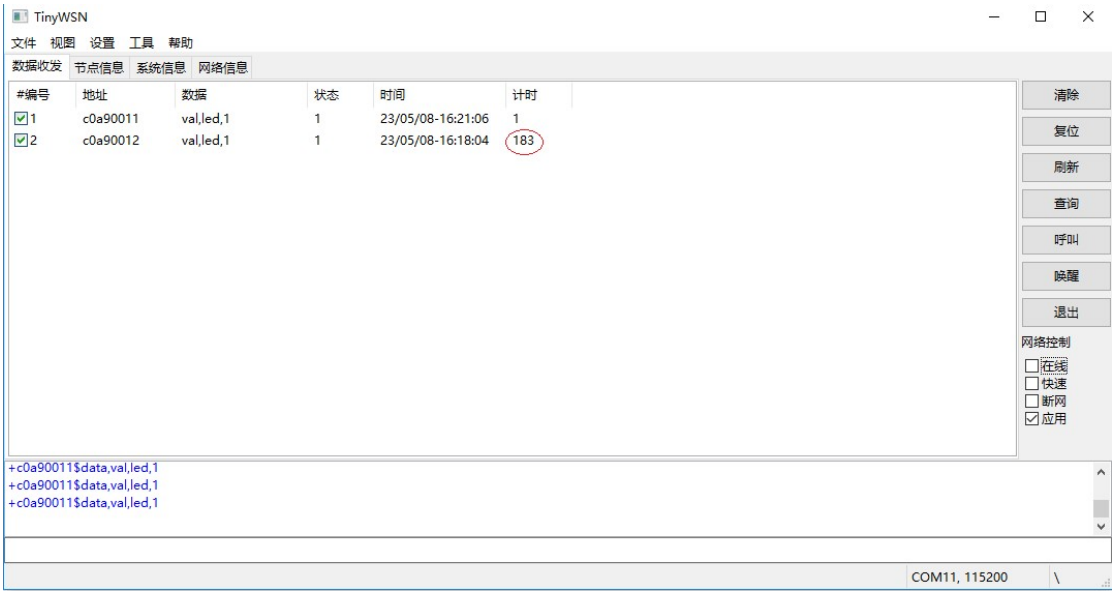
从试验中，可以很直观地体验到小区广播和子网广播的区别。

实战教程-无线唤醒

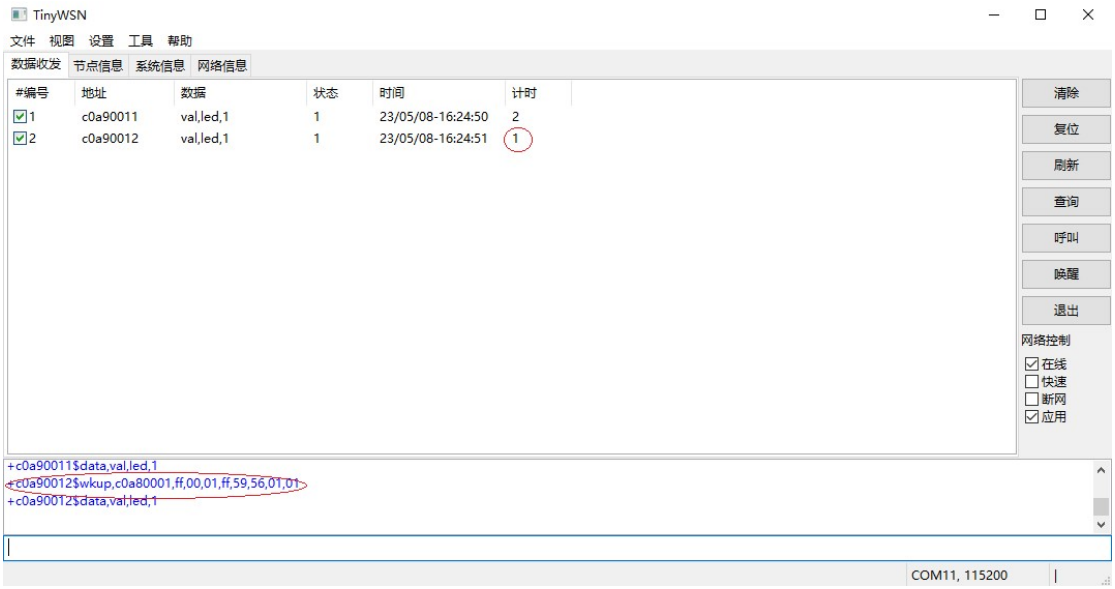
通过 TinyCFG 重新配置节点 c0a90012 为离线节点，只需修改下面参数

参数名称	原来	现在
节点旗标	0x01	0x00

可观察到节点从在线状态变成离线状态，不再上报数据了，计时不停增长



勾选网络控制->在线，然后点击唤醒按钮，然后可以看到唤醒应答，节点重新转入在线模式，开始周期上报数据，也可以下发控制命令了



如果清除勾选网络控制->在线，这是一个强制在线标记，如果清除，离线节点重新进入离线状态。

当点击唤醒按钮时，它发出下列命令，从下面的记录窗口可以看到，作用是全网唤醒所有的睡眠节点

#7f000001$wkup,ffffffff/bfx

也可以在命令框中手工输入命令，通过调整命令参数，可以根据精确的控制唤醒范围和对象，下面是一些唤醒示例

命令	说明
#7f000001$wkup,ffffffff/bfx	全网唤醒所有的睡眠节点
#c0a80011$wkup,ffffffff/bx	小区唤醒，路由节点 c0a80011 小区中的所有睡眠节点
#c0a80011$wkup,ffffffff/bfx	子网唤醒，路由节点 c0a80011 子网下的所有睡眠节点
#7f000001$wkup,c0a90011/bfx	全网范围唤醒的节点 c0a90011

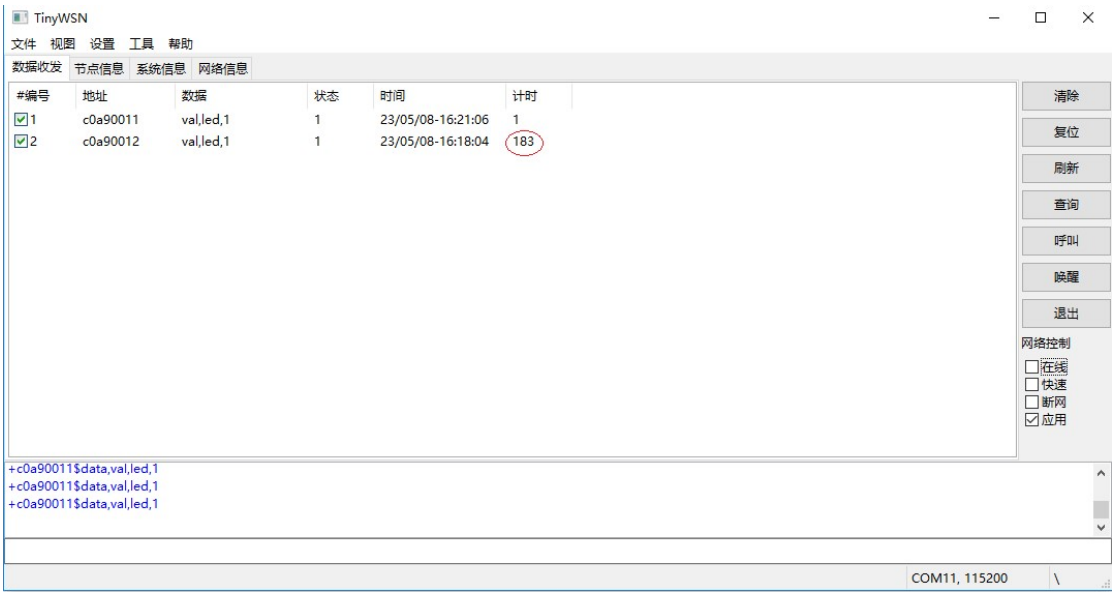
实战教程-异步上报

异步上报是指当节点配置为离线节点时，当它受到外部触发时，重新接入网络后，发送上行消息，对于不同的工作模式，有不同的外部触发源

模式	说明
交互命令 系统报文	当外部输入命令触发了一条上行消息，例如下面命令交互命令 #7f000001$data,val,led,1
嵌入模式	由嵌入的应用决定，例如在本文档中使用的例程 https://gitee.com/tinywsn/fw-stm32l1-wbed-usr/apps/tinysh 它定义按键作为外部告警触发，同时它也设置一个周期上报定时器。

,

下面教程以这个嵌入应用为例，如下图所示配置 c0a90012 为离线节点，可观察到节点进入离线状态，不再更新数据。

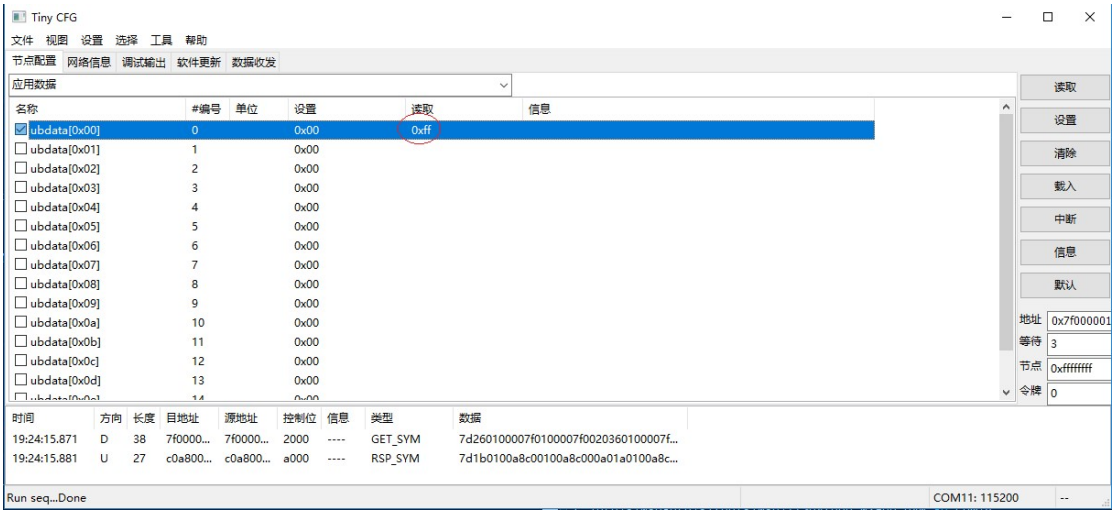


利用按键作为告警触发，看到 LED 状态翻转，然后节点重新接入网络，当 LED 的新状态上报后，节点会重新进入离线状态。

对于周期上报，是当上报定时器溢出时，节点被唤醒，重新接入网络后上报数据。上报的周期是由应用自己决定，本例程的周期存储在 Flash 的用户数据区中的第一个字节，可以通过配置工具 TinyCFG 进行查询和修改，如下图所示，当前读数是 0xff，基本单位是 30 秒，

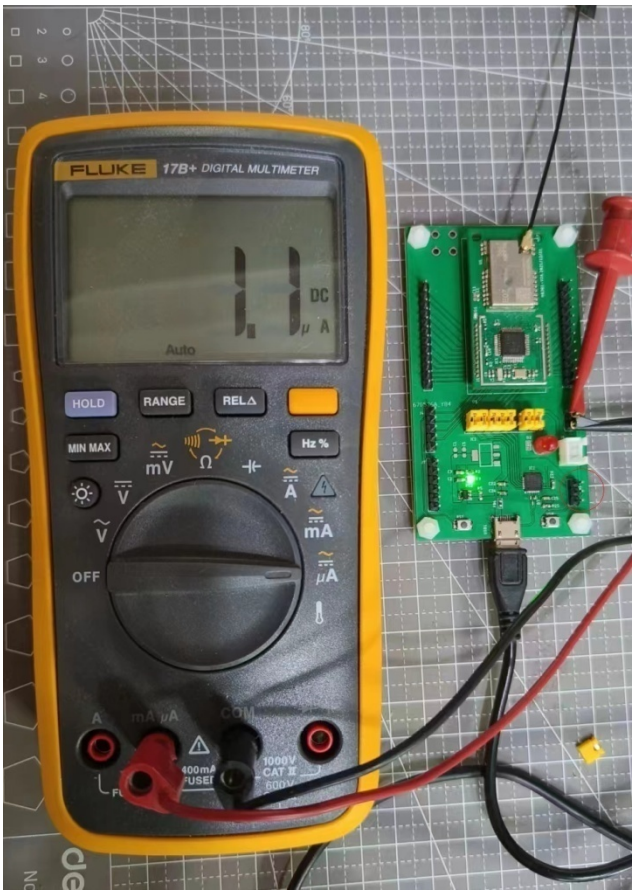
0xff * 30 = 7650 秒 = 127.5 分 = 2.125 小时

也就是说每 2.125 小时，嵌入应用就会被唤醒一次，它就上报自己的状态，具体过程可以参见它实现源码。



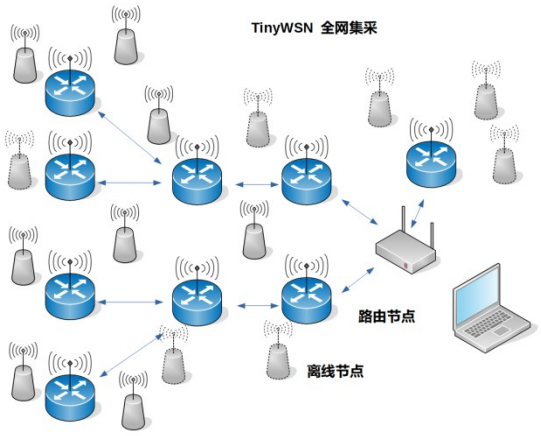
实战教程-功耗测量

测量离线节点功耗，如下图所示，把电流表串入 J1 中，同时需要把摘掉 CN1 的跳线帽，由于上电后，模块会首先尝试入网，会有几十 mA 的电流，所以需要先把表切换到 mA 档，防止电流大导致压降过多，等待模块进入监听状态，再切换到 uA 档读取平均的电流。
可以观察到大部分时间模块处于睡眠状态，维持一个很低功耗状态，中间会有电流的瞬时跳变，是由于模块周期唤醒进入监听状态，但持续时间非常短，占空比大概百分之几。



实战教程-全网集采

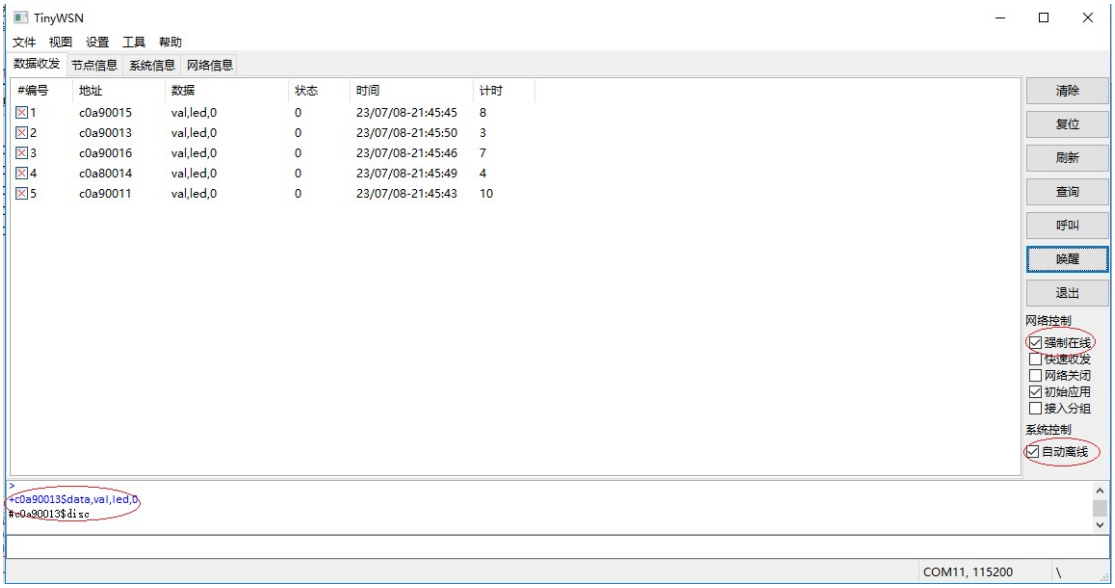
终端节点配置成离线节点（异步节点），它们按自己的周期（每 6 小时）接入上报数据，如下所示：



就这样服务器可以周期收到各个节点的状态信息，但是常常也会有需求同时集中获取所有节点状态的需求，这也称作全网集采，例如电表或水表在月末的时候，需要集采所有的读数。

TinyWSN 全网集采的的方法：唤醒所有离线节点，节点在接入时隙上竞争入网，节点设计了独特的接入控制的算法，可以使得节点高效地入网，节点上报数据后就恢复离线状态，让出无线资源，以便后续者继续接入，直至所有节点数据上报完成。所有的接入竞争只会发生在接入时隙上，已入网节点的通信不会受到干扰，所以即使终端节点数量多，也是能够顺利完成集采的。

如下图所示，收集数据后由网络侧释放的过程

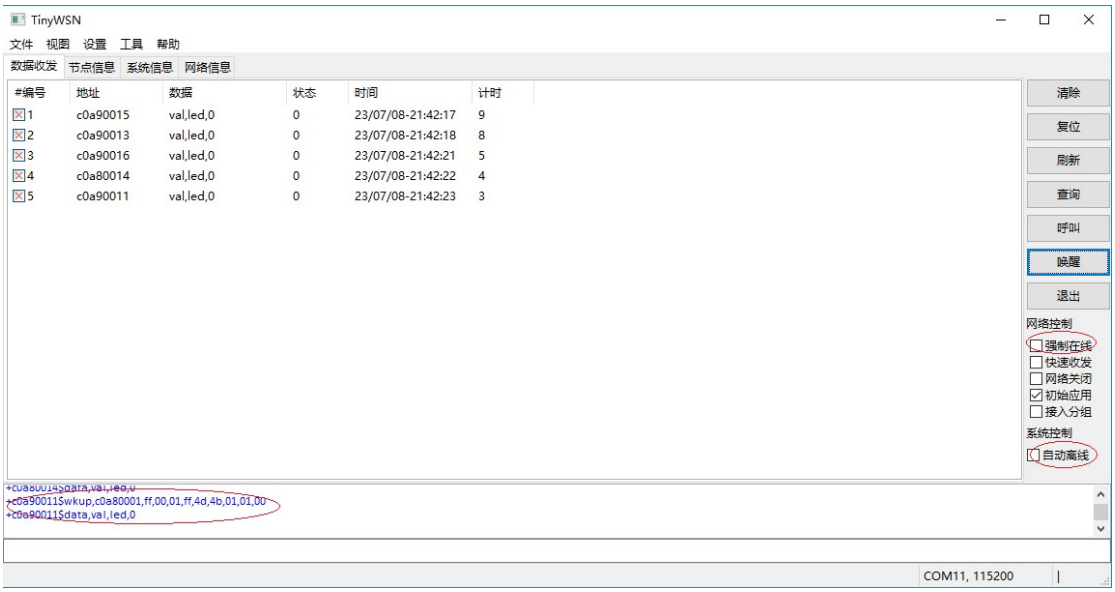


勾选强制在线和自动离线，然后点击唤醒，然后就可以观察到各个节点激活和数据上报，以及恢复离线状态

从命令记录窗口中导出命令和应答，就可以清晰地看到整个操作过程

命令	说明
#7f000001$wkup,ffffffff/bfx	全网唤醒命令
+c0a90011$wkup,c0a80001,ff,00,01,ff,48,50,01,01,00	节点 c0a90011 唤醒响应
+c0a90011$data,val,led,0	节点 c0a90011 状态上报
#c0a90011$disc	断开节点 c0a90011 连接
…	
+c0a90016$wkup,c0a80001,ff,00,01,ff,4d,52,01,01,00	节点 c0a90016 唤醒响应
+c0a90016$data,val,led,0	节点 c0a90016 状态上报
#c0a90016$disc	断开节点 c0a90016 连接

当然也可以上报数据后，由节点自己释放链路



首先强制在线和自动离线都不勾选，点击唤醒，同样也可以观察到节点激活和数据上报，以及释放的过程，导出它的命令和响应

命令	说明
#7f000001$wkup,ffffffff/bfx	全网唤醒命令
+c0a90011$wkup,c0a80001,ff,00,01,ff,48,50,01,01,00	节点 c0a90011 唤醒响应
+c0a90011$data,val,led,0	节点 c0a90011 状态上报
…	
+c0a90016$wkup,c0a80001,ff,00,01,ff,4d,52,01,01,00	节点 c0a90016 唤醒响应
+c0a90016$data,val,led,0	节点 c0a90016 状态上报

由于没有设置强制在线，唤醒的节点在发送完所有的数据后，自动进入离线状态。下表比较两者不同释放方式的特点：

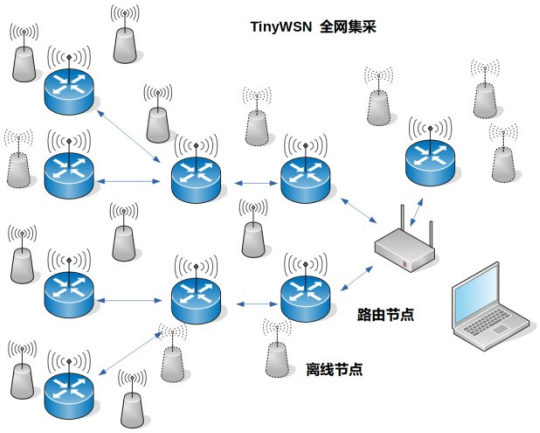
释放方式	说明
节点释放	由于节点上报完自己释放，无需等待网络确认，资源利用率高，能更快地完成采集
网络释放	数据上报后由网络侧释放，使得采集过程更加可靠，而且还可以在释放前，插入其他配置命令，更加灵活

在一些特殊情况下，有些节点数据没有上报，希望再次唤醒没有上报的节点，而已经上报的节点就不要唤醒，避免重新加入接入竞争，可以利用唤醒命令的 token 参数达到这个效果，唤醒命令的格式如下：#addr$wkup,node[,token]，其中 token 是可选项，默认是 0

当 token 是非零值，例如发送命令#7f000001$wkup,ffffffff,5/bfx，节点唤醒上报后重新进入离线状态，它会记录最近唤醒的 token，如果重发这条命令，只要是相同的 token，它就不会响应唤醒，依旧保持离线状态，只有上次唤醒异常的节点会响应。这样就能够有效地解决漏网之鱼的问题。

实战教程-全网集控

全网集控是指对全网所有的节点进行集中控制，例如控制路灯，阀门等等，如下图所示，



所有的终端节点均为离线节点，周期接入上报状态，或者外部告警触发上报，也可以由网络侧进行无线唤醒后进行控制和查询。

TinyWSN 全网集控的方法：

- ✓ 第一种方法是使用通知消息，无需唤醒节点入网，通过广播方式直接发送到离线的睡眠节点，最后再做一次全网集采，确认各个节点的状态
- ✓ 第二种方法就是类似全网集采的做法，唤醒离线节点重新入网，逐一进行配置，确认状态后再释放节点，直至所有节点完成设置

通知消息的命令格式如下，和节点唤醒命令非常类似，但是它可以携带用户数据，可以透明传输应用层的命令

#addr$ntfy,node,payload

其中 addr 是路由节点的地址，node 是目标节点的地址，它们可以是特定模块的地址，也可以是广播地址，这个命令的作用就是让路由节点发通知消息到离线节点。而之前学习的在线节点发送命令就简单一点，只需给出目标节点的地址，路由节点能够自动完成路由。

#addr$data,payload

下面通过一些示例来演示它的用法，

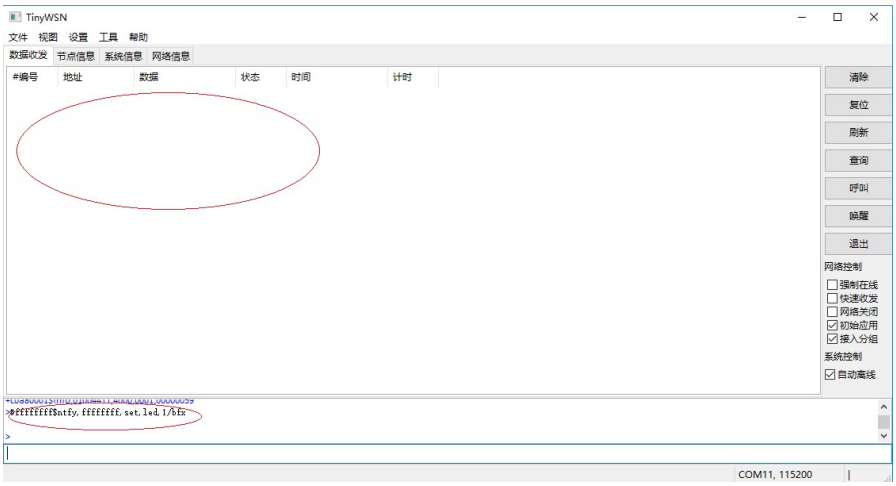
命令	说明
#ffffffff$ntfy,ffffffff,set,led,1/bfx	通知所有的离线节点都点亮 LED，以看到路由节点和目标节点的地址都是广播地址，意味着所有的路由节点都广播这条信息，所有的离线节点都会响应这条命令
#ffffffff$ntfy,c0a80013,set,led,1/bfx	所有的路由节点都广播这条信息，只有节点 0xc0a80013 会响应这条命令
#c0a80005$ntfy,ffffffff,set,led,1	只有路由节点 0xc0a80005 会广播这条信息，无线覆盖范围内的离线节点都会响应
#c0a80005$ntfy,ffffffff,set,led,1/bfx	路由节点 0xc0a80005 以及所有它的子网内的路由节点都会广播这条信息，无线覆盖范围内的离线节点都会响

	应

当使用下面全网点亮的命令后，

#ffffffff$ntfy,ffffffff,set,led,1/bfx

可以观察到所有同层节点几乎同步点亮了 LED，而且响应非常迅速，主要原因是离线节点无需唤醒入网，处于睡眠状态就可以响应这条命令，当然不同网络层次需要转发消息，层次不同节点 LED 点亮时间会有延时。如下图所示，可以看到当节点响应命令后，并没有接入的过程，而是自行恢复睡眠状态。



如果有多条通知消息需要发送，可以借助路由节点的发送应答来实现，使用命令选项‘p’，当各个路由节点广播完消息后，它会回送一个命令应答，如下所示

#ffffffff$ntfy,ffffffff,set,led,1/bfxp

>

+c0a80001$resp,00,25,00,00

+c0a80012$resp,00,25,00,00

+c0a80013$resp,00,25,00,00

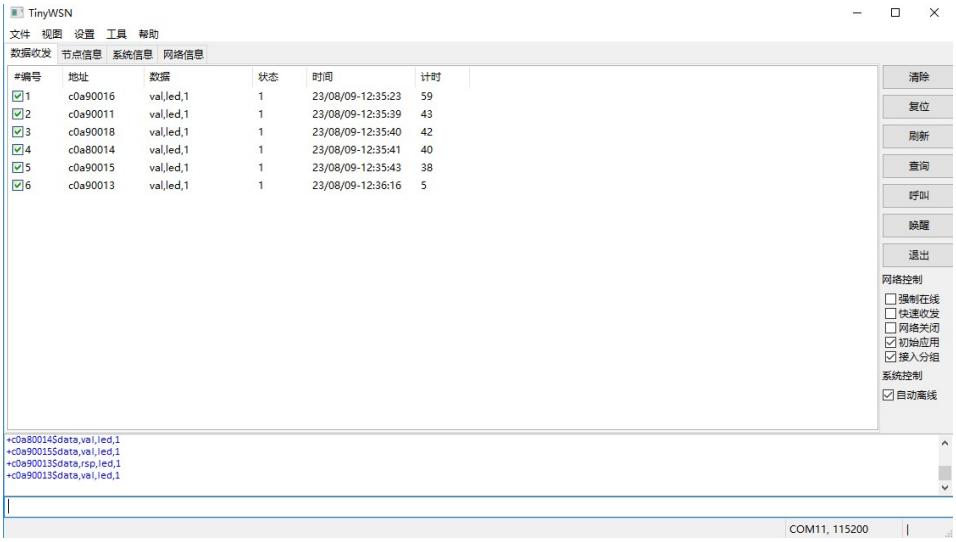
+c0a80014$resp,00,25,00,00

+c0a80023$resp,00,25,00,00

当收集到所有路由节点的响应后，表示目前通知消息已经全网发送完成，就可以发送下一条通知消息了。当所有通知消息发送完成后，最后做一次全网集采，获取各个节点的状态，确认控制的结果。全网集采命令可以使用节点唤醒命令，或者仍然也可以使用通知消息的途径

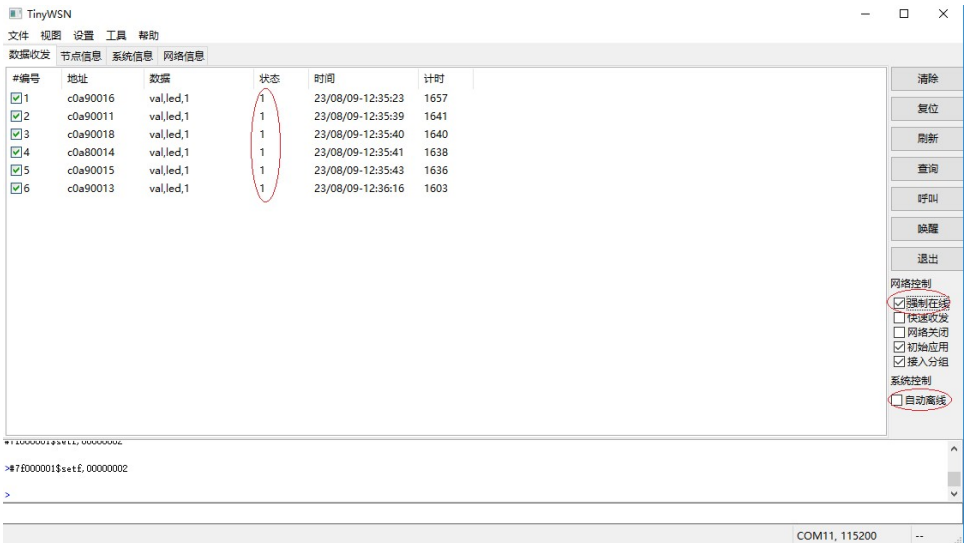
#ffffffff$ntfy,ffffffff,qry,led/bfx

当嵌入应用收到查询命令时，就会返回led的目前状态，由于缓冲区有待发消息，节点自动从离线状态切换到网络接入过程，如下图所示:



当上行消息发送后，节点自动恢复成离线状态，这样就完成了一次全网集采过程，获取了各个节点的状态。

另外一种方法是和全网集采类似的原理，唤醒各个节点逐一设置，确认状态后释放节点， 如下所示， 勾选强制在线，清除自动离线。



命令过程如下表所示， 为每个接入网络的节点逐一进行设置

序号	命令或响应	方向	说明
1	#ffffffff$wkup,ffffffff/bfx	命令	唤醒网络所有的离线节点
2	#c0a80011$data,set,led,1	命令	从入网的节点挑选其中一个 c0a80011，设置它的 LED 状态
3	^c0a80011$data,val,led,1	响应	收到节点 c0a80011 的状态报告，确认设置成功
4	#c0a80011$disc	命令	释放节点 c0a80011，空出无线资源以便新节点接入
5	…	…	为每个入网节点重复 2-4 过程

最后总结一下这两种方法的的特点：

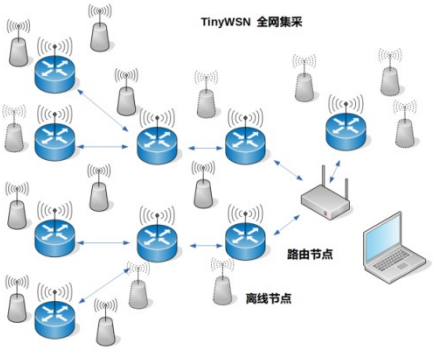
方法	说明
通知消息	节点响应速度快，而且是同步响应，缺点是需要做一次全网采集来收集设置结果
唤醒入网	逐一设置和确认，可靠性高，但是设置过程耗时长，节点逐一响应设置的

全网广播通知消息在发送前，也需要把离线的睡眠节点无线唤醒，以便它能接受数据，为了可靠性消息会多次发送，重复发送的次数是可以配置的，而且消息还可能从其他临近路由节点收到，这样离线节点会收到重复的通知消息，如果需要避免重复接受，应用层的命令中可以加入消息序号进行过滤。

实战教程-全网静默

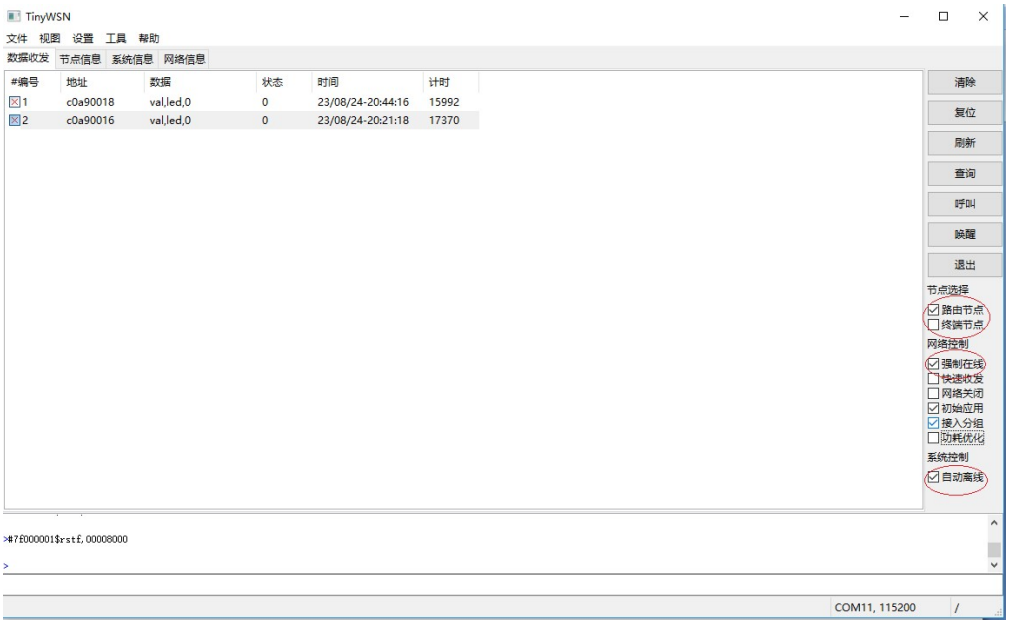
在一些应用场景中，除了对终端节点的功耗，对路由节点的功耗也有严格的要求，这些应用场景中通信的流量比较低，而且数据通常是周期采集的，比如几个小时一次，但是偶尔也会有突发状况，比如终端节点受到外部触发需要主动上报告警消息，全网静默就非常适应这个场景，终端节点和路由节点都处于 uA 级别的功耗（由睡眠周期决定，通常 2~3uA 的功耗），同时它支持自顶向下（top-down）和自下向上(bottom-up)发起的的通信方式。

操作流程如下图所示，终端节点和路由节点都配置成离线节点，终端节点取消周期上报，由网管软件 TinyNMS 控制进行周期的全网集采，用开发板的按键作为终端节点外部触发源，触发上行数据传输。



上电后所有节点都进入离线状态（异步状态），这种状态其实是一种不连续接收状态，也称 DRX，它会周期打开无线接收窗口，扫描用于唤醒的无线前导序列， 这也就是全网静默状态，所有节点处于等待唤醒的低功耗状态，不再有无线信号的发射。

- ✓ 自顶向下（top-down）的模式是由网络侧主动发起的，首先唤醒所有路由节点，构建整个传输网，如下图所示



节点选择中只勾选路由节点和强制在线，点击唤醒按钮，这样只有路由节点会被唤醒，首先时根节点被唤醒，从记录窗口中可以观察到路由节点的唤醒响应

```
+c0a90011$wkup,c0a80001,ff,00,01,ff,48,50,01,01,00
+c0a90016$wkup,c0a80001,ff,00,01,ff,48,50,01,01,00
...
```

当有新的路由节点加入后，可以继续点击唤醒按钮，这样使得更远的路由节点得以唤醒，直至所有的路由节点都被唤醒，可以使用网络信息页面观察目前的网络拓扑结构，

当由路由节点构成的传输网构建好了，就可以进行常规的操作了，可以参考前面的教程

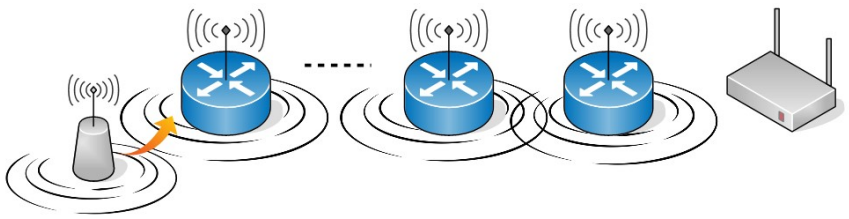
《实战教程-全网集采》

《实战教程-全网集控》

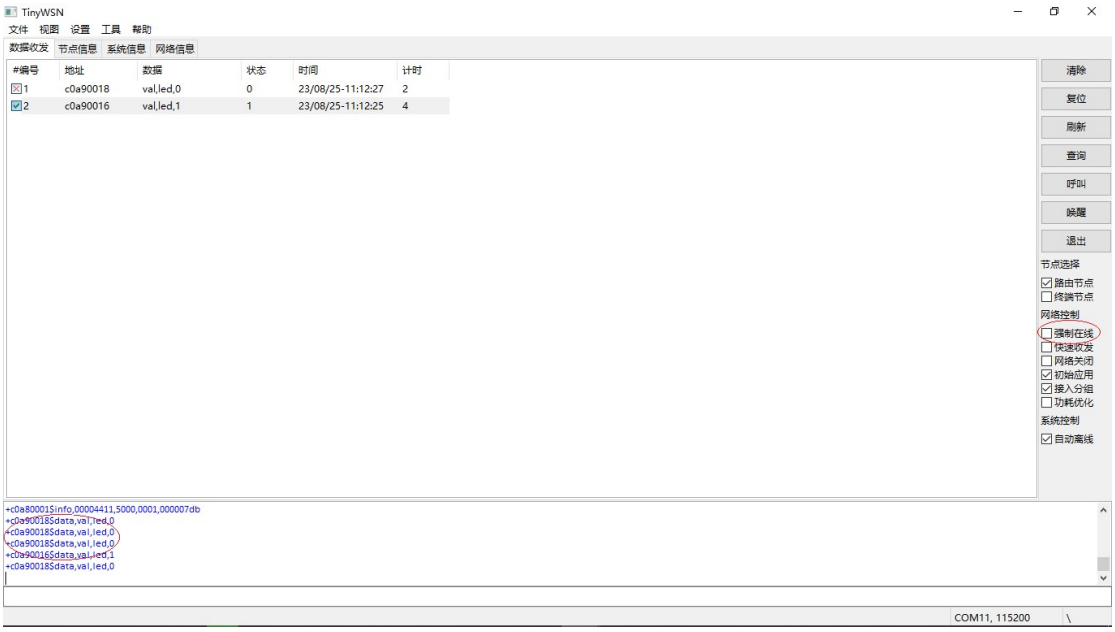
对网络进行集采和控制了，当所有的操作完成后，清除强制在线的勾选，恢复网管软件 TinWSN 的默认配置，这时所有的节点就逐级变成离线节点，最后又进入到全网静默状态了。

✓ 自底向上（bottom-up）的模式是终端节点主动发起的， 终端节点被外部触发源唤醒后，首先尝试扫描周围的路由节点，由于所有的路由节点都处于离线状态，当扫描失败后它会发起唤醒周围路由节点的操作，被唤醒的路由节点继续唤醒周围的路由节点，直至根节点被唤醒，如下图所示：

自低向上（bottom-up) 唤醒



网管软件 TinyNMS 的配置如图所示，清除强制在线状态，避免被唤醒的节点长期保持在线，这也是全网静默的默认配置

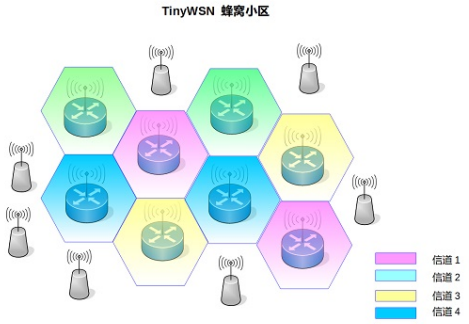


按键触发改变终端节点的指示灯状态，终端节点被唤醒上报状态，可以观察到各级路由节点被逐级唤醒，终端节点状态成功上报，由于清除了强制在线，终端节点在上报完所有消息后进入离线状态，各级路由节点也逐级进入离线状态，最终又恢复成全网静默状态。

全网静默模式有许多优点，它使得所有节点都保持很低的功耗，而且静默时完全没有无线信号发射，使得无线信道保持干净，提高无线信道的利用率，并且无线数据也不容易被侦测到，它的主要缺点是每次通信前都需要重建无线网络，带来了额外的延迟。

实战教程-自动组网

TinyWSN 是一个同步的层次网络架构，网关作为网络的根节点，每个路由节点位于这个网络层次中，它为一个无线小区提供接入服务，整个网络呈现一个蜂窝状的结构，如下图所示



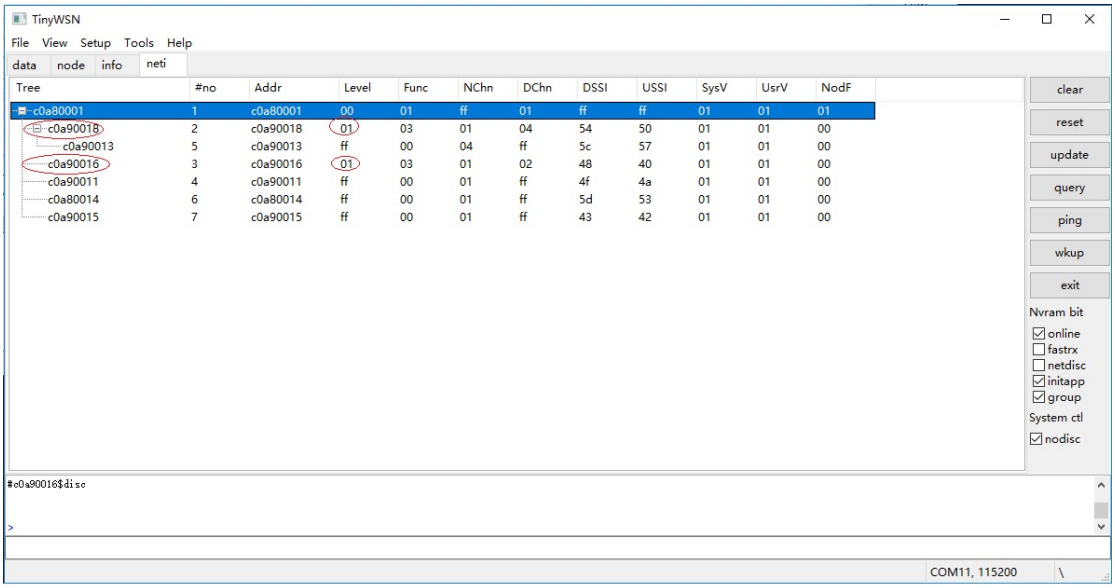
通常会根据网络规划的结果，配置每个路由节点的层次关系，形成整个骨干传输网络，终端节点就近接入。

当然路由节点也可以支持自动组网，可以简化配置过程， 网关作为网络的根节点层次为 0，其他各个路由节点使用默认的配置，上电后自动扫描周围的邻接节点，以根节点为启点，自动分配网络层次。

除了网关节点，所以其他路由节点的的网络层次和网络层次均使用默认值 255，如下表所示

名称	数值	说明
初始层次	255	主要用来决定路由节点在网络中的层次，0 表示根节点，节点模块只能接入层次比自己小的上级节点
层次范围	255	用来节点控制可接入范围，只有符合（上级层次+层次范围 >= 自己层次），默认值表示可以接入任何层次

把路由节点安装在规划的位置上电，从网络管理软件 TinyNMS 从可以观察到各个节点上报的信息，然后再生成网络拓扑图，从中可以看到各个路由节点的自动分配网络层次，如下图所示：

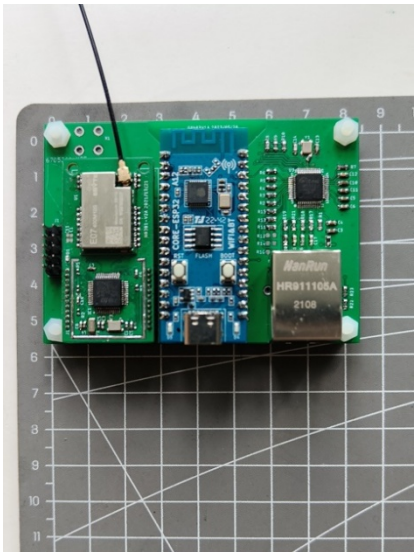


我可以看到路由节点 c0a90018 和 c0a90016 自动接入到网关，而且根据它们目前的所处的网络层次，自动重新分配新的层次号 1，而且 c0a90018 下面还有连有一个终端节点 c0a90013.

当然也可以混合组网，骨干路由节点根据网络规划进行配置，在现场发现的无线盲点，或者网络边缘扩展，就可以利用自动组网能力，免配置补充新的路由节点，从而优化网络的覆盖。

实战教程-WIFI 透传

在本教程中需要使用到一块透传型网关 G0B03，上面安插一块无线节点模块，作为网络的根节点,网络管理程序 TinyNMS 运行在主机上，启动 TCP 服务器



下面的命令序列主要完成 WIFI 初始化，连接 TCP 服务器， 初始化 WSN 接口，最后和 WIFI 建立一个管道，这样无线节点的命令就透明传输到 TCP 服务器上了。

命令	说明
wsn –c init –o usrpkt	初始化节点模块接口，使用命令模式
wifi –c init –o drv	初始化 wifi 驱动，它会根据配置项中的接入点信息，进行扫描寻找接入点，然后使用密码接入
wifi –c conn –o cli	启动 TCP 客户端，连接在配置项定义的主机
ipc –c conn –s wsn –d wifi	在节点模块和 WIFI 之间建立通信管道，这样节点命令接口就被透明传输到远端的 TCP 服务器

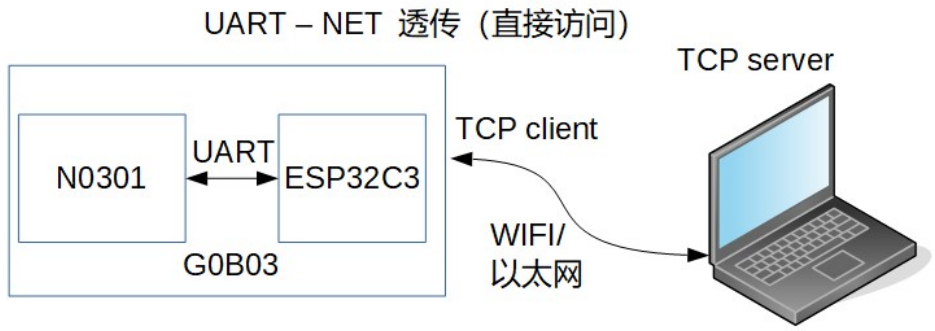
建立连接后，在 TinyNMS 上就可以收到各个节点上报的信息，也可以直接发命令控制节点模块了，就如同在本地使用 USB 接口的开发板一样了。

如果定义下面自动运行配置项，上电就自动运行 wifi 透传，其实它运行的就是上面的命令序列。

nvm –c set –k autorun –v wsn2wifi

实战教程-ETH 透传

ETH 透传和 WIFI 透传是非常相似的，只是传输的物理接口不同，它工作原理如图下所示



下面的命令序列主要完成 ETH 初始化，连接 TCP 服务器， 初始化 WSN 接口，最后和 ETH 建立一个管道，这样无线节点的命令就透明传输到 TCP 服务器上了。

命令	说明
wsn –c init –o usrpkt	初始化节点模块接口，使用命令模式
eth –c init –o drv	初始化 eth 驱动，通过 dhcp 获取 ip 地址
eth –c conn –o cli	启动 TCP 客户端，连接在配置项定义的主机
ipc –c conn –s wsn –d eth	在节点模块和 eth 之间建立通信管道，这样节点命令接口就被透明传输到远端的 TCP 服务器

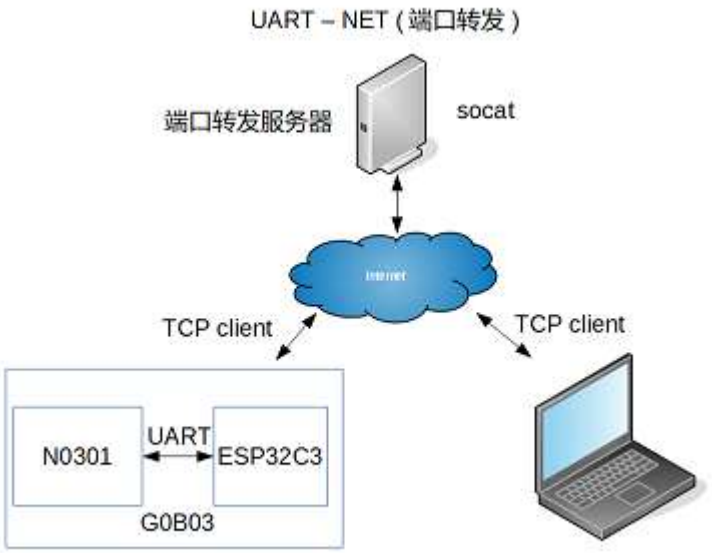
建立连接后，在 TinyNMS 上就可以收到各个节点上报的信息，也可以直接发命令控制节点模块了，就如同在本地使用 USB 接口的开发板一样了。

如果定义了下面自动运行配置，上电就自动运行 eth 透传，其实它运行的就是上面的命令序列。

nvm –c set –k autorun –v wsn2eth

实战教程-端口转发

在上面网络（WIFI 或 ETH）透传教程中，网关和主机都处于同一网段，或者可以通过路由直接访问，否则就要通过如下所示的端口转发机制，需要有一个公网主机作为端口转发主机



网关和主机都作为 TCP 客户端，访问主机的配对的转发端口，数据在端口之间透明转发，公网主机运行一下端口转发命令，在端口 9903 和 9904 之间转发

socat tcp-listen:9903 tcp-listen:9904

桌面主机运行 TinyNMS ，它作为 TCP 客户端连接至公网主机的 9903 端口， 而网关通过 wifi 连接至公网主机的 9904 端口

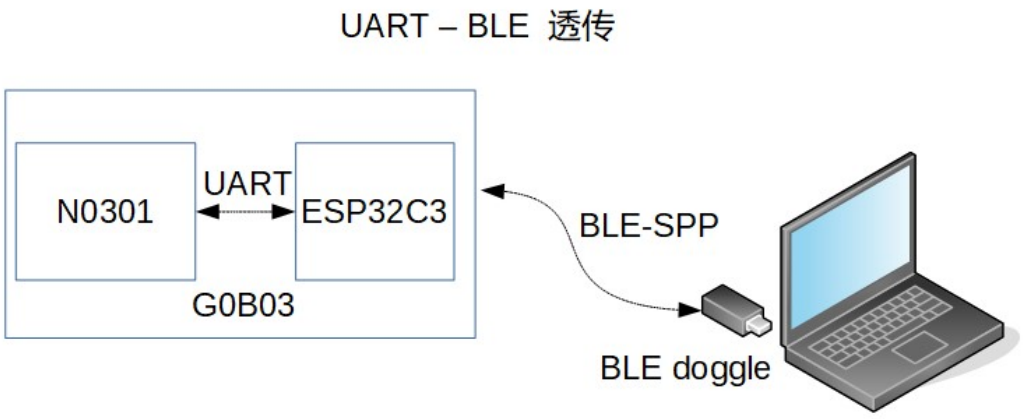
命令	说明
wsn –c init –o usrpkt	初始化节点模块接口，使用命令模式
wifi –c init –o drv	初始化 wifi 驱动，它会根据配置项中的接入点信息，进行扫描寻找接入点，然后使用密码接入
wifi –c conn –o cli	启动 TCP 客户端，连接在配置项定义的主机，主机为公网主机，端口为 9904
ipc –c conn –s wsn –d wifi	在节点模块和 WIFI 之间建立通信管道，这样节点命令接口就被透明传输到远端的 TCP 服务器

通过这样配置 TinyNMS 就和网关连通了，就可以收到各个节点上报的信息，也可以直接发命令控制节点模块了，就如同在本地使用 USB 接口的开发板一样了。如果定义下面自动运行配置项，上电就自动连接到公网主机，实现 wifi 透传功能：

nvm –c set –k autorun –v wsn2wifi

实战教程-BLE 透传

除了一块透传型网关 G0B03，它还需要一个 ESP32C3 的开发板，插入到主机上作为一个 BLE dongle，它提供一个 USB 串口，它和网关 G0B03 建立 BLE 连接，它工作原理如图下所示



BLE doggle 它的命令序列如下，它启动作为一个 BLE 服务器，并且和 USB 串口之间透传

命令	说明
tsh –c init –v text	初始化命令接口(USB 串口）为文本模式
ble –c init –o srv	初始化 ble 驱动，作为服务器，开始广播
ipc –c conn –s tsh –d ble	在网关的命令接口和 ble 之间建立通信管道,这样 BLE 的数据就透明传输到 USB 的串口中

网关模块的命令序列如下，它作为一个 BLE 客户端，连接 BLE 服务器，并且和 WSN 透传

命令	说明
wsn –c init –v usrpkt	初始化命令接口(USB 串口）为文本模式
ble –c init –o cli	初始化 ble 驱动，作为客户端
ble –c scan –o cli	启动客户端扫描，寻找服务器
ble –c conn –o cli	连接到 BLE 服务器
ipc –c conn –s tsh –d ble	在节点模块和 ble 之间建立通信管道，这样节点命令接口就被透明传输到远端 BLE

建立连接后，在 TinyNMS 上就可以收到各个节点上报的信息，也可以直接发命令控制节点模块了，就如同在本地使用 USB 接口的开发板一样了。

在上面例程中，BLE 客户端只会连接已绑定的服务器，如果需要可以运行下面命令，完成重新绑定

命令	说明
ble –c rescan –o cli	启动客户端扫描，重新寻找服务器，

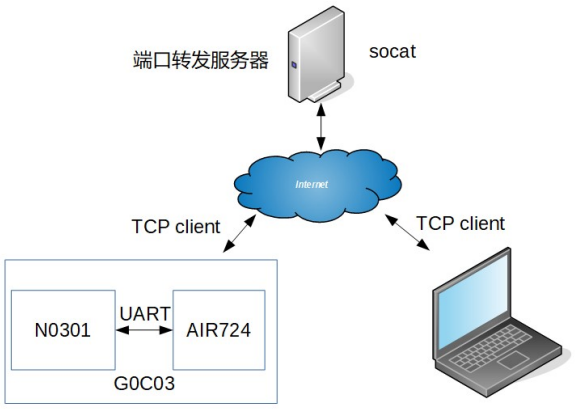
ble –c conn –o cli	连接到 BLE 服务器
ble –c bind–o cli	绑定新的服务器

当完成绑定后，下次就可以运行自动透传功能了。

当定义了下面的上电自动运行配置项，它们就会自动执行透传功能
BLE doggle 的上电自动运行配置：nvm –c set –k autorun –v tsh2ble
网关模块的上电自动运行配置项：nvm –c set –k autorun –v wsn2ble

实战教程-LTE 透传

LTE 透传使用的 G0C03 网关，它使用 Air724 作为 LTE 无线模块，同时也利用它的内嵌 Lua 脚本实现透传功能，如下图所示：



配置好网关的参数，主要是服务器的信息

名称	示例	说明
host	"35.32.85.115"	TCP 服务器地址
port	"9999"	TCP 服务器端口
ktmr	120	连接保持的心跳周期

上电后，它将自动连接到服务器，实现透传功能。

远程登录到主机，运行下面端口转发命令：
socat -v -dd tcp-listen:9904,reuseaddr,keepalive,keepidle=30,keepintvl=12,keepcnt=3,fork tcp-listen:9999,reuseaddr,keepalive,keepidle=30,keepintvl=12,keepcnt=3

然后桌面主机运行 TinyNMS ，它作为 TCP 客户端连接至公网主机的 9904 端口， 而网关通过 LTE 连接至公网主机的 9999 端口，当网关通过 LTE 接入主机后，就可以收到各个节点上报的信息，也可以直接发命令控制节点模块了，就如同在本地使用 USB 接口的开发板一样了。

参考文档

《TinyWSN 网关使用手册》
《TinyWSN 系统报文手册》
《TinyWSN 交互命令手册》
《TinyWSN 内嵌编程手册》
《TinyWSN 配置工具手册》