

无线网络技术教程

——原理、应用与实验(第3版)

配套实验手册

Version 3.4.3

金光、江先亮、邓轲编著

清华大学出版社

2018年9月



<http://www.thinkmesh.net/wireless/>

前言

《无线网络技术教程》(第1版和第2版)出版以来,受到了广大师生的好评,已进入全国超过200所高校。在此基础上,我们修订了《无线网络技术教程:原理、应用与实验》(第3版)(以下简称“教材”)。本实验手册涵盖第3版教材的配套实验内容,与教材中的多种无线网络技术理论知识相对应,包含14个仿真实验和23个实测实验。

读者在使用本手册进行各种无线网络协议的实验前,应当做好如下准备:

1. 已初步学习了相关无线网络技术的背景内容和原理知识,具体见教材相应章节;
2. 对Linux基础知识、NS2和Contiki的使用方法有所了解,具体见本手册实验一;
3. 限于篇幅,本手册大部分实验的理论背景、仿真拓扑、运行场景、结果分析等不一一列出,详细理论背景可以参见教材;

4. 仿真实验采用NS2和Contiki两个开源平台完成,实测实验则需相应硬件器材(如传感器节点、开发板、各种模块等)。本教材第1版和第2版的配套实验均采用师生读者自建实验环境的方式,但许多师生反映,难度不小,工作量大,性能不稳定,效果不尽如人意。所以第3版推出集成化的实验教学系统,详情请看下页。有经验的读者也可自建实验环境。

5. 实测实验基本上都提供了源码,涉及Windows/Android/Linux等系统和C/C++/Java/Python/JavaScript/C#等环境,读者需具备少量基础,通常按照本手册的提示,直接编译运行程序即可。

6. 本实验手册和教材的其它配套电子资源可从清华大学出版社网站或“无线网络技术教学研究平台”(http://www.thinkmesh.net/wireless/)下载,用于教学使用,但不得用于商业用途,请务必尊重作者和出版社的版权;

参与本手册相关实验设计和文档撰写的除作者以外,还包括研究生路金明、张超、高子航、张建锋、尚子宁、庄建辉、朱家骅等。

初学者在本手册实验操作过程中可能会遇到一些问题,如果无法自行解决,也可以与我们进行交流,有问题和建议也请及时反馈。教学网站: <http://www.thinkmesh.net/wireless/>, 联系邮箱是: wireless_net@163.com, 欢迎加入实验教学服务QQ群(627673170)实时交流。

另外,由于本实验手册的电子资源在Windows下整理,每个实验的中文说明文件在实验系统(Ubuntu)中会显示乱码。**请务必在Windows环境中查看README.txt信息。**

考虑到实验教学内容更新较为频繁,我们将不断完善修订,并及时发布在教学网站中。

金光、江先亮、邓轲等

2018年9月



<http://www.thinkmesh.net/wireless/>
欢迎访问无线网络技术教学研究平台

无线网络和物联网实验教学系统

为了方便各校师生顺利开展实验, 我们和湖南新实网络科技有限公司合作开发了“无线网络和物联网实验教学系统”, 该教学系统面向开设“无线网络技术”及相近课程的高校师生, 适用于网络工程、计算机科学与技术、物联网工程、通信工程、软件工程、信息安全、电子、自动化等 IT 各专业。教学系统中包含如下资源:

- (1) 多功能无线网络和物联网实验箱(包含全部硬件);
- (2) 无线网络技术实验手册(目前含 14 项仿真+23 项实测实验, 且不断扩展升级);
- (3) 所有实验项目的源代码、执行文件、说明等;
- (4) 面向各种无线网络协议的仿真实验环境;
- (5) 所有实验的操作视频(支持手机和 PC, 可在线或下载观看);
- (6) 仿真实验的动画示例。

其中, “多功能无线网络和物联网实验箱”为该教学系统的核心部分。实验箱提供统一完整的软件和硬件资源, 和所有配套实验相对应。读者只需按照本实验手册介绍, 即可利用实验箱顺利进行实验, 避免了各位师生需自备器材自搭环境的麻烦。其次, 感兴趣的读者可以利用实验箱中的软硬件资源自行设计科技作品, 深入学习, 完成课程设计、学生科研、毕业设计等。

实验教学系统的购买联系方式如下:

单位名称: 湖南新实网络科技有限公司

联系人: 肖智鹏

网址: www.fastswitch.org

固话: 0731-82033926

手机: 13298658836

QQ: 1013125886

邮箱: MagicLab@magiclab-networks.com



目 录

前 言	2
无线网络和物联网实验教学系统	3
实验一 构建无线网络实验环境	15
0 关于实验系统软硬件环境的说明	15
1 实验要求与目的	15
2 实验背景与原理	15
2.1 实验背景	15
2.2 实验原理	15
3 实验环境与资源	16
3.1 wnt-ns2 环境搭建	16
3.2 wnt-contiki 环境搭建	16
4 实验步骤与演示	17
4.1 wnt-ns2 环境搭建	17
4.2 wnt-contiki 环境搭建	18
6 结果分析与讨论	20
7 注意事项	20
实验二 无线网络环境 RSSI 测量实验	21
1 实验要求与目的	21
2 实验背景与原理	21
2.1 实验背景	21
2.2 实验原理	21
3 实验环境与资源	21
3.1 基础实验	21
3.2 扩展实验	21
4 基础实验步骤与演示	22
4.1 不同距离对 RSSI 值影响	23
4.2 不同高度方位对 RSSI 值影响	24
4.3 不同风速对 RSSI 值的影响	24
4.4 不同阻挡物对 RSSI 值影响	24
4.5 雨天气候对 RSSI 值影响	25
5 扩展实验步骤与演示	25
5.1 不同距离对 RSSI 值影响	28
5.2 不同高度对 RSSI 值影响	28
5.3 不同风速对 RSSI 值影响	29
5.4 不同遮挡物对 RSSI 值影响	29
5.5 不同气候对 RSSI 值影响	30
6 结果分析与讨论	30
7 注意事项	30
实验三 无线局域网组网与管理实验	32
1 实验要求与目的	32
2 实验背景与原理	32
2.1 实验背景	32

2.2 实验原理.....	32
3 实验环境与资源.....	32
3.1 基础实验.....	32
3.2 扩展实验.....	33
4 基础实验步骤与演示.....	33
4.1 无线局域网组网.....	33
4.2 不同 WiFi 信道稳定性测量.....	35
5 扩展实验步骤与演示.....	37
5.1 信道较繁忙时网络传输性能.....	37
5.2 信道较空闲时网络传输性能.....	39
6 结果分析与讨论.....	40
7 注意事项.....	40
实验四 无线局域网信号测量 软件开发实验	41
1 实验要求与目的.....	41
2 实验背景与原理.....	41
3 实验环境与资源.....	41
4 实验步骤与讨论.....	42
4.1 开发平台的配置.....	42
4.2 数据采集模块源码分析.....	43
4.3 数据显示模块源码分析.....	44
4.4 软件执行流程分析.....	44
5 实验结果与分析.....	45
6 扩展与分析.....	45
实验五 无线局域网数据分组分析	46
1 实验要求与目的.....	46
2 实验背景与原理.....	46
3 实验环境与资源.....	46
3.1 基础实验.....	46
3.2 扩展实验.....	47
4 基础实验步骤与演示.....	47
4.1 实验准备.....	47
4.2 Wireshark 用户界面介绍.....	48
4.3 Wireshark 捕获数据包过程.....	49
4.4 结果分析.....	49
5 扩展实验步骤与演示.....	50
5.1 Linux 下无线测量工具 Wireshark.....	50
5.2 Linux 下无线测量工具 Kismet.....	51
6 结果分析与讨论.....	53
7 注意事项.....	53
实验六 无线局域网数据分组分析 软件开发实验	54
1 实验要求与目的.....	54
2 实验背景与原理.....	54
3 实验环境与资源.....	54
4 实验步骤与讨论.....	54

4.1 开发平台的配置.....	54
4.2 程序界面设计.....	56
4.3 协议头部结构定义.....	58
4.4 数据包处理.....	58
4.5 主程序.....	59
5 实验结果与分析.....	59
6 扩展与分析.....	60
实验七 隐藏节点和暴露节点仿真实验.....	61
1 实验要求与目的.....	61
2 实验背景与原理.....	61
2.1 实验背景.....	61
2.2 实验原理.....	61
3 实验环境与资源.....	62
3.1 基础实验.....	62
3.2 扩展实验.....	62
4 基础实验步骤与演示.....	62
4.1 隐藏节点问题仿真.....	62
4.2 暴露节点问题仿真.....	63
5 扩展实验步骤与演示.....	64
5.1 隐藏节点仿真扩展.....	64
5.2 暴露节点问题仿真.....	69
6 结果分析与讨论.....	71
实验八 无线城域网 WiMax 仿真实验.....	72
1 实验要求与目的.....	72
2 实验背景与原理.....	72
2.1 实验背景.....	72
2.2 实验原理.....	72
3 实验环境与资源.....	72
3.1 基础实验.....	72
3.2 扩展实验.....	72
4 基础实验步骤与演示.....	72
5 扩展实验步骤与演示.....	74
5.1 基础实验代码分析.....	74
5.2 修改实验代码.....	76
5.3 运行实验.....	76
6 结果分析与讨论.....	77
实验九 蜂窝移动网络的数据传输实验.....	78
1 实验要求与目的.....	78
2 实验背景与原理.....	78
2.1 实验背景.....	78
2.2 实验原理.....	78
3 实验环境与资源.....	78
4 基础实验步骤与演示.....	79
4.1 拨打/接听电话.....	80

4.2 收发短信.....	81
5 扩展实验步骤与演示.....	83
5.1 访问 HTTP.....	83
6 结果分析与讨论.....	85
7 注意事项.....	85
实验十 卫星网络系统仿真实验.....	86
1 实验要求与目的.....	86
2 实验背景与原理.....	86
2.1 实验背景.....	86
2.2 实验原理.....	86
3 实验环境与资源.....	86
4 实验步骤与演示.....	86
4.1 Iridium 仿真分析.....	86
4.2 Teledesic 仿真分析.....	88
5 实验扩展与分析.....	90
6 结果分析与讨论.....	90
7 注意事项.....	90
实验十一 GPS 室外定位实验.....	91
1 实验要求与目的.....	91
2 实验背景与原理.....	91
2.1 实验背景.....	91
2.2 实验原理.....	91
3 实验环境与资源.....	92
4 基础实验步骤与演示.....	92
4.1 程序烧录.....	92
4.2 硬件配置及连线.....	93
4.3 室外定位.....	93
5 扩展实验步骤与演示.....	94
5.1 实验准备.....	94
5.2 硬件配置及连线.....	96
5.3 测试实验结果.....	97
6 结果分析与讨论.....	97
7 注意事项.....	97
实验十二 AODV 和 DSR 协议仿真实验.....	98
1 实验要求与目的.....	98
2 实验背景与原理.....	98
2.1 实验背景.....	98
2.2 实验原理.....	98
3 实验环境与资源.....	98
3.1 基础实验.....	98
3.2 扩展实验.....	99
4 基础实验步骤与演示.....	99
4.1 AODV 仿真分析.....	99
4.2 DSR 仿真分析.....	100

5 扩展实验步骤与演示.....	101
5.1 基础实验代码分析.....	101
5.2 实验扩展.....	103
6 结果分析与讨论.....	104
7 注意事项.....	104
实验十三 DD 和 S-MAC 协议仿真实验	105
1 实验要求和目的.....	105
2 实验背景与原理.....	105
2.1 实验背景.....	105
2.2 实验原理.....	105
3 实验环境与资源.....	105
3.1 基础实验.....	105
3.2 扩展实验.....	105
4 基础实验步骤与演示.....	106
4.1 DD 协议仿真分析.....	106
4.2 S-MAC 协议仿真分析	107
5 扩展实验步骤与演示.....	108
5.1 基础实验代码分析.....	108
5.2 实验扩展.....	110
6 结果分析与讨论.....	111
7 注意事项.....	111
实验十四 水下无线传感网协议仿真实验.....	112
1 实验要求与目的.....	112
2 实验背景与原理.....	112
2.1 实验背景.....	112
2.2 实验原理.....	112
3 实验环境与资源.....	112
4 基础实验步骤与演示.....	113
5 扩展实验步骤与演示.....	113
5.1 基础实验代码分析.....	113
5.2 实验扩展.....	118
6 结果分析与讨论.....	119
7 注意事项.....	119
实验十五 ZigBee 节点安装和组网基础.....	120
1 实验要求与目的.....	120
2 实验背景与原理.....	120
2.1 实验背景.....	120
2.2 实验原理.....	121
3 实验环境与资源.....	121
4 实验步骤与演示.....	122
4.1 Z-Stack 的安装.....	122
4.2 IAR EW8051 的安装.....	123
4.3 CC Debugger 的驱动安装.....	125
4.4 Smart Flash Programmer 的安装.....	125

4.5	ZigBee_Packet_Sniffer 的安装	126
4.6	ZigBee_Sensor_Monitor 的安装	127
4.7	传感器节点的程序烧录方法	128
4.8	ZigBee 网络数据包分析	129
5	结果分析与讨论	130
6	注意事项	130
实验十六 Arduino 节点安装与开发基础		131
1	实验要求与目的	131
2	实验背景与原理	131
2.1	实验背景	131
2.2	实验原理	131
3	实验环境与资源	131
3.1	Arduino 代码烧写	131
3.2	蓝牙 4.0 模块通信	132
3.3	ZigBee 模块通信	132
4	基础实验步骤与演示	133
4.1	Arduino 代码烧录	133
4.2	蓝牙 4.0 模块通信	134
4.3	ZigBee 模块通信	137
5	结果分析与讨论	140
6	注意事项	140
实验十七 ZigBee 户外环境监测网络		141
1	实验要求与目的	141
2	实验背景与原理	141
2.1	实验背景	141
2.2	实验原理	141
3	实验环境与资源	142
4	基础实验步骤与演示	142
4.1	CC2530 节点程序烧录	142
4.2	单路由单终端网络组建	143
4.3	多路由多终端网络组建	144
4.4	对比性实验环境下实验	144
5	结果分析与讨论	144
6	注意事项	145
实验十八 IEEE 802.15.4 和 ZBR 协议仿真		146
1	实验要求与目的	146
2	实验背景与原理	146
2.1	实验背景	146
2.2	实验原理	146
3	实验环境与资源	146
4	实验步骤与演示	147
4.1	IEEE 802.15.4 仿真分析	147
4.2	ZBR 协议仿真分析	148
5	实验扩展与分析	149

6 结果分析与讨论.....	149
7 注意事项.....	149
实验十九 RFID 和 NFC 数据读写和传输.....	150
1 实验要求和目的.....	150
2 实验原理和背景.....	150
2.1 实验背景.....	150
2.2 实验原理.....	150
3 实验环境和资源.....	150
3.1 RFID 通信实验.....	150
3.2 NFC 读卡通信实验.....	151
4 基础实验步骤与演示.....	152
4.1 RFID 通信实验.....	152
4.2 NFC 读卡通信实验.....	155
5 结果分析与讨论.....	157
6 注意事项.....	157
实验二十 树莓派安装和配置 WiFi 路由器.....	158
1 实验要求与目的.....	158
2 实验背景与原理.....	158
3 实验环境和资源.....	158
4 实验步骤与演示.....	159
4.1 实验准备.....	159
4.2 配置无线网卡.....	160
4.3 安装与配置 hostapd.....	161
4.3 安装与配置 dnsmasq.....	162
5 结果分析与讨论.....	163
6 注意事项.....	164
实验二十一 Contiki 系统测试与组网基础.....	165
1 实验要求与目的.....	165
2 实验背景与原理.....	165
2.1 实验背景.....	165
2.2 实验原理.....	165
3 实验环境与资源.....	165
4 实验步骤和演示.....	165
5 实验扩展和分析.....	168
6 结果分析和讨论.....	168
7 注意事项.....	168
实验二十二 CoAP 协议仿真实验.....	169
1 实验要求与目的.....	169
2 实验背景与原理.....	169
2.1 实验背景.....	169
2.2 实验原理.....	170
3 实验环境与资源.....	170
4 实验步骤与仿真演示.....	170
5 实验扩展与分析.....	173

6 结果分析与讨论.....	173
7 注意事项.....	173
实验二十三 RPL 协议仿真实验	174
1 实验要求与目的.....	174
2 实验背景与原理.....	174
2.1 实验背景.....	174
2.2 实验原理.....	174
3 实验环境与资源.....	174
4 实验步骤与演示.....	174
6 实验扩展与分析.....	177
5 结果分析与讨论.....	178
7 注意事项.....	179
实验二十四 基于 Contiki 和 RPL 的 无线传感网组网实测.....	180
1 实验要求与目的.....	180
2 实验背景与原理.....	180
2.1 实验背景.....	180
2.2 实验原理.....	180
3 实验环境与资源.....	180
4 实验步骤和演示.....	181
5 实验扩展与分析.....	184
6 结果分析与讨论.....	184
7 注意事项.....	184
实验二十五 IEEE 802.11p 和 VANET 仿真	185
1 实验要求与目的.....	185
2 实验背景与原理.....	185
2.1 实验背景.....	185
2.2 实验原理.....	185
3 实验环境与资源.....	185
4 实验步骤与演示.....	186
4.1 IEEE 802.11p 仿真分析	186
4.2 VANET 组网仿真分析.....	187
5 实验扩展与分析.....	188
6 结果分析与讨论.....	188
实验二十六 无线体域网健康监测系统.....	189
1 实验要求与目的.....	189
2 实验背景与原理.....	189
2.1 实验背景.....	189
2.2 实验原理.....	189
3 实验环境与资源.....	189
4 实验步骤和演示.....	190
5 实验扩展与分析.....	194
6 结果分析与讨论.....	194
7 注意事项.....	194
实验二十七 无线室内定位仿真实验.....	195

1 实验要求与目的.....	195
2 实验背景与原理.....	195
2.1 实验背景.....	195
2.2 实验原理.....	195
3 实验环境与资源.....	195
4 实验步骤与演示.....	196
5 实验扩展与分析.....	197
6 结果分析与讨论.....	197
7 参考文献.....	197
实验二十八 WiFi 无线室内定位实践.....	198
1 实验要求与目的.....	198
2 实验背景与原理.....	198
2.1 实验背景.....	198
2.2 实验原理.....	198
3 实验环境与资源.....	199
4 实验步骤与演示.....	200
4.1 实验环境布置.....	200
4.2 实验参数配置.....	200
4.3 定位.....	201
5 结果分析与讨论.....	202
6 扩展与分析.....	202
7 注意事项.....	202
实验二十九 无线网络攻击仿真实验.....	203
1 实验要求与目的.....	203
2 实验背景与原理.....	203
2.1 实验背景.....	203
2.2 实验原理.....	203
3 实验环境与资源.....	203
4 实验步骤与演示.....	204
5 实验扩展与分析.....	205
6 结果分析与讨论.....	205
实验三十 Watchdog 无线网络监测仿真.....	206
1 实验要求与目的.....	206
2 实验背景与原理.....	206
2.1 实验背景.....	206
2.2 实验原理.....	206
3 实验环境与资源.....	206
4 实验步骤与演示.....	206
5 实验扩展与分析.....	207
6 结果分析与讨论.....	208
实验三十一 基于蓝牙和 WiFi 的灯控系统.....	209
1 实验要求与目的.....	209
2 实验背景与原理.....	209
2.1 实验背景.....	209

2.2 实验原理.....	209
3 实验环境与资源.....	209
4 实验步骤与演示.....	210
4.1 蓝牙模式.....	210
4.2 WiFi 模式.....	213
4.3 语音灯控.....	214
5 结果分析与讨论.....	216
6 注意事项.....	216
实验三十二 基于 MQTT 协议的 温度实时监测.....	217
1 实验要求与目的.....	217
2 实验背景与原理.....	217
2.1 实验背景.....	217
2.2 实验原理.....	217
3 实验环境与资源.....	217
4 实验步骤与演示.....	218
4.1 硬件配置.....	218
4.2 开发环境配置.....	219
4.3 实验过程.....	221
5 结果分析与讨论.....	222
6 注意事项.....	222
实验三十三 低功耗广域网物联网(LoRa)数据传输.....	223
1 实验目的和要求.....	223
2 实验背景与原理.....	223
2.1 实验背景.....	223
2.2 实验原理.....	223
3 实验环境与资源.....	224
4 基础实验步骤与演示.....	225
4.1 开发平台的配置.....	225
4.2 LoRa 节点组建.....	226
4.3 LoRa 点对点通信.....	227
5 扩展实验步骤与演示.....	228
5.1 开发平台的配置.....	228
5.2 LoRa 终端节点与 LoRa 网关的组建.....	230
5.3 部署物联网服务器.....	232
5.4 代码烧录及测试.....	233
5 结果分析与讨论.....	235
实验三十四 机器人环境配置与开发基础.....	236
1 实验要求与目的.....	236
2 实验背景与原理.....	236
2.1 实验背景.....	236
2.2 实验原理.....	236
3 实验环境与资源.....	237
4 实验步骤与演示.....	238
4.1 实验准备.....	238

4.2 ROS 多机通信基础	239
4.3 控制机器人基础运动	240
4.4 语音控制机器人	242
5 结果分析与讨论	243
6 注意事项	243
实验三十五 机器人即时定位和地图构建	244
1 实验要求与目的	244
2 实验背景与原理	244
2.1 实验背景	244
2.2 实验原理	244
3 实验环境与资源	245
4 实验步骤与演示	245
4.1 实验准备	245
4.2 基于 gmapping 包的地图构建过程	248
4.3 修正地图与保存地图	250
5 结果分析与讨论	251
6 注意事项	251
实验三十六 基于 WiFi 的无人船控制实验	252
1 实验目的和要求	252
2 实验背景与原理	252
2.1 实验背景	252
2.2 实验原理	252
3 实验环境与资源	252
4 实验步骤与演示	253
4.1 开发板程序烧录	253
4.2 下位机硬件连接	254
4.2 移动端 APP 安装	255
4.3 水面环境下系统实测	255
5 移动端 APP 源码分析	256
6 结果分析与讨论	259
实验三十七 基于 LoRa 的远距离 无人船控制实验	260
1 实验目的和要求	260
2 实验背景与原理	260
2.1 实验背景	260
2.2 实验原理	260
3 实验环境与资源	260
4 实验步骤与演示	261
4.1 开发板程序烧录	261
4.3 开发板硬件连接	262
4.4 水面环境下系统实测	263
5 结果分析与讨论	264

实验一 构建无线网络实验环境

0 关于实验系统软硬件环境的说明

本实验系统的实测实验利用实验箱内各器件和自备 PC/手机进行, NS2 仿真实验是在树莓派中运行, Contiki 实测及仿真实验是在 PC 机中运行。主要考虑如下:

(1) 树莓派除了用于本实验系统外, 还可以用于其他物联网或嵌入式实验的核心控制系统, 便于师生进一步的教学和学生科研;

(2) 许多高校实验室通常已有很多 PC, 许多 PC 机可能已老化(如内存、硬盘空间不够), 且已有分区复杂。但这些旧 PC 的显示器/键盘/鼠标还可继续使用, 所以本实验系统提供了树莓派, 和已有旧显示器/键盘/鼠标可组成一套 Linux 实验教学设备, 发挥更多用途;

(3) 考虑到 Contiki 系统相对独立, 并且 Contiki 相关内容较为深入, 我们建议有关 Contiki 的实验在普通 PC 中运行。

1 实验要求与目的

- 1 人独立完成
- 熟悉 Linux 操作界面和简单命令
- 了解 NS2 仿真实验环境搭建和实验过程
- 了解 Contiki 实验环境搭建和实验过程

2 实验背景与原理

2.1 实验背景

(1) NS2(Network Simulator version 2)由 UC Berkeley 开发, 是源码公开、免费的权威网络仿真平台, 包含模块几乎涉及网络技术所有方面。NS2 为分组级的网络仿真器, 是一个离散事件模拟器。NS2 本身有一个虚拟时钟, 所有仿真由离散事件驱动。可用于仿真各种不同的网络模型, 包括: 网络传输协议, 如 TCP 和 UDP; 业务流量模型, 如 FTP、Telnet、Web CBR 和 VBR 等; 路由器队列管理, 如 Droptail、RED 和 CBQ; 路由算法, 如 Dijkstra 等; 局域网, 多播及 MAC 子层协议; 无线网络协议, 支持许多类型无线网络。有关 NS2 的相关知识请参考《无线网络技术教程: 原理、应用与实验》(第 3 版)(以下简称“教材”)第 2 章, 也可查阅参考文献[1][2][3][4]。

(2) Contiki 为面向物联网(无线传感网)的开源、高可移植、支持网络的多任务操作系统, 适于有内存的嵌入式低端设备。它由一个多任务核心、TCP/IP 协议栈、程序集及低功耗的无线通信栈等组成, 可在每个进程内选择是否支持抢占式多线程, 进程间通信通过事件利用消息来实现。Contiki 采用 C 语言开发, 运行只需几 KB 内存, 包括可选 GUI 子系统, 可提供对本地终端、基于 VNC 的网络化虚拟显示或 Telnet 的图形化支持。Contiki 有关内容参见教材第 9 章, 更多信息详见: <http://www.contiki-os.org/>。

2.2 实验原理

我们建议在树莓派中搭建好本实验手册所涉及到的所有仿真环境, 即进行所有的仿真实验前需要先完成本实验。

为便于读者学习使用, 实验教学系统已搭建好所有仿真环境, 并制作了系统镜像文件 wnt-ns2 和 wnt-contiki。对于 wnt-ns2, 读者需使用 Win32DiskImager 软件, 将之导入树莓派中, 而对于 wnt-contiki, 读者需要使用 VMware Workstation 软件, 运行即可。

3 实验环境与资源

3.1 wnt-ns2 环境搭建

(1) 环境资源:

树莓派套件(树莓派 Raspberry Pi3 一个、8GB TF 卡一张、树莓派电源适配器一个、HDMI 转 VGA 线一根、VGA 线一根)、USB 接口的键盘鼠标一套(需自备)、有源显示器一个(需自备)、PC 一台、wnt-ns2 系统镜像文件、SDFormatter 软件、Win32DiskImager 软件。

(2) 使用说明:

- 各类硬件如图 1.1~1.4 所示, 详细的使用方法见实验步骤;

- wnt-ns2 系统镜像文件为作者制作, 读者可从本实验教学系统官网获取:

<http://www.thinkmesh.net/wireless/>。

- SDFormatter 软件: 作者使用版本为 4.0 版, 安装包大小约为 7MB, 下载方法请见实验资源包 WNTLab/experiments/exp1, 官网下载地址为 <http://www.softpedia.com/get/System/File-Management/SDFormatter.shtml>。

- Win32DiskImager 软件: 作者使用版本为 1.0 版, 安装包大小约为 12MB, 下载方法请见实验资源包 WNTLab/experiments/exp1, 官网下载地址为 <https://sourceforge.net/projects/win32diskimager/>。

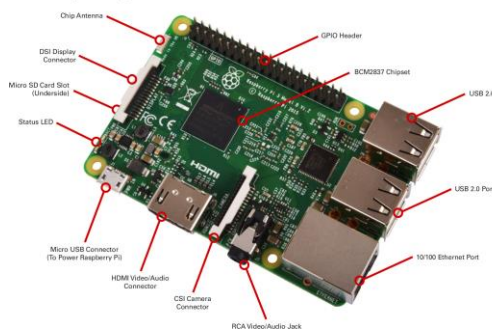


图 1.1 树莓派 Raspberry Pi3



图 1.2 树莓派电源适配器



图 1.3 HDMI 转 VGA 线



图 1.4 8GB TF 卡

3.2 wnt-contiki 环境搭建

(1) 环境资源:

PC 一台、wnt-contiki 系统镜像文件、VMware Workstation 软件。

(2) 使用说明:

- 各类硬件如图 1.1~1.4 所示, 详细使用方法见实验步骤;
- wnt-contiki 系统镜像文件为作者制作, 读者可从本实验教学系统官网获取:

<http://www.thinkmesh.net/wireless/>;

● VMware Workstation 软件: 作者使用版本为 VMware Workstation 12 Pro, 安装包大小约为 470MB, 下载方法请见实验资源包 WNTLab/experiments/exp1, 官网下载地址为:
<https://www.vmware.com/cn.html>;

4 实验步骤与演示

本实验内容: 导入实验系统镜像文件, 并启动仿真环境。

4.1 wnt-ns2 环境搭建

4.1.1 写入系统镜像

(1) 按照 3.1 节中的介绍准备好树莓派套件及相关软件;

(2) 将 TF 卡插入 PC 中, 使用 SDFormatter 软件格式化 TF 卡, 如图 1.5 所示, 格式化完成后拔出 TF 卡并关闭 SDFormatter 软件;



图 1.5 格式化 TF 卡

(3) 开启 Win32DiskImager 软件, 将 wnt-ns2.img 系统镜像烧录到 TF 卡中, 如图 1.6 所示。“镜像文件”选择 wnt-ns2 系统镜像文件, “设备”选择 TF 卡, 然后点击“写入”按钮开始烧录系统, 期间出现对话框, 选择是。出现“Write Successful”提示框时, 说明系统烧录成功;

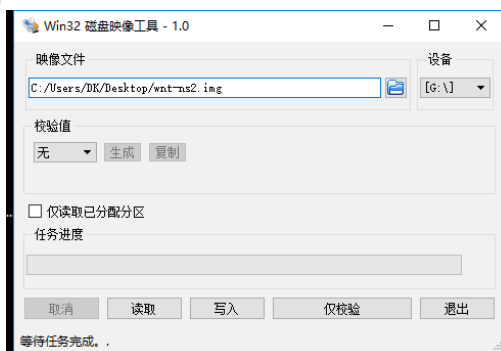


图 1.6 烧录系统镜像

(4) 系统镜像烧录成功后, 将 TF 卡插入树莓派模块的背面, 如图 1.7 红色框处, 至此仿真实验环境已经导入完毕。

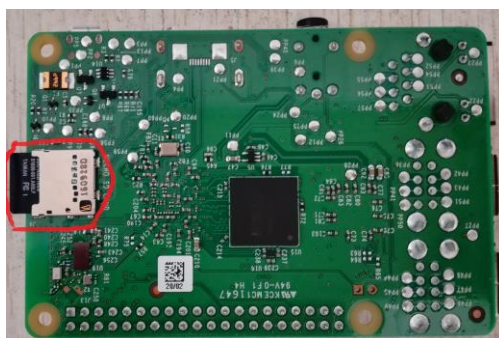


图 1.7 将 TF 卡插入树莓派

4.1.2 启动仿真环境

系统镜像导入完成后, 按照以下步骤启动仿真实验系统:

- (1) 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 并将树莓派连接电源, 如图 1.8 所示;



图 1.8 树莓派连接电源和显示器

- (2) 等待树莓派开机后, 会进入 Ubuntu MATE 系统, 打开文件夹, 可见系统中已经安装了各项仿真环境, 如图 1.9 所示。

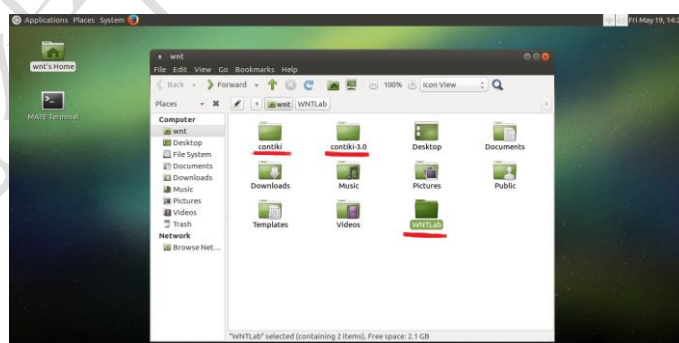


图 1.9 启动仿真实验系统

4.2 wnt-contiki 环境搭建

4.2.1 导入系统镜像

- (1) 按照 3.2 节中的介绍准备好相关软件及资源;
- (2) 将 wnt-contiki 系统镜像(wnt-contiki.zip)解压至任一目录, 解压后的文件如图 1.10 所示;

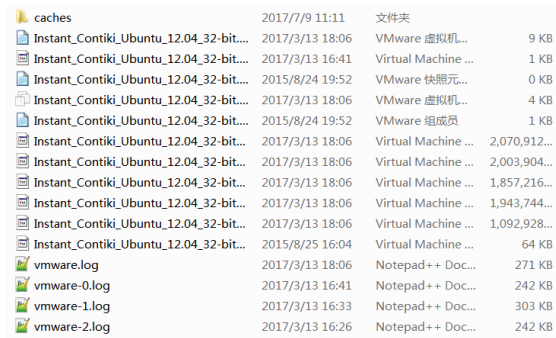


图 1.10 解压 wnt-contiki.zip

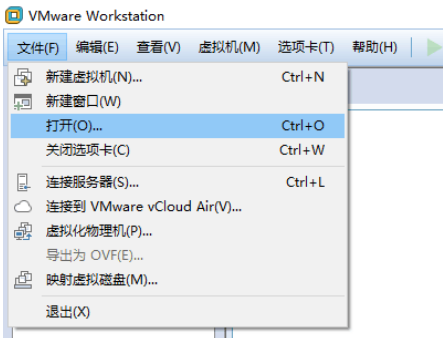


图 1.11 打开并导入 wnt-contiki

(3) 打开 VMware Workstation 软件, 点击菜单栏“文件->打开”按钮, 在弹出的对话框中选中如图 1.12 所示文件并打开;



图 1.12 选择启动文件

(4) 导入成功后可在软件左侧看见 wnt-contiki 虚拟机, 如图 1.13 所示, 此时点击“开启此虚拟机”按钮即可打开 wnt-contiki 实验环境;

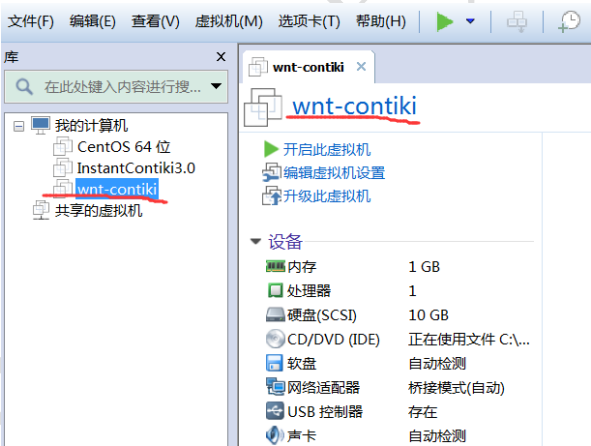


图 1.13 打开 wnt-contiki 实验环境

(5) 打开 wnt-contiki 实验环境后, 打开文件夹, 发现系统中已经安装了各项仿真环境, 如图 1.14 所示。(**需注意**, 虚拟机开机时, 请务必选择以 wnt-contiki 用户登录, 用户密码为: 123456。)

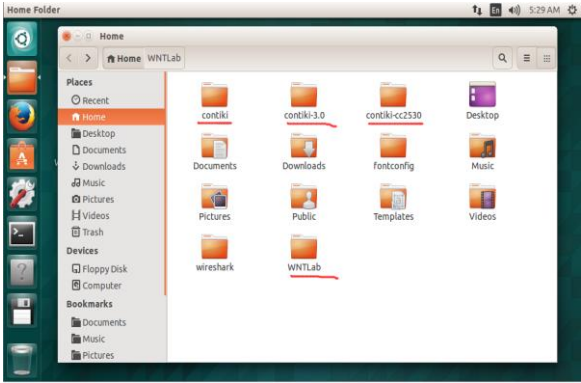


图 1.14 启动 wnt-contiki 实验环境

6 结果分析与讨论

初学者在安装配置实验环境过程中可能会遇到一些问题, 建议使用 Google 等检索相关对策。如果问题无法解决, 可与我们进行交流, 联系邮箱和实验服务 QQ 群见本手册前言。

7 注意事项

请读者务必从官方渠道获取本实验中所使用的各类软件。

无线网络技术教程

实验二 无线网络环境 RSSI 测量实验

1 实验要求与目的

- 2 人共同完成
- 了解 RSSI 的概念和基本原理
- 掌握 RSSI 测量过程和方法
- 在不同无线环境下测量 RSSI 值

2 实验背景与原理

2.1 实验背景

建议读者进行本实验前, 应学习过无线通信的基础知识 (见教材第 2 章)。

针对无线信号测量, 服务集标识符(Service Set Identifier, SSID)用于标识各不同无线网络, 接收信号强度指示(Received Signal Strength Indicator, RSSI)表征无线网络信号质量。

其中 RSSI 值对通话质量、切换、拥塞及网络覆盖、容量等均有显著影响。一般用 dBm 来度量 RSSI, 无线信号多为 mW 级, 故对其进行极化, 转化为 dBm。1mW 即 0dBm, 小于 1mW 就是负数 dBm 值。一般来说, RSSI 值越大, 则接收信号强度越大, 例如 -50dBm 与 -65dBm 比较, 前者表示接收信号的强度大于后者。

2.2 实验原理

基础实验中利用自行开发的软件, 测量不同环境下 WiFi 网络的 RSSI 值, 总结出环境变化对 WiFi RSSI 值的影响。

扩展实验则是对 ZigBee 网络 RSSI 值的测量, 向两个 CC2530 节点中烧录对应代码, 以测量环境变化对 ZigBee 网络 RSSI 值的影响。(进行扩展实验前需先完成实验十五, 也可等课程进度达到实验十五后再完成本实验扩展部分)

3 实验环境与资源

3.1 基础实验

(1) 环境资源:

PC (笔记本) 一台、无线网卡一个、智能手机一部、WiFi RSSI 值测量软件。

(2) 使用说明:

- 无线网卡如图 2.1 所示, 若读者实验所用 PC 无内置无线网卡, 则使用该无线网卡。第一次使用时需安装驱动, 详见实验资源包 WNTLab/experiments/exp2。

- WiFi RSSI 值测量软件提供代码与注释, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp2。

3.2 扩展实验

(1) 环境资源:

PC (笔记本) 一台、CC2530 传感器节点套件两套及软件环境(详见实验十五第 3 节)、

电池及 5V 稳压模块一套、SSCOM 串口调试助手软件、ZigBee RSSI 值测量程序。

(2) 使用说明:

- 电池及 5V 稳压模块如图 2.2 所示。
- SSCOM 串口调试助手: 作者使用版本为 5.12 版, 压缩包大小约 400KB, 下载方法请见实验资源包 WNTLab/experiments/exp2, 官方下载地址为: <http://www.daxia.com/bibis/>。
- ZigBee RSSI 值测量程序提供源代码与注释, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp2。



图 2.1 无线网卡



图 2.2 电池

4 基础实验步骤与演示

基础实验内容为测量 WiFi 的 RSSI 值; 实验要求 2 人共同完成, 其中 1 人持智能手机移动, 另 1 人在 PC 端观察实验结果。

在测量 WiFi 的 RSSI 值时, 实验采用了自行开发的软件, 在 PC 中通过 Windows 系统的 WLAN API 读取无线网卡的信息, 以探测附近无线接入点(Access Point, AP)的 RSSI 值。具体步骤如下:

- (1) 根据 3.1 节获取测量 WiFi 的 RSSI 值的软件: WiFi-RSSI.exe;
- (2) 开启智能手机的 WLAN 热点服务, 并设置 SSID(即热点名称), 这里以“AP1”为例。
(**需注意, 设置的 SSID 应与他人区分, 且 SSID 中不能有中文或特殊字符。**);
- (3) 如果使用台式机 PC 没有自带无线网卡, 需将外置无线网卡插在 PC 的 USB 接口, 并确认 PC 的无线网络中能搜索到自己的 WLAN 热点, 如图 2.3 所示; 如果 PC 自带无线网卡, 跳过该步骤;



图 2.3 PC 中搜索到的 WiFi 信号

- (4) 在 PC 中打开 WiFi-RSSI.exe 软件, 会出现如图 2.4 所示界面, 按照提示输入 SSID 后, 扫描到相应 WiFi 信号并测量出对应 RSSI 值, 如图 2.5 所示;



图 2.4 打开 WiFi-RSSI.exe

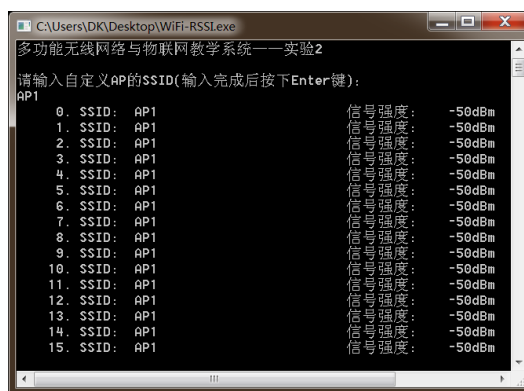


图 2.5 输入 SSID 后扫描到对应信号

(5) 关闭 WiFi-RSSI.exe 软件, 依次按照 4.1 节至 4.5 节所述改变实验环境, 再次打开 WiFi-RSSI.exe 并输入 SSID 测量 RSSI 值。(需注意, 每次改变环境再次测量时, 都需要关闭并再次打开 WiFi-RSSI.exe, 详见本实验第 7 节第(2)点。)

4.1 不同距离对 RSSI 值影响

在空旷区域中, 选择不同距离测量 WiFi 信号的 RSSI 值。本实验主要选择 6 种不同距离进行测量, 实验结果如下:

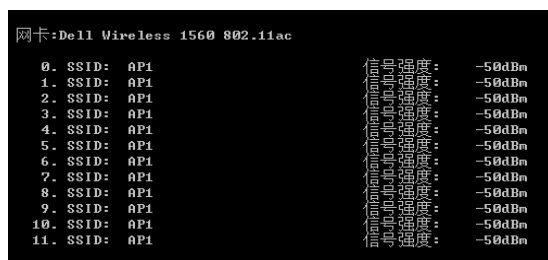


图 2.6 距离 1 米时的 RSSI 值

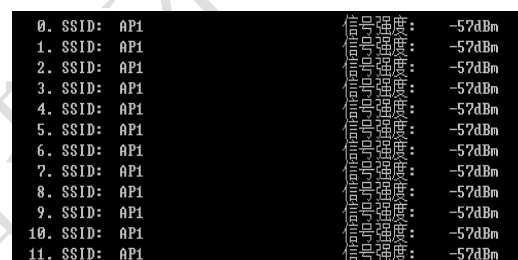


图 2.7 距离 5 米时的 RSSI 值



图 2.8 距离 10 米时的 RSSI 值

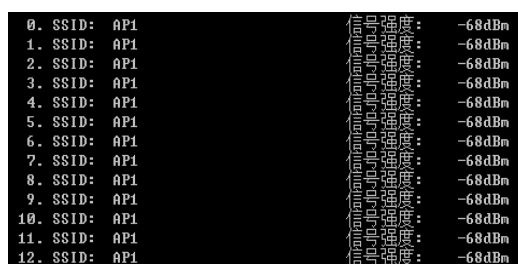


图 2.9 距离 20 米时的 RSSI 值



图 2.10 距离 40 米时的 RSSI 值

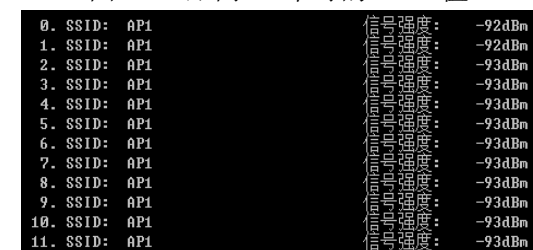


图 2.11 距离 55 米时的 RSSI 值

由上述结果可看出, 随着距离增加, WiFi 的 RSSI 值逐渐减小; 距离越远, 信号越弱。大约 57 米(仅供参考, 应以读者实际实验为准)左右时, 出现大量丢包、连接中断等现象。

4.2 不同高度方位对 RSSI 值影响

为测量不同高度对 WiFi 的 RSSI 值影响, 设置场景一为设置水平距离约 3.5 米、高度差 2 米、直线距离 4 米, 另一场景为距离 4 米的同一水平高度。高度差 2 米又分为 AP 为上方和 AP 为下方 2 种情况。如图 2.12, 2.13 和 2.14 所示, AP1 处于高处, 信号相对较好, AP1 处于下方, 信号相对较差。读者可更多尝试高度差变化时, RSSI 值的动态变化情况, 并思考为什么会有这种结果。

0. SSID: AP1	强度: -54dBm
1. SSID: AP1	强度: -54dBm
2. SSID: AP1	强度: -54dBm
3. SSID: AP1	强度: -54dBm
4. SSID: AP1	强度: -54dBm
5. SSID: AP1	强度: -54dBm
6. SSID: AP1	强度: -54dBm
7. SSID: AP1	强度: -54dBm
8. SSID: AP1	强度: -54dBm
9. SSID: AP1	强度: -54dBm
10. SSID: AP1	强度: -54dBm

图 2.12 相同水平高度时的 RSSI 值

0. SSID: AP1	强度: -53dBm
1. SSID: AP1	强度: -53dBm
2. SSID: AP1	强度: -53dBm
3. SSID: AP1	强度: -53dBm
4. SSID: AP1	强度: -53dBm
5. SSID: AP1	强度: -53dBm
6. SSID: AP1	强度: -53dBm
7. SSID: AP1	强度: -53dBm
8. SSID: AP1	强度: -53dBm
9. SSID: AP1	强度: -53dBm
10. SSID: AP1	强度: -53dBm

图 2.13 高度差 2 米, AP1 居上方的 RSSI 值

0. SSID: AP1	强度: -60dBm
1. SSID: AP1	强度: -60dBm
2. SSID: AP1	强度: -60dBm
3. SSID: AP1	强度: -60dBm
4. SSID: AP1	强度: -60dBm
5. SSID: AP1	强度: -60dBm
6. SSID: AP1	强度: -60dBm
7. SSID: AP1	强度: -60dBm
8. SSID: AP1	强度: -60dBm
9. SSID: AP1	强度: -60dBm
10. SSID: AP1	强度: -60dBm

图 2.14 高度差 2 米, AP1 居下方的 RSSI 值

4.3 不同风速对 RSSI 值的影响

本实验选择顺风 and 逆风情况下, 距离 5 米的 RSSI 值测量, 如图 2.15 和 2.16 所示。

0. SSID: AP1	强度: -51dBm
1. SSID: AP1	强度: -51dBm
2. SSID: AP1	强度: -51dBm
3. SSID: AP1	强度: -51dBm
4. SSID: AP1	强度: -51dBm
5. SSID: AP1	强度: -51dBm
6. SSID: AP1	强度: -51dBm
7. SSID: AP1	强度: -51dBm
8. SSID: AP1	强度: -51dBm
9. SSID: AP1	强度: -51dBm
10. SSID: AP1	强度: -51dBm

图 2.15 顺风 3m/s、距离 5 米的 RSSI 值

0. SSID: AP1	强度: -56dBm
1. SSID: AP1	强度: -56dBm
2. SSID: AP1	强度: -56dBm
3. SSID: AP1	强度: -56dBm
4. SSID: AP1	强度: -56dBm
5. SSID: AP1	强度: -56dBm
6. SSID: AP1	强度: -56dBm
7. SSID: AP1	强度: -56dBm
8. SSID: AP1	强度: -56dBm
9. SSID: AP1	强度: -56dBm
10. SSID: AP1	强度: -56dBm

图 2.16 逆风 3m/s、距离 5 米的 RSSI 值

处于顺风下比逆风的 RSSI 值相对较大些, 风速大小对 RSSI 值也有影响。

4.4 不同阻挡物对 RSSI 值影响

本实验在距离 5 米条件下, 测量了不同阻挡物情况下的 RSSI 值。图 2.17 是全封闭的测量环境示意图, 图 2.18 为测量结果。可看出, RSSI 值约为 -69dBm, 比图 2.7 的空旷环境中信号弱, 表明封闭环境对 RSSI 值有一定影响。

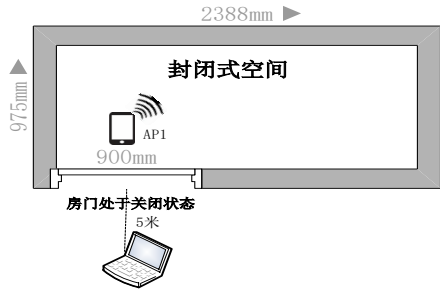


图 2.17 全封闭式墙体示意图

0.	SSID: AP1	强度: -69dBm
1.	SSID: AP1	强度: -69dBm
2.	SSID: AP1	强度: -69dBm
3.	SSID: AP1	强度: -69dBm
4.	SSID: AP1	强度: -69dBm
5.	SSID: AP1	强度: -69dBm
6.	SSID: AP1	强度: -69dBm
7.	SSID: AP1	强度: -69dBm
8.	SSID: AP1	强度: -69dBm
9.	SSID: AP1	强度: -69dBm
10.	SSID: AP1	强度: -69dBm
11.	SSID: AP1	强度: -69dBm
12.	SSID: AP1	强度: -70dBm

图 2.18 全封闭式墙体下的 RSSI 值

图 2.19 是封闭金属盒环境(如金属茶叶盒)，图 2.20 为测量的结果。可看出，RSSI 值约为-81dBm，信号极弱，说明金属封闭环境对信号的阻挡更大。

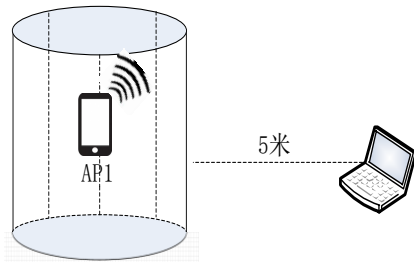


图 2.19 金属壳包裹式示意图

0.	SSID: AP1	强度: -80dBm
1.	SSID: AP1	强度: -80dBm
2.	SSID: AP1	强度: -80dBm
3.	SSID: AP1	强度: -80dBm
4.	SSID: AP1	强度: -80dBm
5.	SSID: AP1	强度: -81dBm
6.	SSID: AP1	强度: -81dBm
7.	SSID: AP1	强度: -81dBm
8.	SSID: AP1	强度: -81dBm
9.	SSID: AP1	强度: -81dBm
10.	SSID: AP1	强度: -81dBm
11.	SSID: AP1	强度: -81dBm
12.	SSID: AP1	强度: -81dBm

图 2.20 金属盒包裹下的 RSSI 值

4.5 雨天气候对 RSSI 值影响

为分析下雨对 RSSI 值的影响，实验在中雨天气距离 10 米的场景中进行测量，图 2.21 为测量结果。与图 2.8 对比，雨天环境对 RSSI 值有一定影响。

0.	SSID: AP1	强度: -68dBm
1.	SSID: AP1	强度: -68dBm
2.	SSID: AP1	强度: -68dBm
3.	SSID: AP1	强度: -68dBm
4.	SSID: AP1	强度: -68dBm
5.	SSID: AP1	强度: -68dBm
6.	SSID: AP1	强度: -68dBm
7.	SSID: AP1	强度: -68dBm
8.	SSID: AP1	强度: -68dBm
9.	SSID: AP1	强度: -68dBm
10.	SSID: AP1	强度: -68dBm
11.	SSID: AP1	强度: -68dBm
12.	SSID: AP1	强度: -68dBm

图 2.21 雨天环境 10 米距离的 RSSI 值

5 扩展实验步骤与演示

扩展实验内容为测量 ZigBee 网络的 RSSI 值；实验要求 2 人共同完成，其中 1 人持终端节点移动，另 1 人持协调器节点在 PC 端观察实验结果。（需注意，进行该实验之前，读者需先完成实验十五，即配置好 ZigBee 网络的实测环境。）

如果已按实验十五配置好相关环境，下面介绍如何编译并向 CC2530 传感器节点烧录代码，具体步骤如下：

(1) 根据 3.2 节准备好所有环境资源，两个 CC2530 节点中一个准备烧录协调器代码，另一个准备烧录终端节点代码；

(2) 连接 CC Debugger 调试器，将下载线和 USB A 型转 B 型线分别与 CC Debugger 调试器相连，如图 2.22 所示；



图 2.22 连接 CC Debugger 调试器

(3) 连接 CC Debugger 调试器完成后, 将下载线一端连接在 CC2530 节点上, 将 USB 一端连接至 PC 的 USB 口; 使用 Mini USB 线连接 CC2530 节点的 Mini USB 口与 PC 的 USB 口, 按下 CC Debugger 的 reset 键, 显示灯为绿色, 连接完成后如图 2.24 所示;

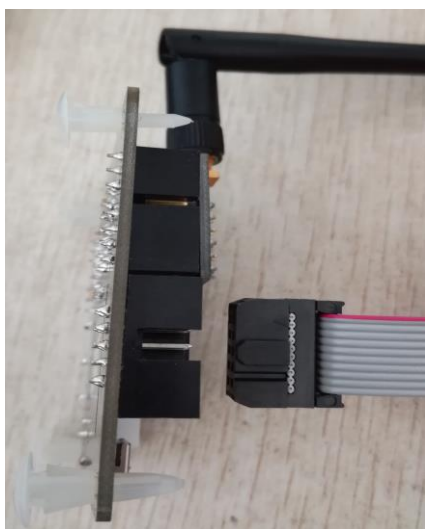


图 2.23 下载线与 CC2530 节点连接



图 2.24 节点与 PC 的连接

(4) 在 PC 中打开 ZigBee RSSI 值测量程序的工程文件。目录为 ZigBee-RSSI\Projects\zstack\Samples\SendTest RSSI\CC2530DB\, 文件名为 SendTest.eww。双击打开, IAR EW 软件自动导入导入工程项目;

(5) 打开工程后, 在左上角选择 CoordinatorEB (协调器节点选择该项, 如果是终端节点或路由器节点, 请选择 EndDeviceEB 或 RouterEB, 下述过程以协调器代码的编译和烧录为例), 并在项目上单击右键, 进行 Rebuild All 操作, 完成协调器节点代码编译, 如图 2.25 所示 (需注意, 编译完成后要确保没有错误, 可以查看软件下方的输出提示。);

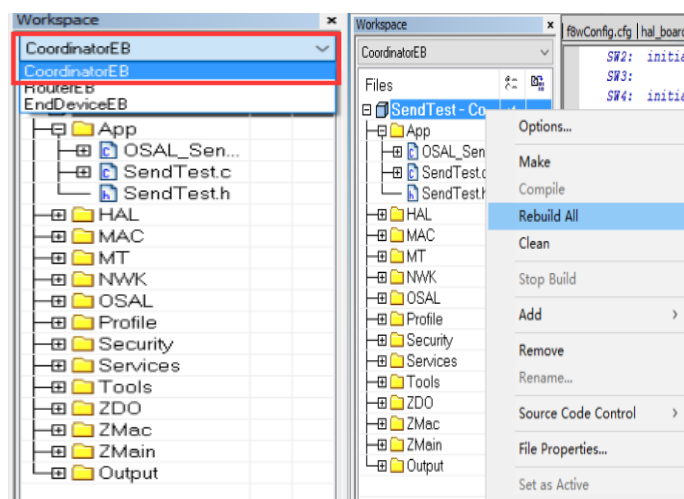


图 2.25 CC2530 节点代码编译

(6) 正确编译后, 选择工具栏中的下载按钮(图 2.26 中黑色方框处)进行代码烧录;

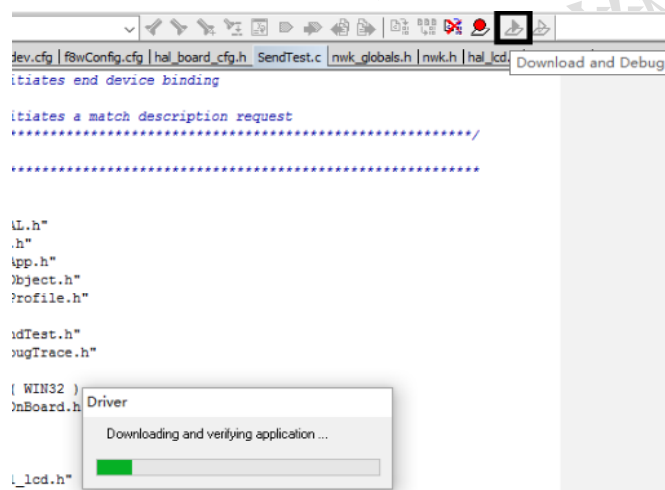


图 2.26 开始烧录代码

另外, 测量 ZigBee 的 RSSI 值还需要终端节点, 因此, 需要重复上述步骤, 在第(3)步中选择 EndDeviceEB, 编译并烧录终端节点代码。(需注意, 代码烧录过程中如果出现错误请参考本实验第7节第(3)点。)

完成上述配置后, 即可进行实际环境中的测量, 具体过程如下:

(1) 使用 Mini USB 线连接协调器节点与 PC, 如图 2.27 所示; 将电池及 5V 稳压模块连接在终端节点的外接电源座处, 如图 2.28 所示;



图 2.27 协调器节点与 PC 连接



图 2.28 终端节点与电池连接

(2) 打开 PC 的设备管理器, 查看设备对应的端口号, 如图 2.29 所示;

(3) 打开 SSCOM 串口调试助手, 选择步骤(2)中查看到的端口号, 设置波特率为 38400,

打开端口, 如图 2.30 所示;



图 2.29 查看设备端口号

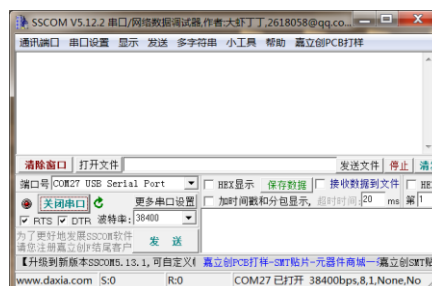


图 2.30 打开 SSCOM 串口调试助手

(4) 先开启协调器节点, 后开启终端节点, 等待数秒, 协调器节点与终端节点连接成功后, SSCOM 串口调试助手中会显示测量得到的 RSSI 值, 详见 5.1 节至 5.5 节;

5.1 不同距离对 RSSI 值影响

在完成前述过程后, 接下来测量空旷环境中, ZigBee 节点不同距离时的 RSSI 值。该实验中, 分别设置协调器节点与终端节点的距离为 1 米、5 米、20 米和 30 米。从图 2.31 至图 2.34 可以看出, 在距离 1 米时, RSSI 为-51dBm, 而当距离逐渐增加时, RSSI 逐渐减小。(注意, 此处测量结果因具体实验环境不同而不同。)

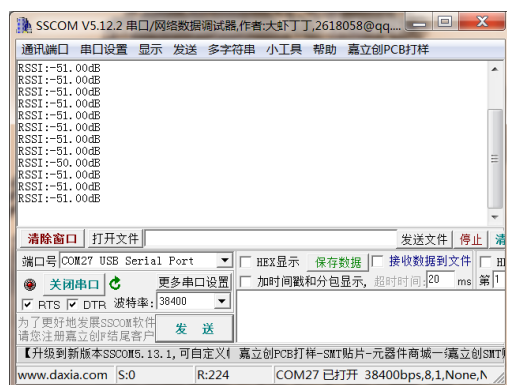


图 2.31 1 米距离下的 RSSI 值

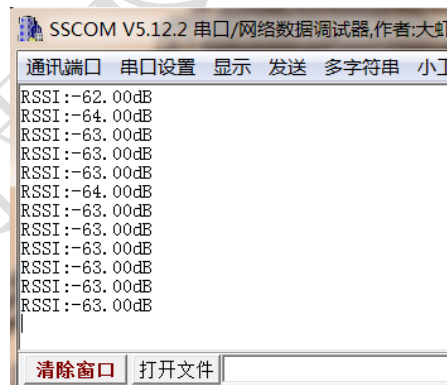


图 2.32 5 米距离下的 RSSI 值

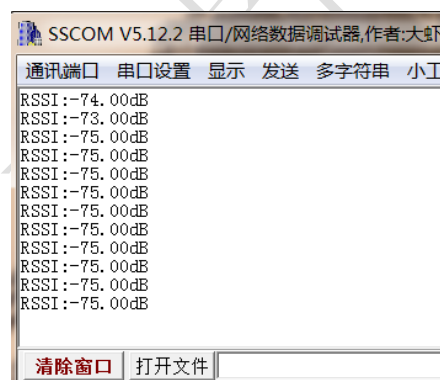


图 2.33 20 米距离下的 RSSI 值



图 2.34 30 米距离下的 RSSI 值

5.2 不同高度对 RSSI 值影响

为测量不同高度对 ZigBee 的 RSSI 值的影响, 分别在以下两种情况下进行实验:

情况一: 协调器节点与终端节点在同一水平高度, 且距离为 5m;

情况二: 协调器节点在终端节点的正上方, 二者高度差为 5m;

从图 2.35 和图 2.36 可看出, 协调器处于高位置时, 信号相对较好。读者可尝试高度差变化时, RSSI 值的动态变化情况。

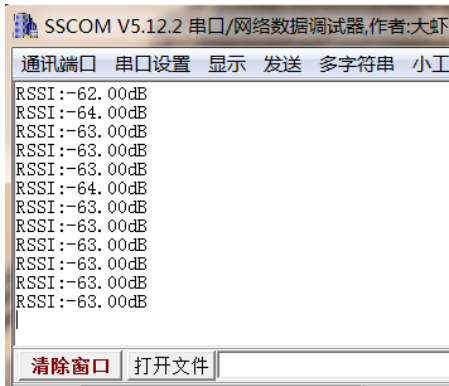


图 2.35 相同高度时的 RSSI 值



图 2.36 高度差 2 米时的 RSSI 值

5.3 不同风速对 RSSI 值影响

为测量风速对 ZigBee 的 RSSI 值影响, 选择风速 5m/s 且节点距离 5 米时, 不同吹风点情况下的 RSSI 值。从图 2.37 和图 2.38 可看出, 风速对 RSSI 值有少量影响。



图 2.37 风速情况 1



图 2.38 风速情况 2

5.4 不同遮挡物对 RSSI 值影响

为测量障碍物遮挡对 ZigBee 的 RSSI 影响, 实验分别设置如图 2.39 和图 2.40 两个场景。从图 2.41 和图 2.42 可以看出, 墙体对于信号的阻碍显而易见, 而金属外壳对于信号的阻碍更大。



图 2.39 封闭墙体场景

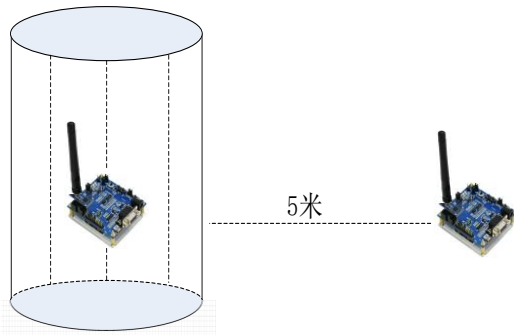


图 2.40 封闭金属壳场景

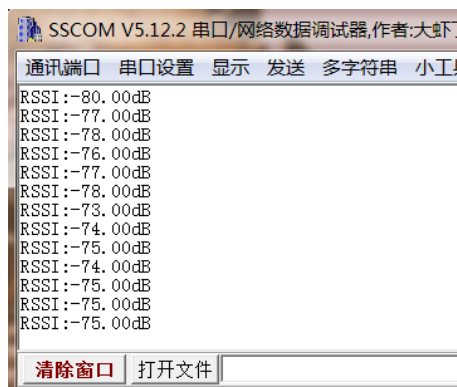


图 2.41 封闭墙体对 RSSI 值影响



图 2.42 封闭金属壳对 RSSI 值影响

5.5 不同气候对 RSSI 值影响

实验在雨天环境进行测量(中雨), 两节点距离分别为 20 米和 30 米, 得到图 2.43 和图 2.44 所示结果。通过对比本节测量结果与 5.1 节测量结果可以看出, 湿度和雨水对信号有不利影响。

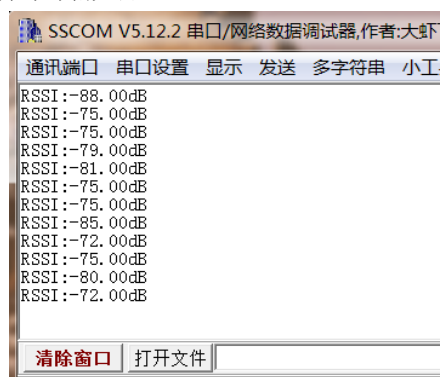


图 2.43 20 米时的 RSSI 值

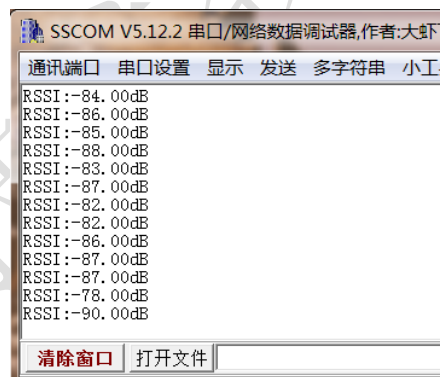


图 2.44 30 米时的 RSSI 值

6 结果分析与讨论

不同设备的 RSSI 受多种因素的影响, 导致 RSSI 出现动态性。本实验测量 WiFi 和 ZigBee 的 RSSI 受环境变化的影响, 注意由于实验测量粒度较粗, 存在一定误差。

7 注意事项

(1) 实验过程中会出现 RSSI 异常情况, 原因可能为: 硬件不可靠, 如安装不当、接头不紧、天线进水、设备老化等; 参数设置不合理, 如初始接入功率设置过大、业务信道最小增益过大等; 外部终端干扰, 如手机信号的干扰。

(2) 因为读者 PC 的操作系统不一, 系统并不一定会自动刷新扫描到的 WiFi 信号, 或者很长时间再刷新一次, 建议每次改变实验环境时需要关闭 WiFi-RSSI.exe 后再次打开。

(3) 第 5 节中烧录代码时如果出现以下错误或问题请参考以下方法, 如果读者发现其它错误可自行查阅解决, 也可反馈给我们, 联系邮箱为: wireless_net@163.com。

a. 如图 2.45 所示错误, 请确认 CC2530 节点的电源开关已打开, 再按下 CC Debugger 的 reset 键并确认其指示灯为绿色, 最后点击对话框中的确定按钮即可;

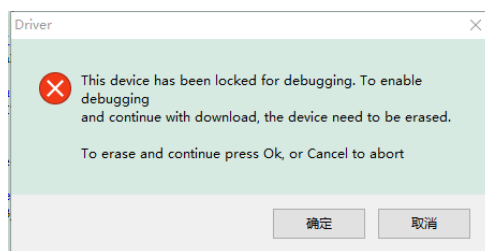


图 2.45 无法调试错误

b. 如图 2.46 所示问题, 此为正常现象, 请不要关闭程序, 耐心等待;

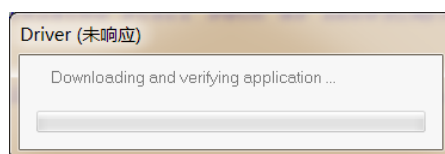


图 2.46 未响应问题

c. 如图 2.47 所示错误, 此错误原因在于读者未进行 Rebuild All 操作, 请仔细根据实验步骤执行 Rebuild All 后重新烧录;

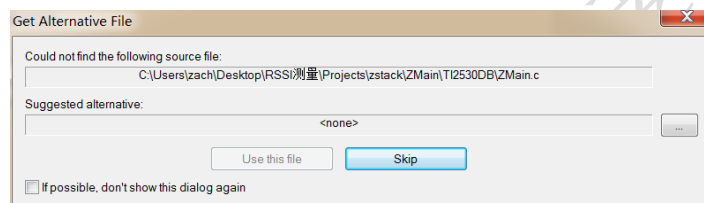


图 2.47 找不到源文件错误

实验三 无线局域网组网与管理实验

1 实验要求与目的

- 1人独立完成基础实验, 4人共同完成扩展实验
- 掌握无线路由器配置方法与步骤
- 学习如何根据无线信道质量更换 AP 的信道

2 实验背景与原理

2.1 实验背景

无线信道指以辐射无线电波为传输方式的数据信道。在 WLAN 中, WiFi 等采用 2.4/5GHz 频段的信道进行数据传输, WiFi 信道分配等知识内容见教材第 3 章。日常工作中, 使用 WiFi 时经常遇到突然变慢的情况, 其由多方面因素造成, 其中无线信号互相干扰为其中之一, 本实验也针对该问题展开分析。

实验中主要用到 inSSIDer 和 iPerf 两个软件, 二者简介如下:

inSSIDer 由 MetaGeek 公司开发的软件, 可用于搜索 WiFi 热点, 并收集它们的详细信息, 以排查 WiFi 网络问题。搜索到 WiFi 热点后, 会详细显示热点的 MAC 地址、制造商、信道、服务设置标识符(SSID)或网络公共名称、安全类型、网速等。此外, 还可显示网络现有信号强度, 以及信号强度随时间的变化情况。

iPerf 为网络性能测试工具, 拥有多个可选参数, 可测量 TCP 和 UDP 的带宽、时延、抖动及丢包率等。在 Windows 中可直接运行 iPerf 的可执行文件, 无需安装; 而在 Ubuntu 系统中可采用命令(sudo apt-get install iperf)进行安装, 然后在命令行中运行。

2.2 实验原理

基础实验包含两部分: 首先对无线路由器进行配置(包括 SSID、密码、信道以及 DHCP 等), 以此组建无线局域网; 再利用 inSSIDer 软件测量不同信道下 WiFi 信号的稳定性, 如 WiFi 信号不稳定, 在路由器的配置系统中更换较为空闲的信道。

扩展实验中利用 iPerf 软件测量信道较为繁忙和较为空闲情况下的网络传输速率: 在服务器上利用该软件监听三个客户端, 客户端同时向服务器传输数据, 以此测量每条数据流的传输速率以及总传输带宽。

3 实验环境与资源

3.1 基础实验

(1) 环境资源:

PC 一台、无线网卡一个、无线路由器(需连接 Internet)一个、树莓派电源适配器一个、inSSIDer 软件。

(2) 使用说明:

- 各类硬件如图 3.1 至 3.3 所示。
- 无线网卡如图 3.1 所示, 若读者实验所用 PC 无内置无线网卡, 则使用该无线网卡。

第一次使用时需要安装驱动, 详见实验资源包 WNTLab/experiments/exp3。

●inSSIDer 软件官网下载地址: <http://www.metageek.com/support/downloads/>, 详细的下载方法及说明请见实验资源包 WNTLab/experiments/exp3。

3.2 扩展实验

(1) 环境资源:

PC 四台、无线网卡四个、无线路由器(需连接 Internet)一个、树莓派电源适配器一个、iPerf 软件。

(2) 使用说明:

- 各类硬件如图 3.1 至 3.3 所示;
- iPerf 软件: 作者使用版本为 3.0.12 版, 压缩包大小约为 1.3MB, 下载方法请见实验资源包 WNTLab/experiments/exp3, 官网下载地址为: <https://iperf.fr/>。



图 3.1 无线网卡



图 3.2 无线路由器



图 3.3 树莓派电源适配器

4 基础实验步骤与演示

基础实验内容为组建无线局域网, 以及使用 inssider 软件测量 WiFi 信号稳定性; 实验要求 1 人独立完成。

4.1 无线局域网组网

- (1) 根据 3.1 节准备好所有的环境资源;
- (2) 将无线路由器的“LAN/WAN”口连接值 Internet, 将“Power”口连接树莓派电源适配器, 如图 3.4 所示;
- (3) 给无线路由器供电, 其指示灯闪烁一次, 等待数秒后指示灯常亮, 表示无线路由器已开机, 如图 3.5 所示。长按无线路由器的 Reset 键 5 秒以上, 指示灯闪烁后无线路由器恢复出厂设置成功;



图 3.4 连接无线路由器



图 3.5 无线路由器开机

- (4) 如果 PC 没有无线网卡, 将无线网卡插在 PC 的 USB 接口; 如果 PC 自带无线网卡,

跳过该步骤:

- (5) 此时 PC 可搜索到无线路由器的信号, 如图 3.6 所示, 连接该无线网络;
- (6) 查看无线路由器背面标识的路由器 IP, 如图 3.7 所示, 此处为 192.168.1.253;



图 3.6 连接无线网络



图 3.7 路由器 IP

- (7) 在 PC 中的浏览器地址栏输入第(6)步中得到的路由器 IP 地址, 并进入;
- (8) 根据提示设置管理员密码后, 即进入无线路由器配置系统, 如图 3.9 所示;



图 3.8 设置管理员密码



图 3.9 无线路由器配置系统

4.1.1 无线路由器基本配置

- (1) 无线工作模式设置: 点击左侧菜单栏中“工作模式”项, 选择模式为“AP: 接入点模式”, 并且点击保存, 如图 3.10 所示;

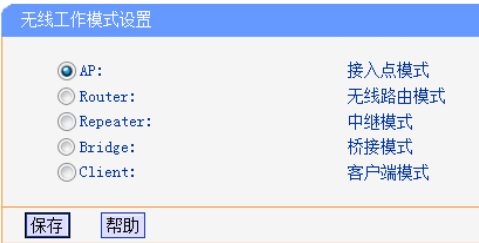


图 3.10 无线工作模式设置

- (2) 基本设置: 点击左侧菜单栏中“无线设置-基本设置”项, 配置 SSID 号、信道、模式、频段带宽等信息完毕后, 并且点击保存, 如图 3.11 所示;
- (3) DHCP 服务设置: 点击左侧菜单栏中“DHCP 服务器-DHCP 服务”选项卡, 选择 DHCP

服务器为启用状态, 设置无线局域网的地址范围, 使接入该 AP 的设备能够动态获取 IP 地址, 并且点击保存, 如图 3.12 所示。

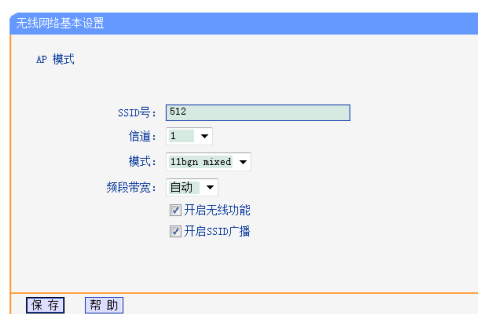


图 3.11 基本设置



图 3.12 DHCP 配置

4.1.2 无线路由器安全配置

完成 4.1.1 节配置后, 路由器已可正常工作, 但处于无保护和无认证状态。接下来介绍安全性配置。具体设置过程: 点击左侧菜单栏中“无线设置-无线安全设置”项, 选择认证模式“WPA-PSK/WPA2-PSK”、设置认证类型为“自动”、加密算法为 AES, 然后设置 PSK 密钥和组密钥更新周期。设置完成后如图 3.13 所示。



图 3.13 安全配置

4.2 不同 WiFi 信道稳定性测量

为了测量 WiFi 信号的稳定性, 实验采用 1 台配置了无线网卡和 inSSIDer 软件的 PC, inSSIDer 运行界面如图 3.14 所示 (**注意**, 关于无线网卡的说明请参考注意事项与资源。)。单击“开始扫描”按钮, inSSIDer 会扫描周围环境中的 WiFi 信号, 并显示其对应的路由器 MAC 地址、SSID 或网络公共名称、所用信道、安全类型、网速等信息, 如图 3.15 所示。

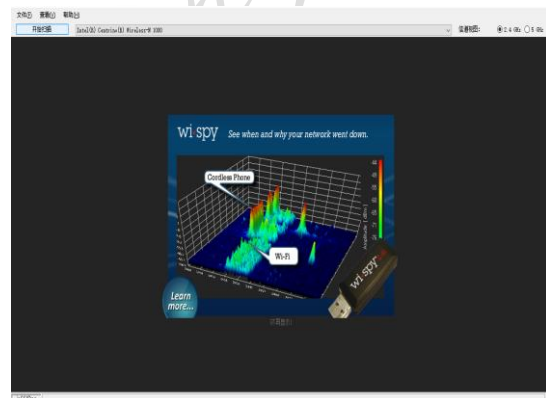


图 3.14 inSSIDer 软件初始界面图



图 3.15 扫描到 WiFi 信号信息图

图 3.15 底部图表示 WiFi 信号占用信道的情况。放大信道位图观察, 发现大部分 WiFi 信号为 1、6、11 信道, 其它信道几乎未使用。实验 PC 所连接的 WiFi 热点(SSID: 512)为信道 1, 且当前占用信道 1 较多, 信道可能发生拥堵, 即 WiFi 信号易受干扰。

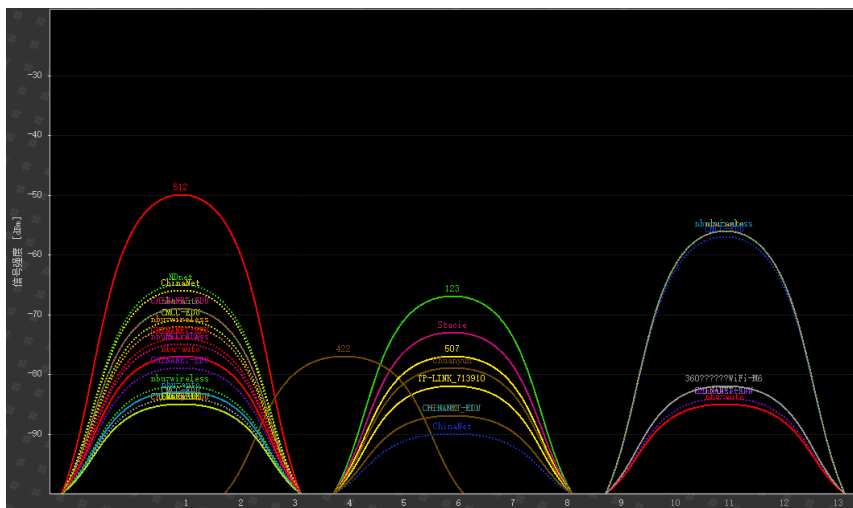


图 3.16 WiFi 信号占用信道的情况

时间位图表示周围 WiFi 信号强度随时间的变化情况。放大时间位图发现 PC 所连接 WiFi 信号(SSID: 512)的强度随时间变化有一定波动。

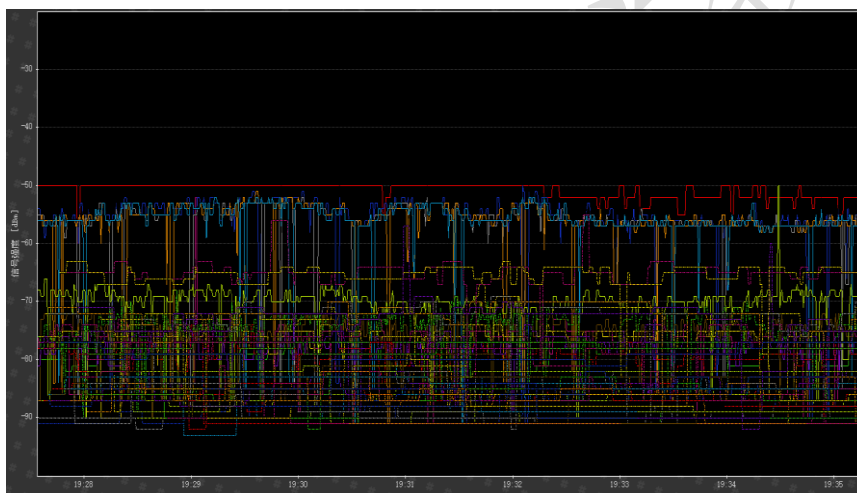


图 3.17 WiFi 信号强度随时间的变化情况

为避免信道拥挤, 需对无线路由器信道进行重新配置, 具体如图 3.18 所示, 将信道 1 修改为信道 9。保存设置, 重启路由器。

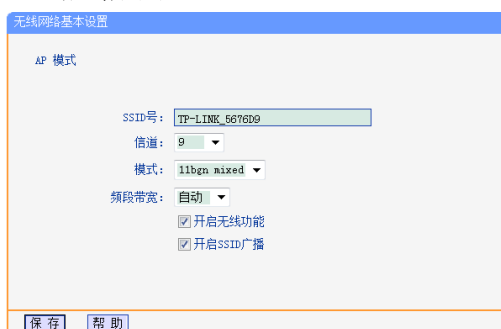


图 3.18 无线路由器信道修改

使用 inSSIDer 再次扫描并观察无线信道状态, 可看到, 实验配置的无线路由器已更改为信道 9, 且信道质量明显提升。

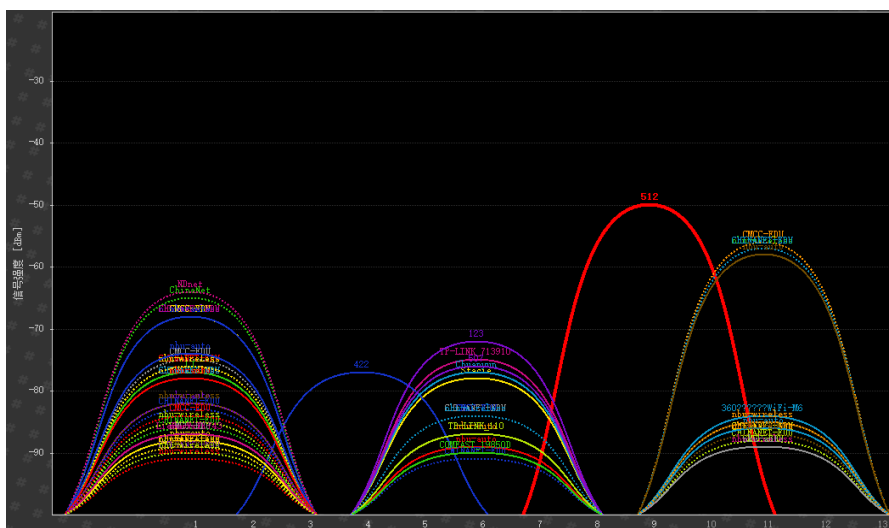


图 3.19 更改后的 WiFi 信道情况

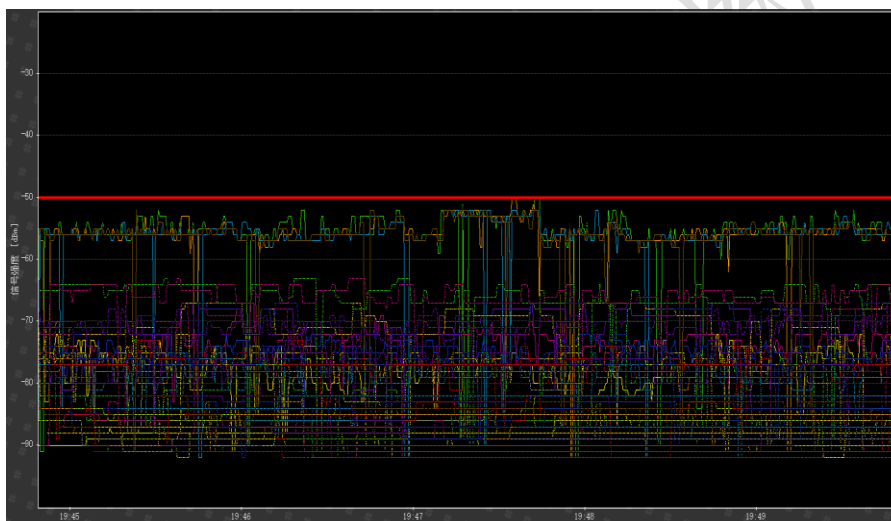


图 3.20 更改后的 WiFi 信号强度随时间变化情况

5 扩展实验步骤与演示

扩展实验内容为测量不同网络状态下的传输性能，即在基础实验中组建的无线局域网下，分别测量信道较为繁忙和较为空闲时的网络传输性能；实验要求 4 人共同完成，根据图 3.21 所示，4 人分别操作 4 台 PC。

5.1 信道较繁忙时网络传输性能

- (1) 根据 3.2 节准备好所有环境资源，4 台 PC 都需插上无线网卡并且安装 iPerf 软件；
- (2) 将 4 台 PC 都连接至 4.1 节中组建的无线局域网；
- (3) 根据第 4 节内容将无线路由器的信道设置为较为繁忙的信道，如信道 1；
- (4) 实验拓扑如图 3.21 所示，PC0 为服务器，PC1、PC2 和 PC3 为客户端。在 PC0 中，利用 iPerf 工具开启 3 个服务，命令执行格式为：**iperf -s -p PORT**。其中 PORT 可自行设置，建议为 5000 以后的端口号数值。在本实验中，端口号分别设置为 50626、50627 和 50628，具体如图 3.22、图 3.23 和图 3.24 所示。（需注意，在执行所有 iperf 命令时，需要在 iPerf.exe 所在的目录下，否则无法执行，具体请参考本实验第 7 节第(4)点。）

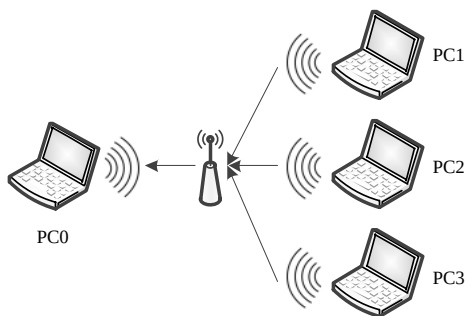


图 3.21 实验拓扑示意图

```
C:\Users\DK>cd C:\Users\DK\Desktop\iperf-2.0.0\bin
C:\Users\DK\Desktop\iperf-2.0.0\bin>iperf -s -p 50626
-----
Server listening on TCP port 50626
TCP window size: 8.00 KByte (default)
-----
```

图 3.22 监听端口 50626 的 iPerf 服务

```
C:\Users\DK>cd C:\Users\DK\Desktop\iperf-2.0.0\bin
C:\Users\DK\Desktop\iperf-2.0.0\bin>iperf -s -p 50627
-----
Server listening on TCP port 50627
TCP window size: 8.00 KByte (default)
-----
```

图 3.23 监听端口 50627 的 iPerf 服务

```
C:\Users\DK>cd C:\Users\DK\Desktop\iperf-2.0.0\bin
C:\Users\DK\Desktop\iperf-2.0.0\bin>iperf -s -p 50628
-----
Server listening on TCP port 50628
TCP window size: 8.00 KByte (default)
-----
```

图 3.24 监听端口 50628 的 iPerf 服务

(5) 在 PC0 开启 3 个服务后, 紧接着在 PC1、PC2 和 PC3 中的命令提示符中执行命令: `iperf -c IP_ADDRESS -p PORT -i 5 -t 20`。其中 IP_ADDRESS 为 PC0 的局域网 IP 地址, PORT 分别设置为步骤(4)中开启的 3 个服务对应的端口, -i 表示输出间隔, 5 为具体的间隔(秒), -t 为测量时长, 20 表示 20 秒, 执行结果如图 3.25 至图 3.28 所示。(需注意, 关于 IP_ADDRESS 即 PC0 局域网 IP 地址的说明, 请参考本实验第 7 节第(5)点。)

```
[300] local 192.168.1.104 port 54012 connected with 192.168.1.101 port 50626
[ ID] Interval      Transfer    Bandwidth
[300] 0.0- 5.0 sec  3.73 MBytes  6.27 Mbits/sec
[300] 5.0-10.0 sec  3.38 MBytes  5.66 Mbits/sec
[300] 10.0-15.0 sec  3.79 MBytes  6.36 Mbits/sec
[300] 15.0-20.0 sec  2.83 MBytes  4.74 Mbits/sec
[300] 0.0-20.0 sec  13.7 MBytes  5.75 Mbits/sec
```

图 3.25 PC1 向 PC0 发送数据包

```
[156] local 192.168.1.103 port 54141 connected with 192.168.1.101 port 50627
[ ID] Interval      Transfer    Bandwidth
[156] 0.0- 5.0 sec   720 KBytes  1.18 Mbits/sec
[156] 5.0-10.0 sec   616 KBytes  1.01 Mbits/sec
[156] 10.0-15.0 sec   720 KBytes  1.18 Mbits/sec
[156] 15.0-20.0 sec   608 KBytes  996 Kbits/sec
[156] 0.0-20.0 sec   2.61 MBytes 1.09 Mbits/sec
```

图 3.26 PC2 向 PC0 发送数据包

```
[280] local 192.168.1.110 port 49803 connected with 192.168.1.101 port 50628
[ ID] Interval      Transfer    Bandwidth
[280] 0.0- 5.0 sec   664 KBytes  1.09 Mbits/sec
[280] 5.0-10.0 sec   632 KBytes  1.04 Mbits/sec
[280] 10.0-15.0 sec   656 KBytes  1.07 Mbits/sec
[280] 15.0-20.0 sec   880 KBytes  1.44 Mbits/sec
[280] 0.0-20.0 sec   2.77 MBytes 1.16 Mbits/sec
```

图 3.27 PC3 向 PC0 发送数据包

```
[284] local 192.168.1.101 port 50626 connected with 192.168.1.104 port 54012
[ ID] Interval      Transfer    Bandwidth
[284] 0.0-20.0 sec  13.7 MBytes 5.76 Mbits/sec

[284] local 192.168.1.101 port 50627 connected with 192.168.1.103 port 54141
[ ID] Interval      Transfer    Bandwidth
[284] 0.0-20.0 sec   2.61 MBytes 1.09 Mbits/sec

[284] local 192.168.1.101 port 50628 connected with 192.168.1.110 port 49803
[ ID] Interval      Transfer    Bandwidth
[284] 0.0-20.0 sec   2.77 MBytes 1.16 Mbits/sec
```

图 3.28 PC0 收到 PC1-PC3 发送的数据包

5.2 信道较空闲时网络传输性能

- (1) 完成 5.1 节实验后, 将无线路由器的信道设置为较空闲的信道, 如信道 9;
- (2) 依次执行 5.1 节中的步骤(4)、(5), 运行结果如图 3.29 至图 3.32 所示。对比图 3.25 和图 3.28 可见, 空闲信道下数据传输率比繁忙信道下有所提高。

```
[276] local 192.168.1.104 port 53999 connected with 192.168.1.101 port 50626
[ ID] Interval      Transfer    Bandwidth
[276] 0.0- 5.0 sec   3.55 MBytes 5.96 Mbits/sec
[276] 5.0-10.0 sec   4.48 MBytes 7.52 Mbits/sec
[276] 10.0-15.0 sec   3.92 MBytes 6.58 Mbits/sec
[276] 15.0-20.0 sec   4.23 MBytes 7.10 Mbits/sec
[276] 0.0-20.1 sec   16.2 MBytes 6.76 Mbits/sec
```

图 3.29 PC1 向 PC0 发送数据

```
[156] local 192.168.1.103 port 54138 connected with 192.168.1.101 port 50627
[ ID] Interval      Transfer    Bandwidth
[156] 0.0- 5.0 sec   688 KBytes  1.13 Mbits/sec
[156] 5.0-10.0 sec   768 KBytes  1.26 Mbits/sec
[156] 10.0-15.0 sec   888 KBytes  1.45 Mbits/sec
[156] 15.0-20.0 sec   800 KBytes  1.31 Mbits/sec
[156] 0.0-20.1 sec   3.08 MBytes 1.28 Mbits/sec
```

图 3.30 PC2 向 PC0 发送数据

```
[280] local 192.168.1.110 port 49799 connected with 192.168.1.101 port 50628
[ ID] Interval      Transfer    Bandwidth
[280] 0.0- 5.0 sec   680 KBytes  1.11 Mbits/sec
[280] 5.0-10.0 sec   744 KBytes  1.22 Mbits/sec
[280] 10.0-15.0 sec   872 KBytes  1.43 Mbits/sec
[280] 15.0-20.0 sec   1.23 MBytes 2.06 Mbits/sec
[280] 0.0-20.0 sec   3.48 MBytes 1.46 Mbits/sec
```

图 3.31 PC3 向 PC0 发送数据

```
[268] local 192.168.1.101 port 50626 connected with 192.168.1.104 port 53999
[ ID] Interval      Transfer    Bandwidth
[268] 0.0-20.1 sec  16.2 MBytes 6.77 Mbits/sec
```

```
[268] local 192.168.1.101 port 50627 connected with 192.168.1.103 port 54138
[ ID] Interval      Transfer      Bandwidth
[268] 0.0-20.1 sec  3.08 MBytes   1.29 Mbits/sec

[268] local 192.168.1.101 port 50628 connected with 192.168.1.110 port 49799
[ ID] Interval      Transfer      Bandwidth
[268] 0.0-19.9 sec  3.48 MBytes   1.46 Mbits/sec
```

图 3.32 PC0 收到 PC1、PC2 和 PC3 发送的数据

6 结果分析与讨论

本实验先讨论了如何进行无线局域网的组网(包括基本配置和安全配置), 然后利用 inSSIDer 分析周围环境中的 WiFi 热点信号, 确定信道状态, 最后分别在不同信道状态下测量网络传输性能。

7 注意事项

实验过程中需要注意:

- (1) 信号强度值的单位为 dBm, 其绝对值越小意味着信号越强。一般地, WiFi 信号强度处于 -120dBm~-30dBm 范围, 而 >-50dBm 为较强信号, <-90dBm 很差, 甚至可能无法连接;
- (2) 实验所用主机必须要安装无线网卡。打开 inSSIDer 软件后一般会默认选择无线网卡, 若所选网卡错误将无法检测, 此时读者需要手动选择;
- (3) 时间位图能够反映信号强度的稳定性, 但存在一定随机性, 实验中需要多观察一段时间。
- (4) 执行所有 iperf 命令时, 需在 iPerf.exe 所在的目录下, 否则无法执行。可使用 cd 命令进入该目录, 具体如图 3.33 所示, 读者需将该路径换成自己安装 iperf 软件的路径。

```
C:\Users\DK>cd C:\Users\DK\Desktop\iperf-2.0.0\bin
C:\Users\DK\Desktop\iperf-2.0.0\bin>
```

图 3.33 进入 iPerf.exe 所在目录

- (5) 查看 Windows 系统的 PC 局域网 IP 地址的方法如下: 打开系统的“命令提示符”; 输入命令 ipconfig; 在结果中找到相应网卡的网络连接信息, 如图 3.34 所示即为 PC 在局域网中的 IP 地址。

```
管理员: 命令提示符
C:\Users\DK>ipconfig

Windows IP 配置

无线局域网适配器 无线网络连接 2:

    媒体状态 . . . . . : 媒体已断开
    连接特定的 DNS 后缀 . . . . . :

以太网适配器 本地连接 2:

    媒体状态 . . . . . : 媒体已断开
    连接特定的 DNS 后缀 . . . . . :

无线局域网适配器 无线网络连接:

    连接特定的 DNS 后缀 . . . . . : lan
    本地链接 IPv6 地址 . . . . . : fe80::f068:3869:41bf:94a9%14
    IPv4 地址 . . . . . : 192.168.1.224
    子网掩码 . . . . . : 255.255.255.0
    默认网关 . . . . . : 192.168.1.1
```

图 3.34 查看局域网 IP 地址

实验四 无线局域网信号测量

软件开发实验

1 实验要求与目的

- 1 人独立完成
- 理解和学习无线网络程序开发库
- 无线网络分析工具开发

2 实验背景与原理

本实验主要针对无线局域网进行软件开发, 理论知识内容见教材第3章, 信号测量采用 Windows 系统提供的 wlanapi 开发库, 数据显示与图形化采用 Java 内置表格类 JTable 和开源图表绘制类库 JFreeChart。

3 实验环境与资源

(1) 环境资源:

PC 一台、无线网卡一个、Winpcap 网络开发包、Visual Studio 2015 软件、IntelliJ IDEA 软件、Java 环境。

(2) 使用说明:

- 无线网卡如图 4.1 所示, 若读者实验所用 PC 无内置无线网卡, 则使用该无线网卡。第一次使用时需要安装驱动, 详见实验资源包 WNTLab/experiments/exp4。

- Winpcap 网络开发包: 作者使用版本为 4.1.2 版, 压缩包大小约 700KB, 下载方法请见实验资源包 WNTLab/experiments/exp4, 官网下载地址: <http://www.winpcap.org/devel.htm>。

- Visual Studio 2015 软件官网下载地址: <https://www.visualstudio.com>, 详细的下载方法及说明请见实验资源包 WNTLab/experiments/exp4。Visual Studio 2015 仅支持 Windows 7 SP1 及以上系统, 本实验以 Windows 10 系统中使用 VS 2015 为例。

- IntelliJ IDEA 是 Java 语言的集成开发工具, 详细的下载方法及说明请见实验资源包 WNTLab/experiments/exp4, 官网下载地址为: <http://www.jetbrains.com/idea/>。

- Java 环境主要是 JDK, 它是 Java 语言的软件开发工具包, 包含了 Java 的运行环境 JRE 和 Java 工具。作者使用版本为 9.0.1, 安装包大小约 370MB, 下载方法请见实验资源包 WNTLab/experiments/exp4, 官网下载地址为: <http://www.oracle.com/technetwork/cn/java/javase/downloads/index.html>。

- 本实验电子资源提供 VS 环境工程文件与 Java 源程序代码作为参考, 读者可自行导入工程, 完成环境配置后进行修改和编译。



图 4.1 无线网卡

4 实验步骤与讨论

4.1 开发平台的配置

由于 Windows 系统自带 wlanapi.dll 动态链接库, 所以 VS 2015 开发环境无需任何配置。

下面介绍 IDEA 的开发环境设置。将下载的 jfreechart 压缩包解压到指定目录, 如“C:\实验4\jfreechart”, 里面包含 jar 包、文档和示例程序等, 如图 4.2 所示。

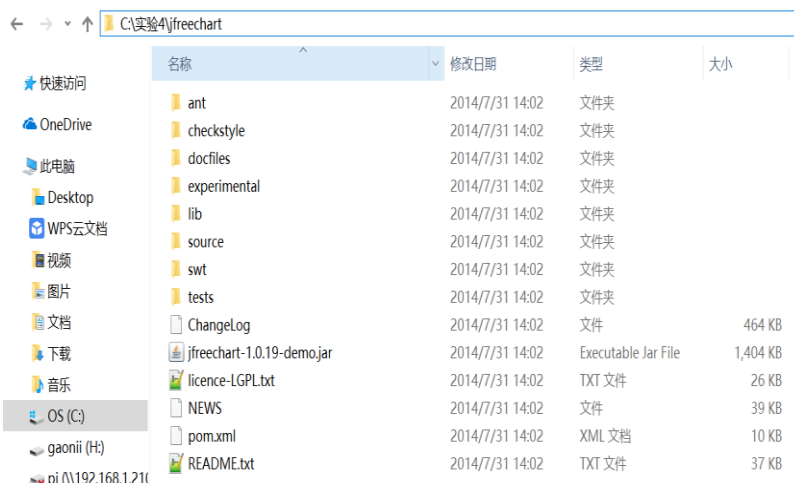


图 4.2 jfreechart 目录

获得 jfreechart 后, 在 IDEA 工程导入 JDK 和 jfreechart 所提供的 jar 包, 具体如下。

(1) 导入 JDK: 在 IDEA 工具栏中选择 File->Project Structure, 在左方 Project Settings 下选择 Project 进行配置, 并在右方 Project SDK 下选择 New...->JDK, 添加 JDK 本地安装路径, 如图 4.3 所示。

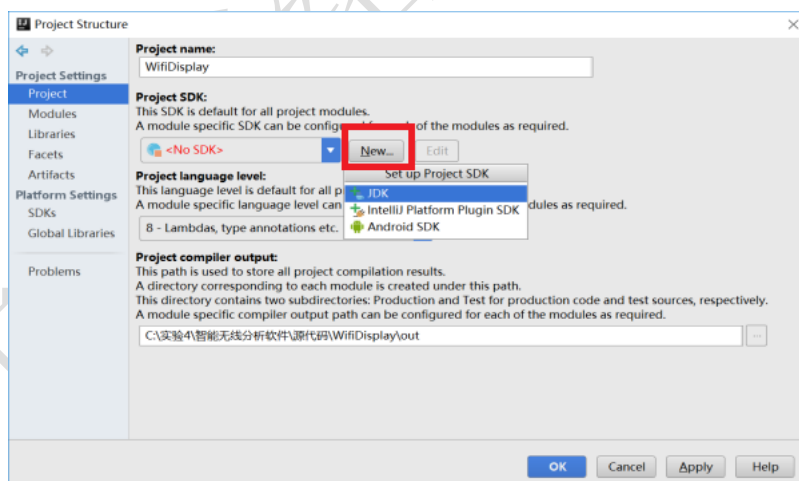


图 4.3 添加 JDK 路径

(2) 导入 jfreechart jar 包: 在 IDEA 中选择 File->Project Structure, 在左方 Project Settings (下选择 Modules 进行配置, 并在右方选择 Dependencies->“+”按钮->JARs or directories..., 添加 JFreeChart 所提供的 jfreechart-1.0.19-demo.jar 包, 如图 4.4 所示。

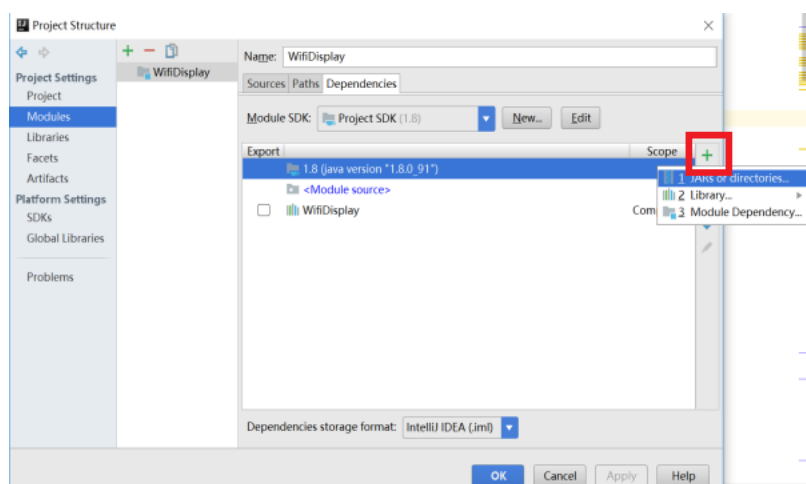


图 4.4 添加 jar 包

为验证配置是否成功, 可运行 WifiDisplay 工程中 MakeGraph.java 文件。若配置成功, IDEA 会生成数据显示界面。

4.2 数据采集模块源码分析

在 WifiScan 工程中, 程序通过 C++调用 wlanapi 提供的 WlanEnumInterfaces 函数获取当前网卡信息, 并通过当前网卡信息调用 WlanGetNetworkBssList 函数, 获取对应基本服务集 BSS 集合列表。遍历 BSS 集合列表, 获取 BSS 对应的信息, 如 MAC 地址、所在信道、SSID、RSSI、链路质量、BSS 类型等, 存入临时文件, 等待数据显示模块读取。

以输出 BSS 信息为例, 其代码如下所示, 其余详见 WifiScan.cpp 文件。

```
VOID PrintBssInfo(__in PWLAN_BSS_ENTRY pBss) {
    WCHAR strSsid[DOT11_SSID_MAX_LENGTH+1];
    UINT i;
    PBYTE pIe=NULL;
    if (pBss!=NULL) {
        // MAC address
        for (i=0;i<6;i++)
            wcout<<setw(2)<<setfill(L'0')
                <<hex<<(UINT)pBss->dot11Bssid[i]<<L" ";
        wcout<<" ";
        //SSID
        wcout<<SsidToStringW(strSsid,sizeof(strSsid)/sizeof(WCHAR),
            &pBss->dot11Ssid);
        wcout<<" ";
        //CHANNEL
        ULONG Channel = 0;
        if((pBss->ulChCenterFrequency)/1000<3000)
            Channel = ((pBss->ulChCenterFrequency)/1000-2412)/5+1;
        else
            Channel = ((pBss->ulChCenterFrequency)/1000-5745)/20+149;
        wcout<<dec<<Channel;
        wcout<<" ";
        //Rssi
        if (pBss->IRssi > (-34)) pBss->IRssi = -34;//avoid some invalid case
        wcout<< pBss->IRssi;
        wcout<<" ";
        //LinkQuality
        wcout<< pBss->uLinkQuality;
        wcout<<" ";
    }
}
```

```
switch (pBss->dot11BssType) {
    case dot11_BSS_type_infrastructure:wcout<<"infrastructure"; break;
    case dot11_BSS_type_independent:wcout<<"independent"; break;
    default:wcout<<"any";
}
wcout << ",";
}
}
```

4.3 数据显示模块源码分析

在 WifiDisplay 工程中，程序通过运行同一目录下的 WifiScan.exe，将实时获得的 BSS 数据存入临时文件，并通过 Java 读取临时文件中的 BSS 数据，使用 JTable 和 JFreeChart 进行数据显示。

该工程由三大类组成，分别为 ImportData、BSSDataUnit 与 MakeGraph，其功能模块图如 4.5 所示。

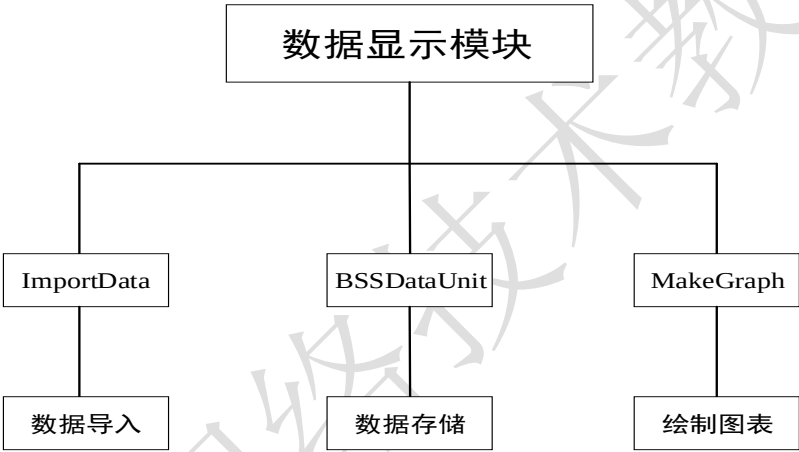


图 4.5 数据显示功能模块图

4.4 软件执行流程分析

软件执行流程如图 4.6 所示，具体实现可参考源码。数据采集模块由 C++实现，编译生成 WifiScan.exe 程序。数据显示模块由 Java 实现，编译生成 WifiDisplay.jar 程序。注意，两者须放在同一目录下。

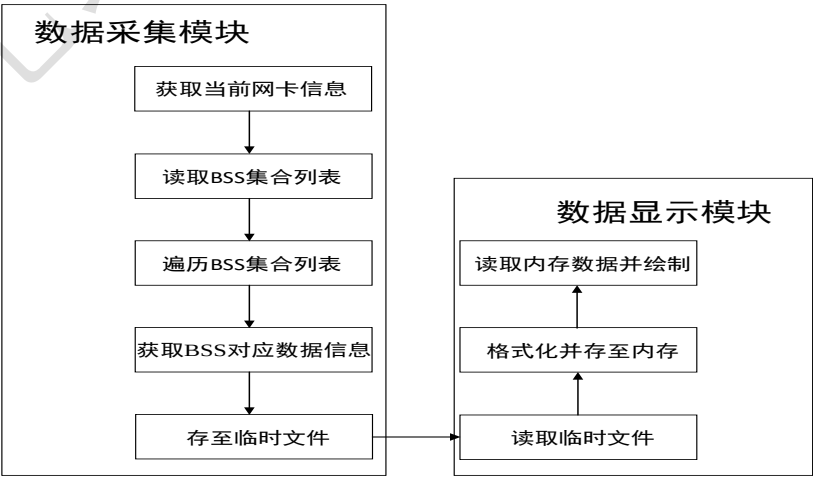


图 4.6 软件执行流程图

5 实验结果与分析

通过 Cmd 命令行进入软件所在目录，输入指令 `java -jar WifiDisplay.jar`，运行结果如图 4.7 所示，且每 1000ms 更新一次数据。

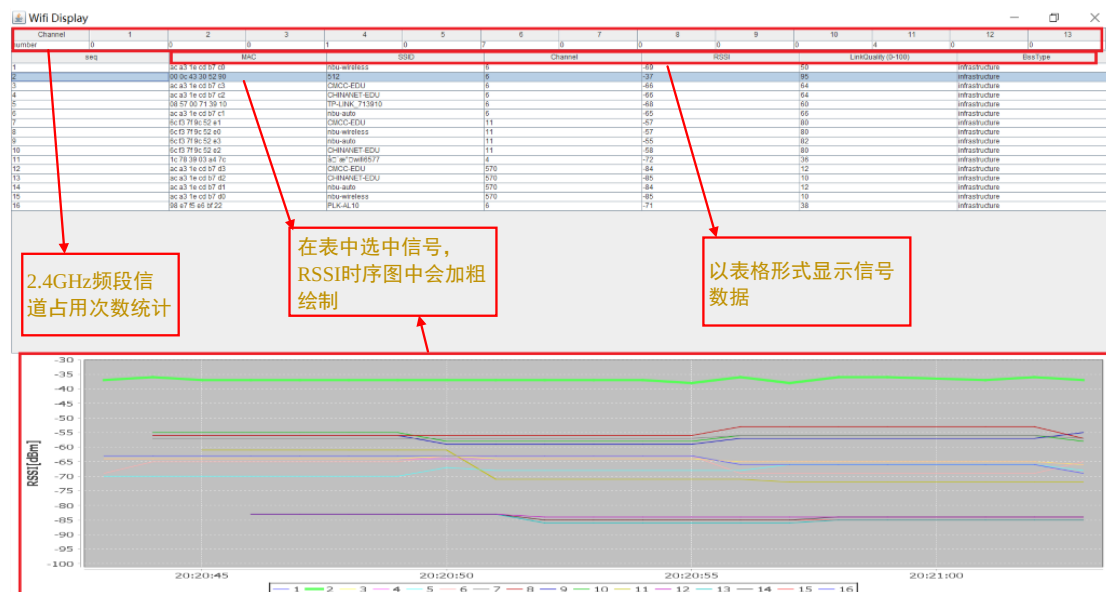


图 4.7 运行界面

6 扩展与分析

在掌握使用 wlanapi 开发库获取无线网络数据的基础上，尝试在 Windows 环境中使用 C/C++ 语言自行开发数据显示模块，功能与本实验 Java 环境下开发的 GUI 相同。

实验五 无线局域网数据分组分析

1 实验要求与目的

- 1人独立完成
- 熟悉常见的无线测量工具的特点和用途
- 学习 Windows 上的 Wireshark 数据包捕获和分析
- 学习 Linux 上的 Wireshark 和 Kismet 数据包捕获和分析

2 实验背景与原理

(1) WLAN 理论内容见教材第3章, WLAN 无线测量工具可分为3类: a.有线测量工具改进或衍生, 如 Wireshark 和 Mognet; b.无线环境专用测量工具, 如 Kismet、WiFiscanner 和 Wavemon; c.实验性无线测量工具, 如 Airtafr, Jigsaw 和 WisMon。表 4.1 列出了常见无线测量工具的性能参数。

表 5.1 无线测量工具简介

工具	输入方法/格式			输出格式					平台
	包捕获	Pcap	Linux 无线工具 API	字符界面	图形界面	文字	Pcap	数据库	
Airtafr			√	√					Linux
Easysnuffle	√			√	√				Linux
Jigsaw	√	√					√		
Kismet	√	√		√		√	√		Linux&Windows (受限)
Mognet	√						√		多平台 (多数用 Java 编写)
Wavemon		√		√					Linux
WiFiscanner	√			√			√		Linux
Wireshark	√	√			√	√	√		多平台
Wismon	√				√	√			Linux
Wit		√						√	多平台 (多数用 Perl 编写)

Linux 系统因其开放性而更通用, Linux 上的无线测量工具较少受到限制, 可获得较全面的无线网络测量信息, 而 Windows 平台会有较多限制(许多无线网络信息会被过滤)。

(2) Wireshark 相关介绍参见教材 3.5 节。

(3) Kismet 可以(和适当的硬件一起)嗅探 IEEE 802.11 的通信, 支持插件扩展, 能支持其他的通信介质。Kismet 通过被动获取数据包来识别网络, 可探测标准命名的网络和隐藏网络, 通过数据通信推测出未发射信标的网络。

3 实验环境与资源

3.1 基础实验

(1) 环境资源:

PC 一台、无线网卡一个、Wireshark 软件(Windows 版)。

(2) 使用说明:

●无线网卡如图 5.1 所示, 若读者实验所用 PC 无内置无线网卡, 则使用该无线网卡。第一次使用时需要安装驱动, 详见实验资源包 WNTLab/experiments/exp5。

●Wireshark 软件: 作者使用版本为 2.2.7, 安装包大小约 50MB, 下载方法请见实验资源包 WNTLab/experiments/exp5, 官网下载地址: <https://www.wireshark.org/download.html>。

3.2 扩展实验

(1) 环境资源:

PC 一台、无线网卡一个、wnt-contiki 虚拟机系统。

(2) 使用说明:

●wnt-contiki 虚拟机系统为实验一第 4.2 节搭建的环境, 已集成了本实验所需软件: Wireshark(Linux 版)。



图 5.1 无线网卡

4 基础实验步骤与演示

基础实验内容为在 Windows 系统的 PC 中使用 Wireshark 软件测量无线局域网; 实验要求 1 人独立完成。

4.1 实验准备

(1) 根据 3.1 节准备好所有的环境资源;

(2) 如果 PC 不自带无线网卡, 则将无线网卡插在 PC 的 USB 接口, 并在系统的网络连接中确认已有无线网卡, 如图 5.2 所示; 如果 PC 自带无线网卡, 请跳过该步骤;

(3) 点击系统任务栏右边区域的无线网络图标, 可打开无线网络连接程序, 列出了所有搜索到的无线网络, 找到对应无线网络进行连接, 即可建立与 AP 间的通信, 如图 5.3 所示。

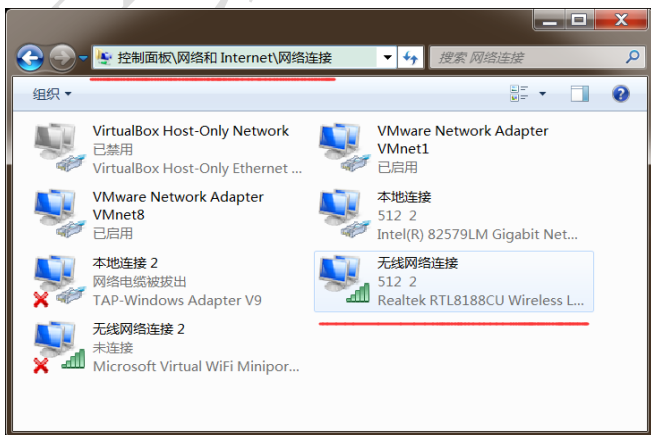


图 5.2 查看无线网络连接



图 5.3 连接无线局域网

4.2 Wireshark 用户界面介绍

Wireshark 提供了非常完善的图形可视化界面, 如图 5.4 所示。与大多数图形界面程序一样, Wireshark 主窗口由如下部分组成:

- **主菜单:** 包含各种设置和功能;
- **主工具栏:** 提供常用功能, 包括开始捕获和停止捕获等;
- **过滤工具栏 (Filter toolbar):** 自定义过滤方法过滤所有数据包;
- **捕获包 (Capture):** 提供常用捕获包的操作;
- **帮助 (Capture Help):** 提供软件的使用帮助;
- **文件/在线 (Files/Online):** 提供一些有用的链接操作;
- **状态栏:** 显示当前程序状态以及数据包的更多详情;

(需注意, 界面可能会因为软件版本的不同而略有差别, 请读者视情况使用。)

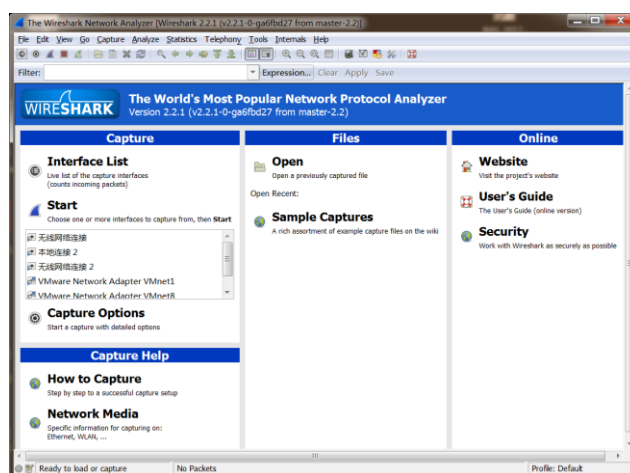


图 5.4 Wireshark 界面

Wireshark 捕获界面如图 5.5 所示, 该界面由 3 个面板组成, 各面板宽度都可调节:

- (1) **包列表(Packet List):** 显示打开文件的每个包的摘要, 包括源地址和目标地址、协议类型、数据包长度、数据包等。点击其中一条 (即一个数据包), 更多信息将显示在另外两个面板中。
- (2) **包细节(Packet details):** 显示 Packet list 面板中所选择包的所有信息, 信息分层显示。
- (3) **包字节(Packet bytes):** 显示 Packet list 面板选择的包的数据, 以及在 Packet details 面板中选中的字段。

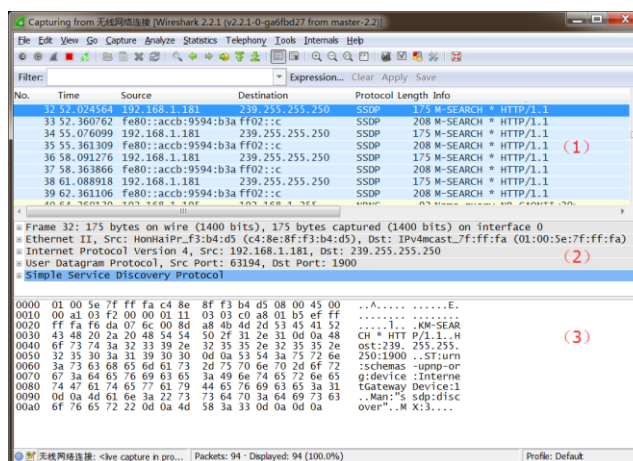


图 5.5 Wireshark 捕获界面

4.3 Wireshark 捕获数据包过程

- (1) 开始捕获前, 断开无线网络的连接, 如图 5.6 所示;
- (2) Wireshark 的 Capture 面板中包含了所有网络接口名称, 选中第 4.1.1 节中查看到的无线网卡, 如图 5.7 所示, 正确选择后, 点击 Start 开始捕获。



图 5.6 断开无线网络连接

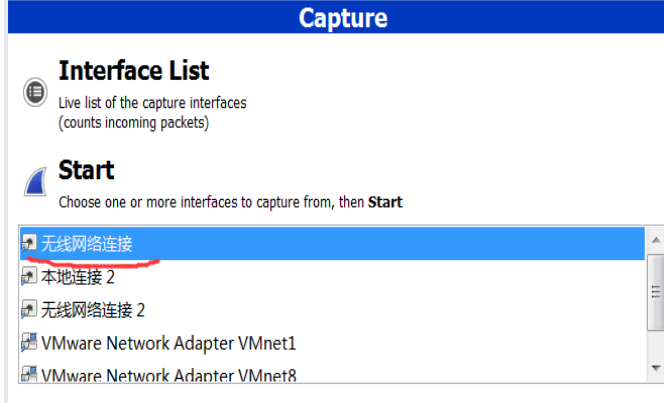


图 5.7 Wireshark 的 Capture 面板

- (3) 开始捕获后, 因为此时没有连接无线网络, 捕获界面将为空, 如图 5.8 所示;
- (4) 连接到无线网络, 可观察到已有数据包被成功捕获, 如图 5.9 所示。接下来打开浏览器, 输入 URL: <http://www.thinkmesh.net/wireless/>, 等页面显示完成, 然后停止捕获。

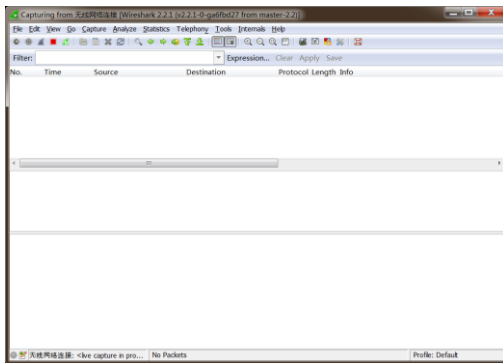


图 5.8 开始捕获图

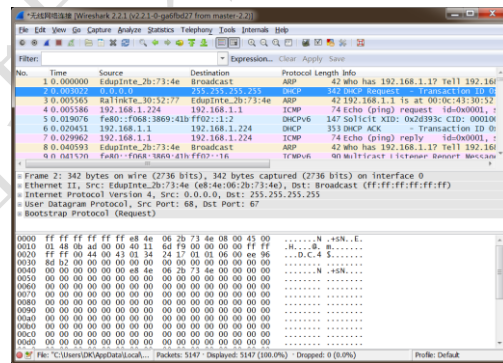


图 5.9 连接无线网络后捕获到数据包

4.4 结果分析

之前的一系列操作中, 主要包括以下几种数据包的交换:

EAPOL Key: 由于无线 AP 采用 WPA 方式进行加密, 所以需进行 WPA 密钥的验证, WPA 验证包括 4 次握手过程, 如图 5.10。

DHCP: 站点向 AP 获取 IP 地址, 包括 Discover/offer/Request 和 ACK4 个过程, 如图 5.11。

Time	Source	Destination	Protocol	Length	Info
1 0.00000000	Tp-LinkT_63:d3:b8	Tp-LinkT_0b:3a:b6	EAPOL	113	Key (Message 1 of 4)
2 0.00143800	Tp-LinkT_0b:3a:b6	Tp-LinkT_63:d3:b8	EAPOL	135	Key (Message 2 of 4)
3 0.01011000	Tp-LinkT_63:d3:b8	Tp-LinkT_0b:3a:b6	EAPOL	169	Key (Message 3 of 4)
4 0.01122100	Tp-LinkT_0b:3a:b6	Tp-LinkT_63:d3:b8	EAPOL	113	Key (Message 4 of 4)

图 5.10 EAPOL Key

10 14.1749550	0.0.0.0	255.255.255.255	DHCP	348	DHCP Discover - Transaction ID 0xea921da6
11 14.1853690	192.168.1.1	192.168.1.102	DHCP	590	DHCP Offer - Transaction ID 0xea921da6
12 14.1862560	0.0.0.0	255.255.255.255	DHCP	373	DHCP Request - Transaction ID 0xea921da6
13 14.2034810	192.168.1.1	192.168.1.102	DHCP	590	DHCP ACK - Transaction ID 0xea921da6

图 5.11 DHCP

DNS: 向 DNS 服务器获取域名对应的 IP 地址, 包括询问/应答两个过程, 如图 5.12。

TCP: 在本机和服务器之间建立可靠的网络连接, 为数据传输做好准备, 包括标准的 3

次握手, 如图 5.12。

HTTP: 网页内容通过 HTTP 协议传输, 其基本过程为 GET/Response, 如图 5.12。

436	21.8572270	192.168.1.102	192.168.1.1	DNS	76	Standard query 0x66d0 A ee1ab.nbu.edu.cn
437	21.8627850	192.168.1.1	192.168.1.102	DNS	234	Standard query response 0x66d0 A 210.33.16.22
438	21.8649230	192.168.1.102	210.33.16.22	TCP	62	pvunfwien > http [SYN] Seq=0 win=65535 Len=0 MSS=14
439	21.8697280	210.33.16.22	192.168.1.102	TCP	62	http > pvunfwien [SYN, ACK] Seq=0 Ack=1 win=65535 Len=0
440	21.8697630	192.168.1.102	210.33.16.22	TCP	54	pvunfwien > http [ACK] Seq=1 Ack=1 win=65535 Len=0
441	21.8704170	192.168.1.102	210.33.16.22	HTTP	379	GET /wireless/html/wifi/wiresharktest.html HTTP/1.1
442	21.8779300	210.33.16.22	192.168.1.102	HTTP	575	HTTP/1.1 200 OK (text/html)

图 5.12 DNS, TCP, HTTP

为便于数据包寻找, 可在过滤器中输入对应协议进行简单过滤。在包列表中只能看到简略信息, 更多信息可在包细节面板中查看。

但是, 所有数据包都和从有线网络中捕获到的数据包类似 (除了无线密钥验证), 很大程度上无法体现无线网络特征。一般 Windows 平台下只有使用特定硬件才能获取这些信息, 因此, 下面在 Linux 平台下获得那些被过滤掉的无线信息。

5 扩展实验步骤与演示

扩展实验内容为在 Linux 系统的虚拟机中分别使用 Wireshark、Kismet 软件测量无线局域网; 实验要求 1 人独立完成。

5.1 Linux 下无线测量工具 Wireshark

5.1.1 实验准备

- (1) 根据 3.2 节准备好所有的环境资源;
- (2) 使用 VMware 软件打开 wnt-contiki 虚拟机, 并在 VMware 软件的工具栏中点击“可移动设备”, 找到对应无线网卡, 点击“连接(断开与主机的连接)”, 即在虚拟机中使用无线网卡;

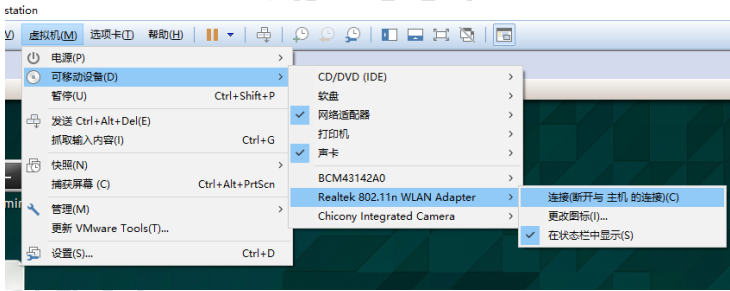


图 5.13 虚拟机中连接无线网卡

- (3) 查看虚拟机中的网络连接, 此时虚拟机可搜索到无线网络, 如图 5.14 所示, 选择相应的网络进行连接;

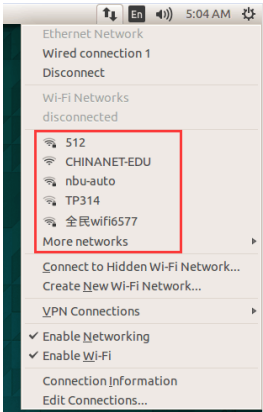


图 5.14 连接无线网络

- (4) 打开终端, 执行命令: `ifconfig`, 可看到本机有 3 个网络接口: `eth*`有线以太网、`lo`

本地环回、wlan*无线以太网, 如图 5.15 所示。(注意, *代表数字, 比如此处为 eth0、wlan3。)

```
wnt-contiki@wnt-contiki:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:0c:29:a6:2e:61
          inet6 addr: fe80::20c:29ff:fea6:2e61/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:185 errors:0 dropped:0 overruns:0 frame:0
          TX packets:554 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16145 (16.1 KB)  TX bytes:103989 (103.9 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:79 errors:0 dropped:0 overruns:0 frame:0
          TX packets:79 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:31555 (31.5 KB)  TX bytes:31555 (31.5 KB)

wlan3     Link encap:Ethernet  HWaddr e8:4e:06:2b:8a:e7
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

图 5.15 网络接口信息

5.1.2 实验过程

(1) 打开桌面 Wireshark 软件快捷方式, 在弹出窗中输入密码: 123456, 并确认, 如图 5.18 所示;

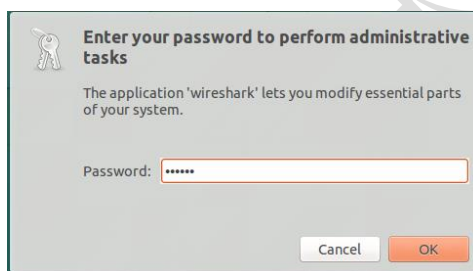


图 5.18 输入密码

(2) 输入密码并确认后, 进入 Wireshark 软件的主界面, 界面基本与第 4 节中介绍的 Windows 版一致, 如图 5.19 所示, 读者可参考 4.2 节介绍;

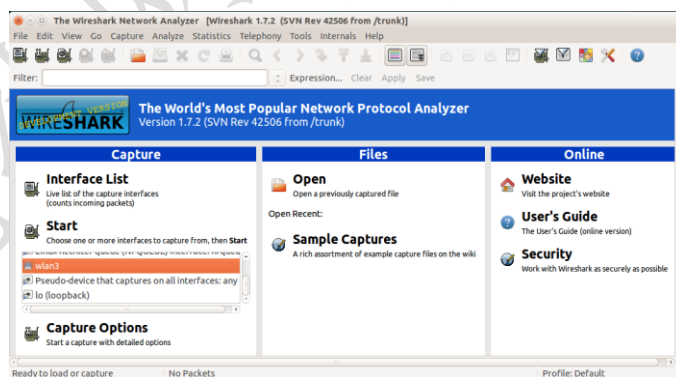


图 5.19 Linux 版 Wireshark 软件主界面

(3) 接下来捕获数据包, 先断开虚拟机的无线网络连接, 选择网卡接口为 wlan3, 具体步骤同 4.3 节, 此处不再赘述; 捕获完成后, 对数据包的分析可参考 4.4 节。

5.2 Linux 下无线测量工具 Kismet

Kismet 软件以字符界面为主, 如图 5.20。运行软件后, 选择默认 Yes, 按回车, 然后继续选择 Yes, 最后选择 start, 会提示找不到数据源。选择 Yes, 第一栏 Intf 设置为 mon0, 输入第二个 name, 然后 add, 如图 5.21 所示。

点击右上角 Close Console Windows, 可看到 Kismet 主界面, 如图 5.22 所示。

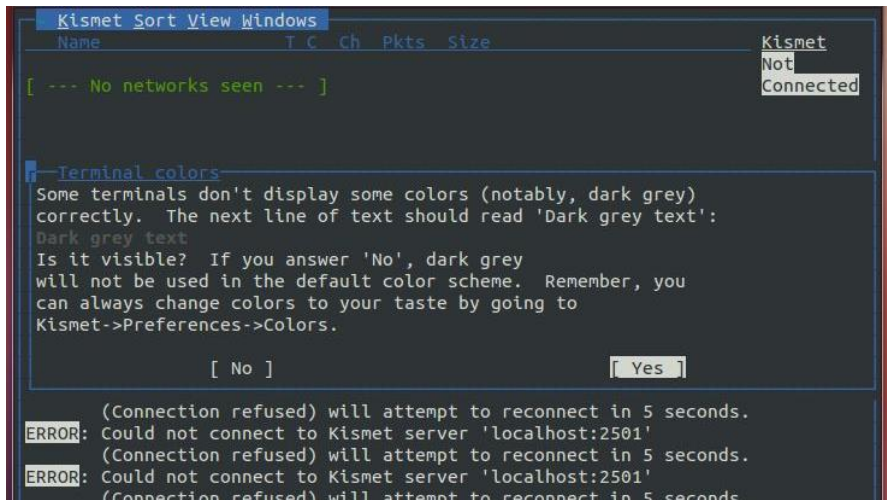


图 5.20 启动 Kismet

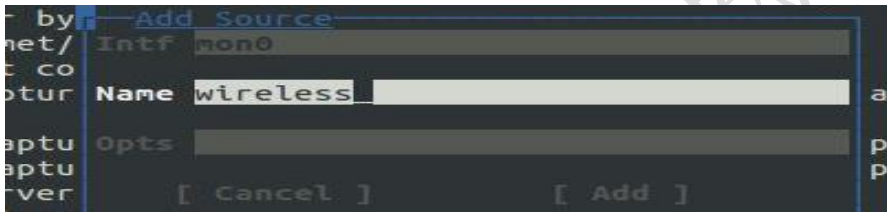


图 5.21 添加端口

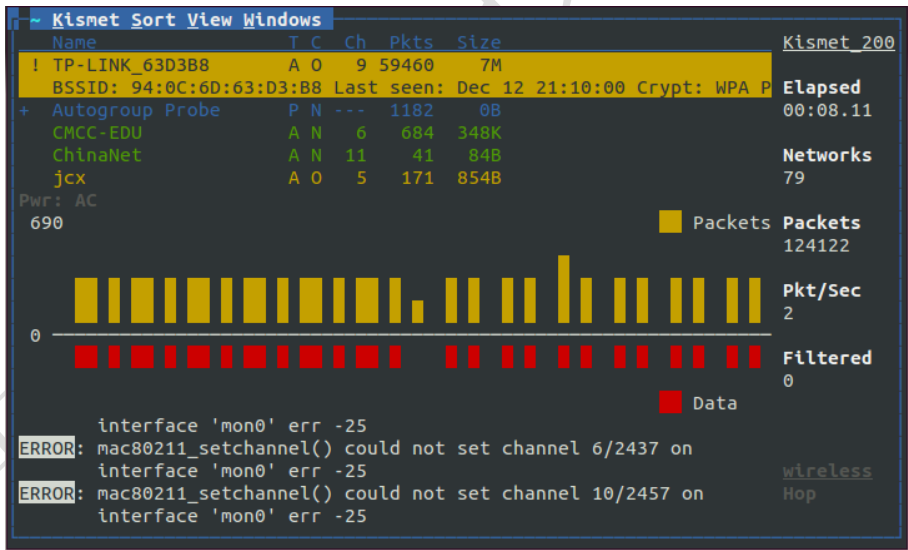


图 5.22 Kismet 主界面

主界面上面可以看到 4 个菜单：

- kismet: 包括软件的一些基本操作。
- Sort: 里面有各种 SSID 的排序规则, 默认自动排序。
- View: 选择需要显示的内容。
- Windows: 选择要进入的窗口。

下面可看到探测到的 SSID, 以及捕获包的数量和速度, 最下面是控制台信息。

双击其中一个 SSID, 可以看到该 SSID 的一些基本信息, 如信号强度/发包速度, 如图 5.23。如无可在 View 里更改, clients 可看到连接到该 SSID 的客户端网卡地址, 如图 5.24。

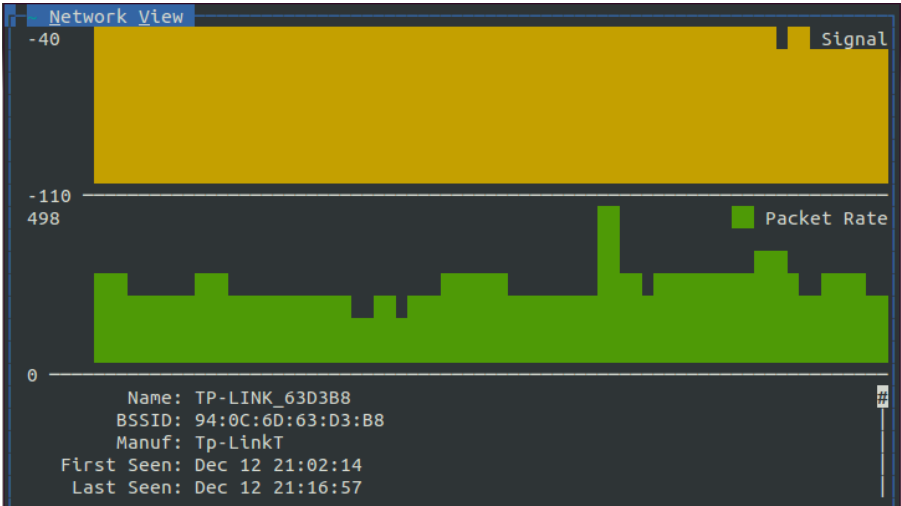


图 5.23 AP 信息

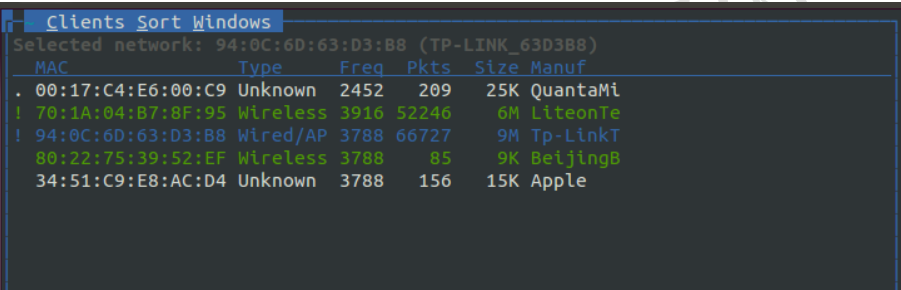


图 5.24 连接到 AP 的站点信息

6 结果分析与讨论

本实验分别在 Windows 和 Linux 系统下使用 Wireshark, 捕获无线局域网数据包进行分析, 并介绍了在 Linux 系统下 Kismet 软件的使用。实验结果以数据和图形方式显示, 读者可自行进行分析。

7 注意事项

实验过程中, 需要注意实验的步骤, 同时保证实验中的提示都正确, 在遇到问题时, 可去软件官方网站查找资料解决。

实验六 无线局域网数据分组分析

软件开发实验

1 实验要求与目的

- 1 人独立完成
- 理解和学习网络程序开发包
- 无线网络测量工具开发

2 实验背景与原理

本实验主要为无线局域网的软件开发, WLAN 知识内容见教材第 3 章, 网络测量使用 Winpcap 网络开发包, 与 Linux 下的 Libpcap 开发包相似, 两者函数基本通用, 可方便地将基于 Libpcap 的程序移植到 Windows 平台下。

3 实验环境与资源

(1) 环境资源:

PC 一台、无线网卡一个、Winpcap 网络开发包、Visual Studio 2013 软件。

(2) 使用说明:

- 无线网卡如图 6.1 所示, 若读者实验所用 PC 无内置无线网卡, 则使用该无线网卡。第一次使用时需要安装驱动, 详见实验资源包 WNTLab/experiments/exp6。

- Winpcap 网络开发包: 作者使用版本为 4.1.2 版, 压缩包大小约 700KB, 下载方法请见实验资源包 WNTLab/experiments/exp6, 官网下载地址: <http://www.winpcap.org/devel.htm>。

- Visual Studio 2013 软件官网下载地址: <https://www.visualstudio.com>, 详细下载方法及说明请见实验资源包 WNTLab/experiments/exp6。



图 6.1 无线网卡

4 实验步骤与讨论

4.1 开发平台的配置

下面介绍 Visual Studio 2013 的开发环境设置。将下载的 WinPcap 开发包解压到指定目录, 如“C:\WpdPack”, 里面包含 Lib, Include, 文档和示例程序, 如图 6.2 所示。

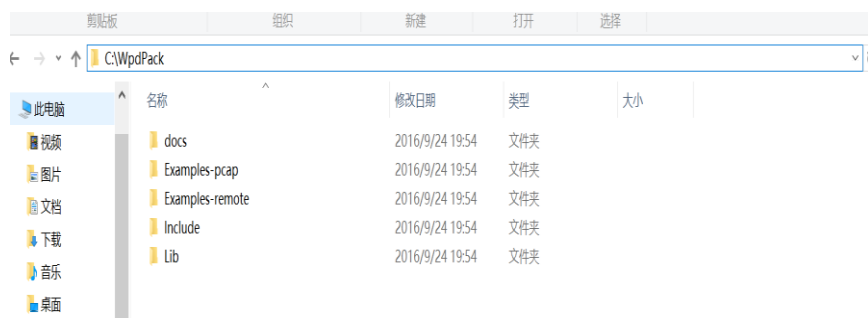


图 6.2 WinPcap 开发包目录

获得 WinPcap 开发包后, 接下来在项目中添加 WinPcap 的文件, 具体如下。

(1) 添加 include 文件: 选择项目->属性->配置属性->VC++目录, 在显示以下内容的目录中选择包含文件, 然后添加 WinPcap 开发包中的 Include 目录, 如图 6.3 所示。

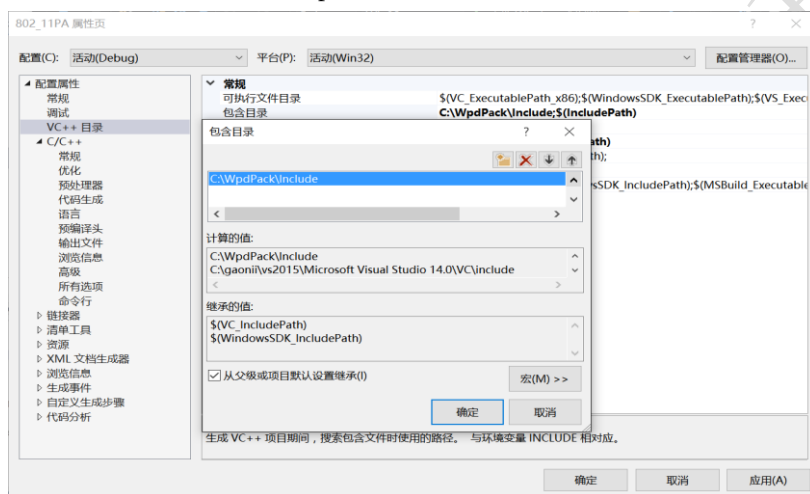


图 6.3 添加 Include 目录

(2) 添加库文件: 在上一个步骤的基础上, 选择“库文件”, 然后添加 WinPcap 开发包中的 Lib 目录, 如图 6.4 所示。

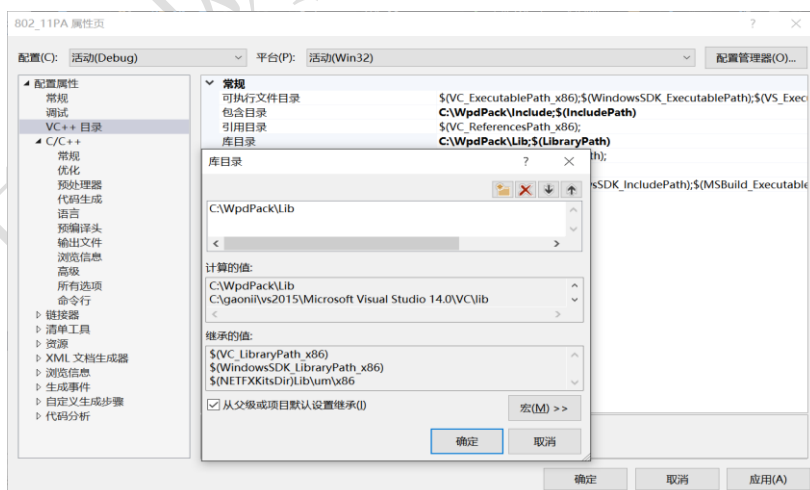


图 6.4 添加 Lib 目录

(3) 添加预处理定义 (项目创建后设置): 项目->属性->配置属性->C/C++->预处理器, 在预处理定义中添加 WPCAP 和 HAVE_REMOTE, 如图 6.5 所示。

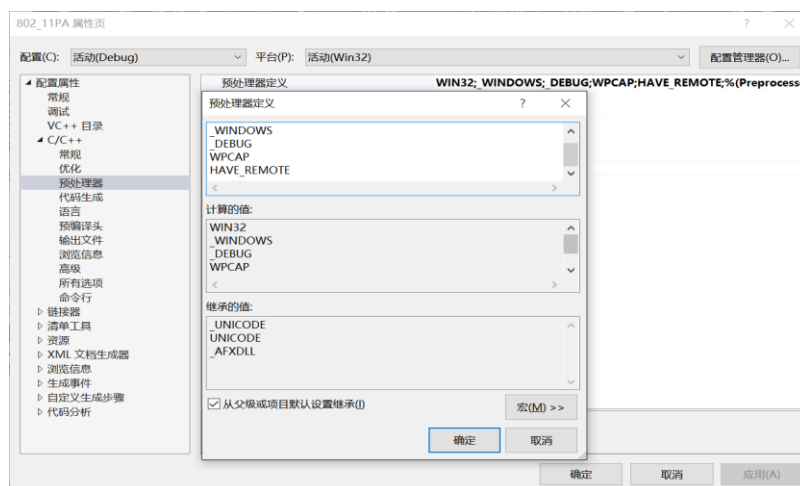


图 6.5 添加预处理器定义

(4) 添加 wpcap.lib 库文件：项目->项目属性->配置属性->链接器->命令行，在附加选项中加入 wpcap.lib，如图 6.6 所示。

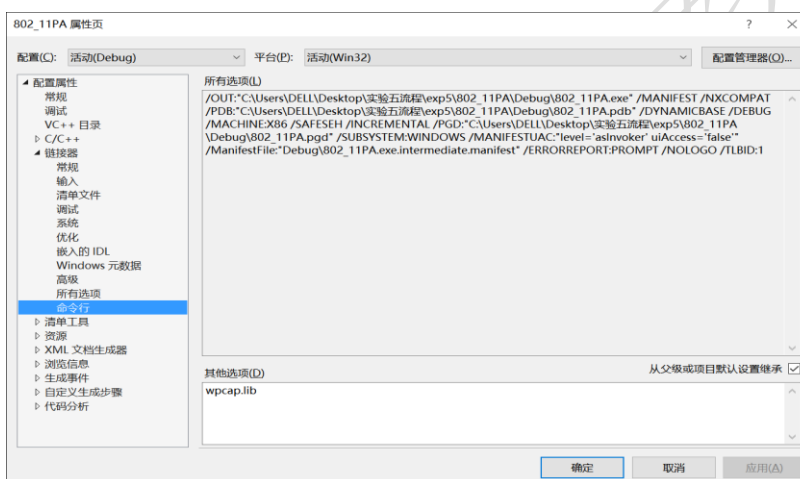


图 6.6 添加 wpcap.lib 库文件

(5) 在使用 WinPcap API 的源文件中添加 `#include "pcap.h"`。

为了验证配置是否成功，可运行 WinPcap 开发包中的示例程序，例如 Examples-pcap 文件夹中的 UDPdump。

4.2 程序界面设计

新建 MFC 工程，点击文件->新建->项目，在 Visual C++ 中选择 MFC->MFC 应用程序，输入项目名称，点击确定，进入 MFC 应用程序向导对话框。这里选择基于对话框，其他保持默认，点击完成即可。

添加程序界面中包含的各个控件，程序主界面如图 6.7 所示。



图 6.7 程序主界面

添加控件、修改参数方法：在解决方案资源管理器视图中，打开资源文件目录中的 rc 文件，如图 6.8。接着打开 rc 文件目录中的 Dialog 文件夹中的对话框文件，如图 6.9。打开对话框界面左边的工具栏，即可拖拽控件放置在对话框内，如图 6.10。控件参数只要右键点击对话框中的控件，打开属性栏就可修改。

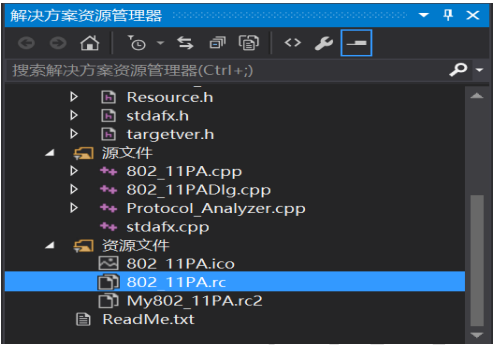


图 6.8 打开 rc 文件

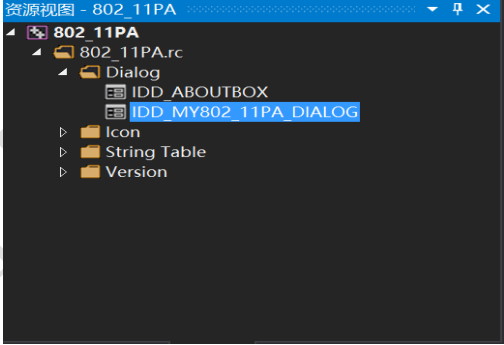


图 6.9 打开 DIALOG 文件

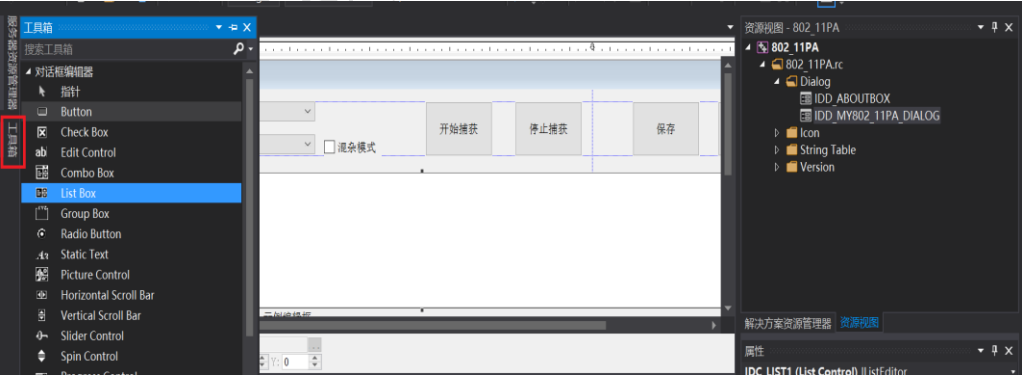


图 6.10 打开工具箱拖拽控件

下拉框：采用“Combo Box”，设置“外观-Type”栏为“DropList”。

包捕获列表：采用“List Control”，添加一个适当大小的列表框控件，设置“外观-View”项为“Report”，“外观-single selection”为“true”。

包细节：采用“Tree Control”，设置“Full Row Select”、“Has Buttons”、“Info Tip”、“Lines At Root”均为“True”。

包字节：采用“Edit Control”，设置“Multiline”、“Readonly”、“Auto Vscroll”、“Horizon Scroll”、“Vertical Scroll”均为“True”。

其他控件设置比较简单, 不再一一解释。

4.3 协议头部结构定义

新建头文件, 文件名为 Protocol_Define.h, 编写各类协议头部结构, 如 MAC 帧的头部:

```
typedef struct ethhdr {
    u_char dest[6];           //6 个字节目标地址
    u_char src[6];           //6 个字节源地址
    u_short type;             //2 个字节类型
};
```

其他各类协议定义可参考示例中的 Protocol_Define.h 文件。

4.4 数据包处理

不同协议的数据包处理采用自底向上的处理方式, 其流程如图 6.11 所示。

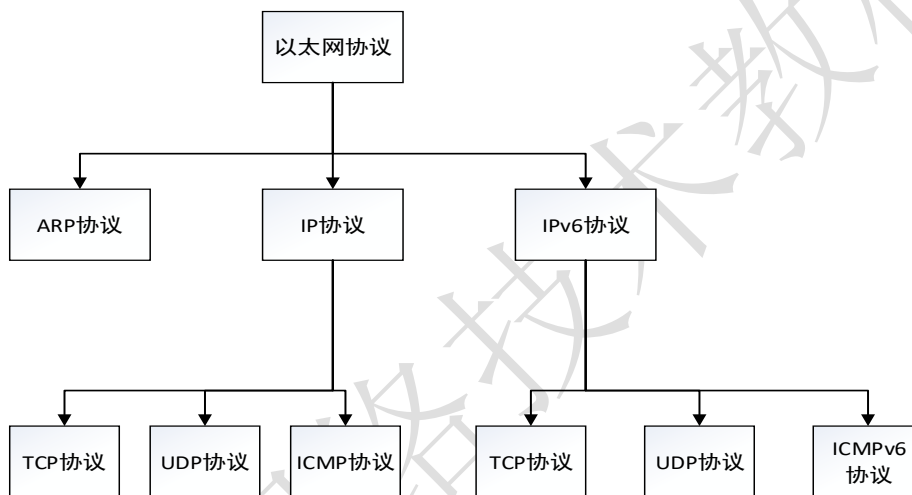


图 6.11 数据包处理流程

以 MAC 层为例, 其代码如下所示, 其余各层详见 Protocol_Analyzer.cpp 文件:

```
int analyze_frame(const u_char *pkt, struct datapkt *data, struct pktcount *npacket){
    int i;
    struct ethhdr *ethh = (struct ethhdr*)pkt;
    data->ethh = (struct ethhdr*)malloc(sizeof(struct ethhdr));
    if(NULL == data->ethh)
        return -1;
    for(i=0; i<6; i++){
        data->ethh->dest[i] = ethh->dest[i];
        data->ethh->src[i] = ethh->src[i];
    }
    npacket->n_sum++;
    /*由于网络字节顺序原因, 需要对*/
    data->ethh->type = ntohs(ethh->type);
    //处理 ARP 还是 IP 包?
    switch(data->ethh->type){
        case 0x0806:
            return analyze_arp((u_char*)pkt+14, data, npacket); //mac 头大小为
            break;
        case 0x0800:
            return analyze_ip((u_char*)pkt+14, data, npacket);
            break;
    }
}
```

```
case 0x86dd:
    return analyze_ip6((u_char*)pkt+14,data,npacket);
    return -1;
    break;
default:
    npacket->n_other++;
    return -1;
    break;
}
return 1;
}
```

4.5 主程序

主程序执行流程如图 6.12 所示, 具体实现可参考 802.11PADlg.cpp。注意, 响应列表操作的两个事件——包选中操作 (OnLvnItemchangedList1) 和包着色器 (OnNMCustomdrawList1), 需要在对应的列表控件中添加事件才可正常执行。添加方法: 右击列表控件, 选择添加事件处理程序, 即可添加所需事件, 如图 6.13 所示。

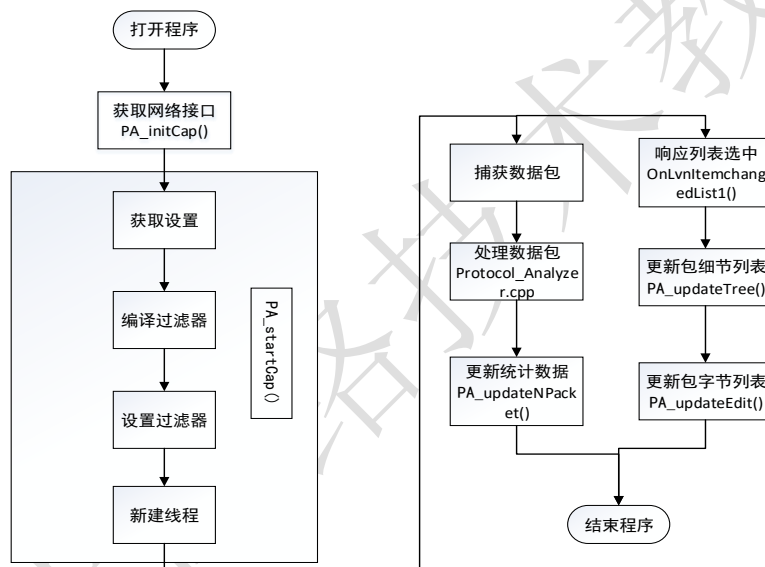


图 6.12 主程序执行流程

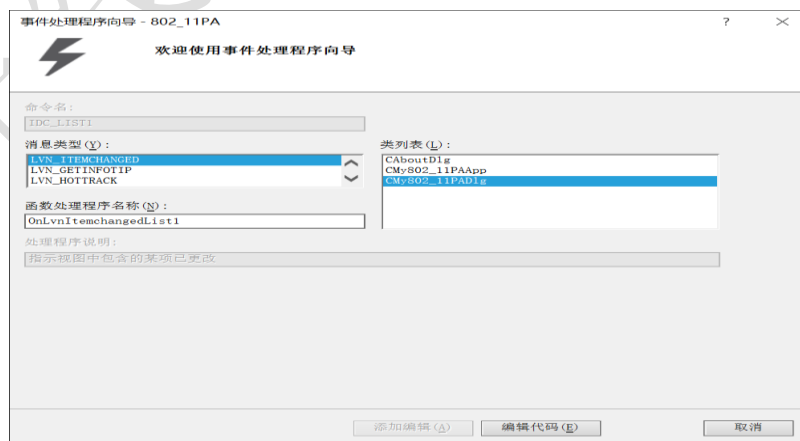


图 6.13 添加事件

5 实验结果与分析

程序在 Visual Studio 2013 编译执行之后, 参考执行结果如图 6.14 所示:

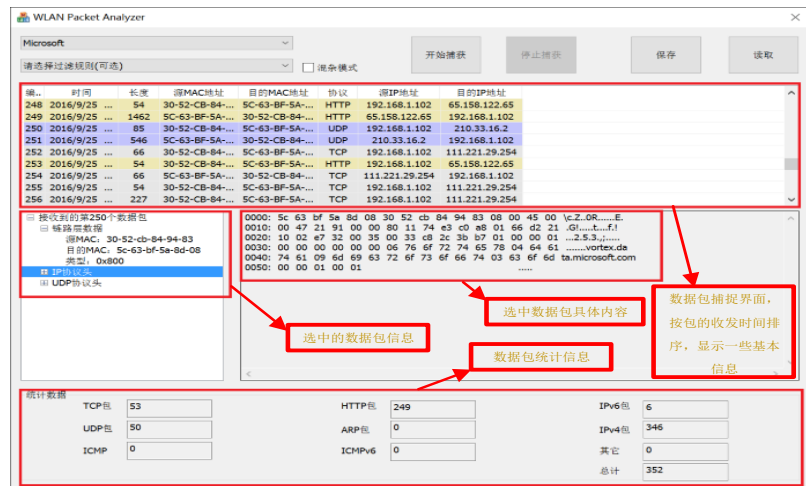


图 6.14 执行结果界面及注释

6 扩展与分析

在掌握 Windows 环境开发 WiFi 数据包捕获和分析的基础上, 尝试在 Linux 环境中使用 C/C++语言自行开发 WiFi 分析测量软件, 功能与本实验软件相同。

实验七 隐藏节点和暴露节点仿真实验

1 实验要求与目的

- 1人独立完成
- 学习无线网络中隐藏和暴露节点问题
- 利用 NS2 分析隐藏和暴露节点问题
- 了解 OTcl 脚本语言的基础知识

2 实验背景与原理

2.1 实验背景

隐藏节点指处于接收节点的覆盖范围内而在发送节点的覆盖范围外的节点。由于监听不到发送节点的数据传输, 隐藏节点不受限制地发送分组到该接收节点, 导致分组在接收节点处冲突。隐藏节点可以分为隐发送节点和隐接收节点。

暴露节点是指在发送节点覆盖范围内而在接收节点覆盖范围外的节点, 暴露节点因监听到发送节点的发送而可能延迟发送。但它其实是在接收节点的通信范围之外, 它的发送不会造成冲突, 这种延迟发送并不必要。

2.2 实验原理

隐藏节点仿真拓扑如图 7.1 所示, 节点 n_1 和 n_2 同时想发送数据给中间节点 n_0 , 但 n_1 和 n_2 都不在彼此传输范围内。所以当 n_1 发送数据给 n_0 时, n_2 并未检测到 n_1 正在传输, 会认为目前网络中无数据传输, 会将数据传输给 n_0 。即 n_1 和 n_2 同时将数据传输给 n_0 , 使得数据在 n_0 处产生冲突, 导致数据不可用。这种因传送距离而发生误判的问题称为隐藏节点问题。

为解决隐藏节点问题, 可使用教材 3.8 节介绍的 RTS/CTS 控制信息来避免冲突。发送方发出数据前, 先送出 1 个 RTS 包, 告知传输范围内的所有节点不要有任何发送操作。如果接收方目前空闲, 则响应 1 个 CTS 包, 告诉发送方可开始发送数据, 此 CTS 包也会告知在接收方信号传输范围内的所有其它节点不要进行任何传输操作。详见教材 3.8 节。



图 7.1 隐藏节点问题仿真拓扑

隐藏节点仿真拓扑如图 7.2 所示, 暴露节点问题和隐藏节点问题不同, 当一个节点要发送数据给另一节点时, 因邻节点也正在发送数据, 影响了原来节点的数据传送。参考教材 3.8 节, 4 个节点 S_1 、 S_2 、 R_1 、 R_2 , 其中 R_1 、 R_2 均不在对方传输范围内, 而 S_1 、 S_2 均在彼此传送范围内。因此, 当 S_1 正发送数据给 R_1 时, S_2 却不能将数据发送给 R_2 , 因为 S_2 会检测到 S_1 正发送数据, 如果其也发送数据的话, 就会影响 S_1 的数据发送。事实上, S_2 可以将数据发给 R_2 的, 因为 R_2 并不在 S_1 的传送距离内。为缓解暴露节点问题, 同样可采用 RTS/CTS 机制。当某个节点侦听到邻节点发送的 RTS, 但未监听到对应 CTS 时, 可判定本身是一个暴露节点, 所以允许传送数据到其它邻节点。当 S_2 收到 S_1 发出的 RTS, 但未收到对应 CTS 时, S_2 可推测自己是暴露节点, 则可同时发送数据。更多知识请参考教材 3.8 节。



图 7.2 暴露节点问题仿真拓扑

3 实验环境与资源

3.1 基础实验

(1) 环境资源:

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;
- 本实验主要用到仿真实验平台中的 NS2 软件, 官方 NS2 版本中未包含本实验所需 mUDP、mUdpSink 模块, 实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;
- 实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp7/exp7_basic。

3.2 扩展实验

(1) 环境资源:

同 3.1 节。

(2) 使用说明:

- 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp7/exp7_extend。

4 基础实验步骤与演示

基础实验内容为对隐藏节点、仿真节点的基本问题进行仿真; 实验要求 1 人独立完成。

4.1 隐藏节点问题仿真

本实验步骤如下:

- (1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 7.3 所示。



图 7.3 连接树莓派套件

- (2) “home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本。本手册各不同实验在不同子文件夹中, 实验七的子文件夹为“exp7”。

(2) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

```
cd WNTLab/experiments/exp7/exp7_basic/
```

再输入以下命令运行本实验, 如图 7.4 所示。

```
ns Hidden_Terminal.tcl
```

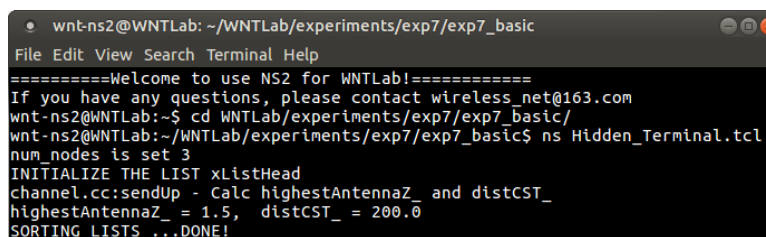


图 7.4 运行仿真实验

(3) 上述命令执行完成后会弹出 NAM 动画窗口, 利用 NAM 动画观察实验结果, 如图 7.5 所示。观看 NAM 动画时, 运行的步长调整为 2ms 即可。

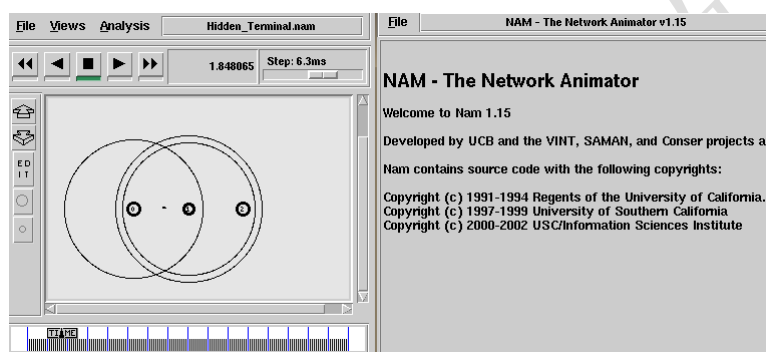


图 7.5 实验结果演示

仿真程序运行说明:

(1) 仿真开始时(0~1s), 节点间相互广播路由(DSDV)报文, 建立路由表。从 1.5s 开始, 节点 0 开始向节点 1 发送数据, 需要注意, 此时节点 0 需要先发送 RTS(实验中设置 802.11 的 $RTSThreshold=0$, 满足数据包大于 $RTSThreshold$, 表示开启 RTS/CTS, 解决隐藏终端问题; 实现隐藏终端请设置 $RTSThreshold=3000$, 满足数据包小于 $RTSThreshold$, 因为数据包不会大于 3000), 以保证信道(0→1)的顺利占用。

(2) 在 2s 时, 节点 2 也开始向节点 1 发送数据, 同样, 节点 2 也发送 RTS 报文, 此时可得知节点 1 的信道不空闲, 因此节点 2 开始退避, 等待一段时间重新尝试发送。

(3) 仿真过程(2~15s)中, 可看到节点 2 和节点 0 交替向节点 1 发送数据, 而非同时传输。仿真时间到达 15s 时, 两条流传输同时结束, 此后无数据传输, 偶尔有链路保活报文。

(4) 在 20s 时, 仿真实验结束。

4.2 暴露节点问题仿真

本实验步骤如下:

(1) 打开 Terminal (快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在目录:

```
cd WNTLab/experiments/exp7/exp7_basic
```

再输入以下命令运行本实验, 如图 7.6 所示;

```
ns Exposed_Terminal.tcl
```

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp7/exp7_basic
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp7/exp7_basic/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp7/exp7_basic$ ns Exposed_Terminal.tcl
num_nodes is set 4
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 120.0
SORTING LISTS ...DONE!
```

图 7.6 运行仿真实验

(2) 利用 NAM 动画观察实验结果, 如图 7.7 所示。在观看 NAM 动画时, 运行的步长调整为 2ms 即可。

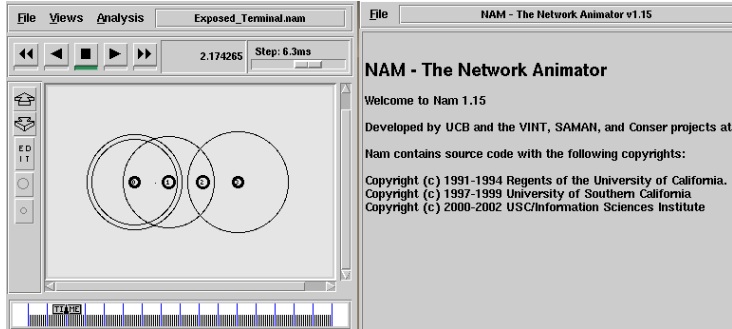


图 7.7 实验结果演示

仿真程序运行说明:

- (1) 仿真开始时(0~1s), 节点间相互广播路由(DSDV)报文, 建立路由表。
- (2) 从 1.5s 开始, 节点 1 开始向节点 0 发送数据, 需注意, 此时节点 1 需要先发送 RTS(实验中设置 802.11 的 RTSThreshold=0, 满足数据包大于 RTSThreshold, 表示开启 RTS/CTS, 解决暴露终端问题; 实现暴露终端请设置 RTSThreshold=3000, 满足数据包小于 RTSThreshold, 因为数据包不会大于 3000), 以保证信道(1→0)的顺利占用。
- (3) 在 2s 时, 节点 2 也开始向节点 3 发送数据, 同样, 节点 2 也发送 RTS 报文, 以获得信道(2→3)的顺利占用, 并正确传输数据。
- (4) 仿真时间到达 15s 时, 两条流传输同时结束, 此后无数据传输, 偶尔有链路保活报文。
- (5) 在 20s 时, 仿真实验结束。

5 扩展实验步骤与演示

为便于读者理解, 下面对基础实验中的代码进行简单分析, 并根据分析分别对隐藏节点和暴露节点仿真实验进行修改, 最后请读者根据修改案例自行完成 5.3 节中的作业。

5.1 隐藏节点仿真扩展

5.1.1 隐藏节点代码解释

分析代码前, 读者需要了解 OTcl 脚本语言(见教材第 2 章)。下面给出隐藏节点基础仿真实验的具体代码, 并且给出代码结构分析, 具体代码细节及含义请见电子资源的 exp7_basic 文件夹中的代码文件(Hidden_Terminal.tcl)。

#仿真模块参数设置

Mac/802_11 set RTSThreshold_	0.....	1
Antenna/OmniAntenna set X_	0.....	2
Antenna/OmniAntenna set Y_	0.....	3
Antenna/OmniAntenna set Z_	1.5.....	4
Antenna/OmniAntenna set Gt_	1.0.....	5
Antenna/OmniAntenna set Gr_	1.0.....	6
Phy/WirelessPhy set CPTresh_	10.0.....	7

Phy/WirelessPhy set CStresh_	8.91754e-10.....	8
Phy/WirelessPhy set RXThresh_	2.81838e-9.....	9
Phy/WirelessPhy set bandwidth_	2e6.....	10
Phy/WirelessPhy set Pt_	0.281838.....	11
Phy/WirelessPhy set freq_	9.14e+6.....	12
Phy/WirelessPhy set L_	1.0.....	13
#仿真变量设置		
set val(chan)	Channel/WirelessChannel.....	14
set val(prop)	Propagation/TwoRayGround.....	15
set val(netif)	Phy/WirelessPhy.....	16
set val(mac)	Mac/802_11.....	17
set val(ifq)	Queue/DropTail/PriQueue.....	18
set val(ll)	LL.....	19
set val(ant)	Antenna/OmniAntenna.....	20
set val(ifqlen)	100.....	21
set val(rp)	DSDV.....	22
#ns实例化和trace文件设置		
set ns [new Simulator]		23
set f [open Hidden_Terminal.tr w]		24
\$ns trace-all \$f		25
\$ns eventtrace-all		26
set nf [open Hidden_Terminal.nam w]		27
\$ns namtrace-all-wireless \$nf 500 500		28
#仿真拓扑设置		
set topo [new Topography]		29
\$topo load_flatgrid 500 500		30
create-god 3		31
#节点配置		
set chan [new \$val(chan)]		32
\$ns node-config -adhocRouting \$val(rp) \		33
-llType \$val(ll) \		34
-macType \$val(mac) \		35
-ifqType \$val(ifq) \		36
-ifqLen \$val(ifqlen) \		37
-antType \$val(ant) \		38
-propType \$val(prop) \		39
-phyType \$val(netif) \		40
-channel \$chan \		41
-topoInstance \$topo \		42
-agentTrace ON \		43
-routerTrace OFF \		44
-macTrace ON \		45
-movementTrace OFF		46
#节点实例化		
for {set i 0} {\$i < 3} {incr i} {		47
set node (\$i) [\$ns node]		48
\$node(\$i) random-motion 0		49
}		50
#节点位置设置		
\$node_0 set X 30.0		51
\$node_0 set Y 130.0		52
\$node_0 set Z 0.0		53
\$node_1 set X 180.0		54
\$node_1 set Y 130.0		55
\$node_1 set Z 0.0		56
\$node_2 set X 330.0		57
\$node_2 set Y 130.0		58
\$node_2 set Z 0.0		59
#代理和数据流设置		
set udp [new Agent/mUDP]		60
\$udp set_filename sd1		61
\$ns attach-agent \$node_0 \$udp		62
set null [new Agent/mUdpSink]		63
\$null set_filename rd1		64
\$ns attach-agent \$node_1 \$null		65
\$ns connect \$udp \$null		66

set cbr [new Application/Traffic/CBR]	67
\$cbr attach-agent \$udp	68
\$cbr set type_ CBR	69
\$cbr set packet_size_ 1000	70
\$cbr set rate_ 1Mb	71
\$cbr set random_ false	72
\$ns at 1.5 "\$cbr start"	73
\$ns at 15.0 "\$cbr stop"	74
set udp2 [new Agent/mUDP]	75
\$udp2 set_filename sd2	76
\$ns attach-agent \$node_ (2) \$udp2	77
set null2 [new Agent/mUdpSink]	78
\$null2 set_filename rd2	79
\$ns attach-agent \$node_ (1) \$null2	80
\$ns connect \$udp2 \$null2	81
set cbr2 [new Application/Traffic/CBR]	82
\$cbr2 attach-agent \$udp2	83
\$cbr2 set type_ CBR	84
\$cbr2 set packet_size_ 1000	85
\$cbr2 set rate_ 1Mb	86
\$cbr2 set random_ false	87
\$ns at 2.0 "\$cbr2 start"	88
\$ns at 15.0 "\$cbr2 stop"	89
#节点初始化	
for {set i 0} {\$i < 3} {incr i} {	90
\$ns initial_node_pos \$node_ (\$i) 30	91
\$ns at 20.0 "\$node_ (\$i) reset"	92
}	93
#仿真完成处理函数	
proc finish {} {	94
global ns f nf val	95
\$ns flush-trace	96
close \$f	97
close \$nf	98
exec nam Hidden_Terminal.nam &	99
exit 0	100
}	101
#仿真结束	
\$ns at 20.0 "finish"	102
\$ns at 20.1 "puts \"NS EXITING...\"; \$ns halt"	103
#启动仿真	
\$ns run	104

第 1~13 行, 对仿真中需用到的模块参数进行设置, 需注意, 此处并未给出所有参数的设置, 仅对必要的参数进行修改, 若未给出则意味采用 NS2 中的默认值;

第 14~22 行, 对仿真中需用到的变量进行设置, 该部分代码仅为便于修改代码(多处采用相同模块时仅需修改变量即可), 并未增加任何功能;

第 23~28 行, 实例化 ns 变量, 同时设置仿真过程中的 trace 和 nam 输出文件, 以便对实验结果进行分析和演示;

第 29~31 行, 对仿真实验的拓扑范围和全局节点记录进行设置, 以便控制节点位置和运动范围;

第 32~46 行, 在节点实例化前对参数进行配置, 如路由协议、天线模型、物理层模块等, 实现正确有效的隐藏节点实验分析;

第 47~50 行, 对节点进行实例化, 所有节点具有相同配置, 如需更改节点配置, 需在实例化节点前重新用 node-config 进行配置;

第 51~59 行, 对实例化的节点进行位置的设定, 注意, 位置不能超过拓扑范围;

第 60~89 行, 设置两条传输层为 mUDP 的数据流, 应用层都为 CBR, 其中第 1 条流从节点 0 到节点 1, 速率为 1Mbps, 数据包大小为 1000 字节, 仿真起止时间为 1.5s~15.0s; 第 2 条流从节点 2 到节点 1, 速率和数据包大小同第 1 条流, 仿真起止时间为 2.0s~15.0s。节

点1为两条流的共同目标节点, 此时将会出现隐藏节点问题;

第90~93行, 对节点大小进行设置, 用于nam演示, 同时为实验结束时的节点进行reset事件设置, 本实验的实验结束时间为20.0s;

第94~101行, 对仿真结束时的后续处理函数(finish)功能定义, 主要有关闭打开的文件、启动演示动画等, 读者可在其基础上自行扩展;

第102~103行, 仿真结束事件的设置, 此处调用了之前定义的finish函数, 同时输出提示信息, 表明仿真实验的结束并停机;

第104行, 启动ns运行仿真。

5.1.2 隐藏节点修改案例

4.1节基础实验中, 两个发送节点向中间节点发送数据, 二者形成了相互的节点隐藏效应。这里将该问题更改如下: 增加一个发送节点向中间节点发送数据, 如图7.8, 此时节点 n_1 、 n_2 和 n_3 之间会产生相互的节点隐藏效应, 影响数据传输。同样使用RTS/CTS控制报文来解决该问题。

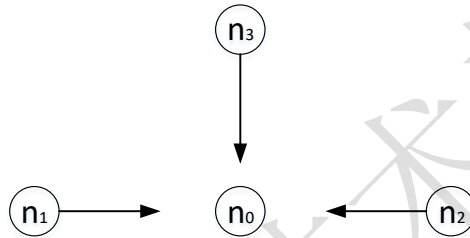


图 7.8 隐藏节点修改拓扑

(1) 修改代码。

根据5.1.1节中对代码的解释, 此时需要修改节点的位置设置并且在数据流的设置中增加一条数据流。下面给出主要修改部分的代码及其解释, 具体请参考电子资源exp7/exp7_extend文件夹中的代码文件(Hidden_Terminal_extend.tcl)

```
#节点位置设置
$node_0 set X_ 30.0.....57
$node_0 set Y_ 130.0.....58
$node_0 set Z_ 0.0.....59
$node_1 set X_ 180.0.....60
$node_1 set Y_ 130.0.....61
$node_1 set Z_ 0.0.....62
$node_2 set X_ 330.0.....63
$node_2 set Y_ 130.0.....64
$node_2 set Z_ 0.0.....65
$node_3 set X_ 180.0.....66
$node_3 set Y_ 230.0.....67
$node_3 set Z_ 0.0.....68

#增加一条数据流
set udp3 [new Agent/mUDP] .....102
$udp3 set_filename sd3.....103
$ns attach-agent $node_3 $udp3.....104
set null3 [new Agent/mUdpSink] .....105
$null3 set_filename rd3.....106
$ns attach-agent $node_1 $null3.....107
$ns connect $udp3 $null3.....108
set cbr3 [new Application/Traffic/CBR] .....109
$cbr3 attach-agent $udp3.....110
$cbr3 set type_ CBR.....111
$cbr3 set packet_size_ 1000.....112
$cbr3 set rate_ 1Mb.....113
$cbr3 set random_ false.....114
$ns at 2.5 "$cbr3 start".....115
$ns at 15.0 "$cbr3 stop".....116
```

第 57~68 行代码：对实例化的节点进行位置的设定，注意其中增加了一个节点。

第 102~116 行代码：新增一条数据流，同前两条一样，设置传输层为 mUDP，应用层为 CBR。该条流从节点 3 到节点 1，速率为 1Mbps，数据包大小为 1000 字节，仿真起止时间为 2.5s~15.0s。

(2) 实验步骤。

首先打开终端(命令“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在的目录；

```
cd WNTLab/experiments/exp7/exp7_extend
```

再输入以下命令运行本实验，如图 7.9 所示；

```
ns Hidden_Terminal.tcl
```

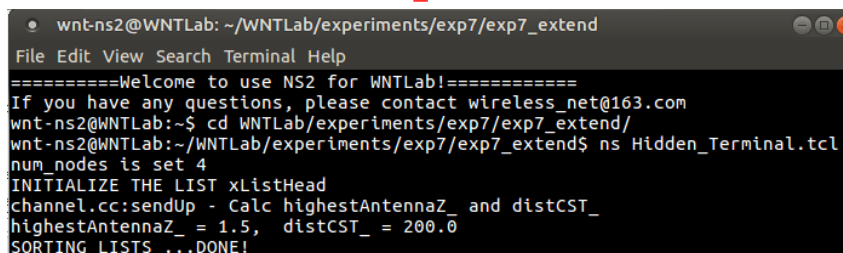


图 7.9 实验步骤

(3) 运行结果。

利用 NAM 动画观察实验结果，如图 7.10 所示。在观看 NAM 动画时，运行步长调整为 2ms 即可。

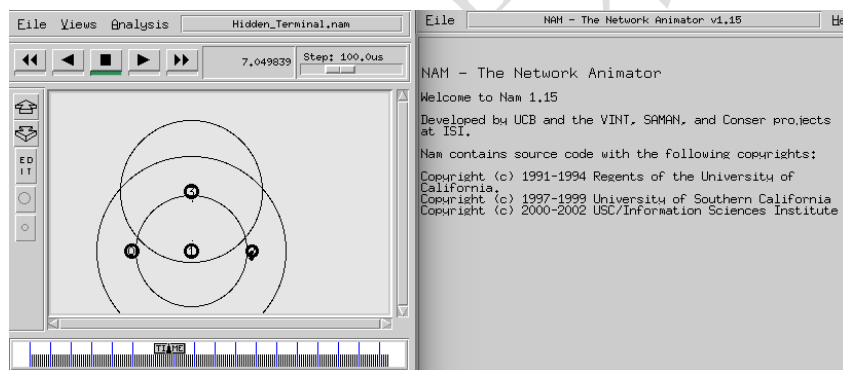


图 7.10 实验结果演示

仿真程序运行说明：

(1) 仿真开始时(0~1s)，节点间相互广播路由(DSDV)报文，建立路由表。

(2) 从 1.5s 开始，节点 0 开始向节点 1 发送数据，需要注意，此时节点 0 需先发送 RTS(实验中设置 802.11 的 RTSThreshold=0，满足数据包大于 RTSThreshold，表示开启 RTS/CTS，解决隐藏终端问题；实现隐藏终端请设置 RTSThreshold=3000，满足数据包小于 RTSThreshold，因为数据包不会大于 3000)，以保证信道(0→1)的顺利占用。

(3) 在 2s 时，节点 2 也开始向节点 1 发送数据，同样，节点 2 也发送 RTS，此时可得知节点 1 的信道不空闲，因此节点 2 开始退避，等待一段时间重新尝试发送。

(4) 在 2.5s 时，节点 3 也开始向节点 1 发送数据，同样，节点 3 也发送 RTS，此时可得知节点 1 的信道不空闲，因此节点 2 开始退避，等待一段时间重新尝试发送。

(5) 仿真过程(2.5~15s)中，可看到节点 0、节点 2 和节点 3 交替向节点 1 发送数据，而非同时传输。仿真时间到达 15s 时，三条流传输同时结束，此后无数据传输，偶尔有链路活报文。

(6) 在 20s 时，仿真实验结束。

5.2 暴露节点问题仿真

5.2.1 暴露节点代码解释

暴露节点与隐藏节点仿真实验的最大区别在于节点位置、数据流的设置。下面给出暴露节点基础仿真实验的主要代码，并且给出代码结构分析，具体代码请见电子资源的 exp7_basic 文件夹中的代码文件(Hidden_Terminal.tcl)。

#节点位置设置

\$node_(0) set X_ 30.0.....	51
\$node_(0) set Y_ 200.0.....	52
\$node_(0) set Z_ 0.0.....	53
\$node_(1) set X_ 130.0.....	54
\$node_(1) set Y_ 200.0.....	55
\$node_(1) set Z_ 0.0.....	56
\$node_(2) set X_ 230.0.....	57
\$node_(2) set Y_ 200.0.....	58
\$node_(2) set Z_ 0.0.....	59
\$node_(3) set X_ 330.0.....	57
\$node_(3) set Y_ 200.0.....	58
\$node_(3) set Z_ 0.0.....	59

#代理和数据流设置

set udp [new Agent/mUDP]	60
\$udp set_filename sd1.....	61
\$ns attach-agent \$node_(1) \$udp.....	62
set null [new Agent/mUdpSink]	63
\$null set_filename rd1.....	64
\$ns attach-agent \$node_(0) \$null.....	65
\$ns connect \$udp \$null.....	66
set cbr [new Application/Traffic/CBR]	67
\$cbr attach-agent \$udp.....	68
\$cbr set type_ CBR.....	69
\$cbr set packet_size_ 1000.....	70
\$cbr set rate_ 1Mb.....	71
\$cbr set random_ false.....	72
\$ns at 1.5 "\$cbr start".....	73
\$ns at 15.0 "\$cbr stop".....	74
set udp2 [new Agent/mUDP]	75
\$udp2 set_filename sd2.....	76
\$ns attach-agent \$node_(2) \$udp2.....	77
set null2 [new Agent/mUdpSink]	78
\$null2 set_filename rd2.....	79
\$ns attach-agent \$node_(3) \$null2.....	80
\$ns connect \$udp2 \$null2.....	81
set cbr2 [new Application/Traffic/CBR]	82
\$cbr2 attach-agent \$udp2.....	83
\$cbr2 set type_ CBR.....	84
\$cbr2 set packet_size_ 1000.....	85
\$cbr2 set rate_ 1Mb.....	86
\$cbr2 set random_ false.....	87
\$ns at 2.0 "\$cbr2 start".....	88
\$ns at 15.0 "\$cbr2 stop".....	89

第 51~59 行，对实例化的节点进行位置的设定，注意，位置不能超过拓扑范围；

第 60~89 行，设置两条传输层为 mUDP 的数据流，应用层都为 CBR，其中第 1 条流从节点 1 到节点 0，速率为 1Mbps，数据包大小为 1000 字节，仿真起止时间为 1.5s~15.0s；第 2 条流从节点 2 到节点 3，速率和数据包大小同第 1 条流，仿真起止时间为 2.0s~15.0s。节点 1、2、4 会出现相互的节点暴露效应。

5.2.2 暴露节点修改案例

4.2 节基础实验中，两个发送节点分别向两个接收节点发送数据，二者形成了相互的节

点暴露效应。这里将该问题修改如下：增加一个发送节点向一个接收节点发送数据，如图 7.10，此时节点 S_1 、 S_2 和 S_3 之间会产生相互的节点暴露效应，影响数据传输。同样使用 RTS/CTS 控制报文来解决该问题。

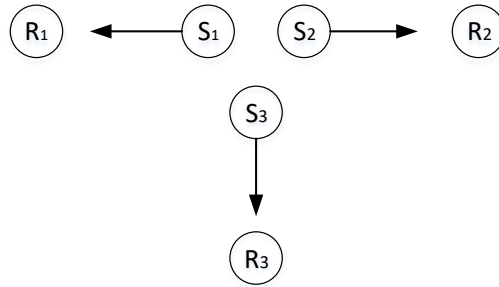


图 7.11 暴露节点问题仿真拓扑

(1) 修改代码。

根据 5.2.1 中对代码的解释，此时需在数据流的设置中增加两个节点和一条数据流，下面给出主要代码及其解释，具体请参考电子资源 exp7/exp7_extend 文件夹中的代码文件 (Exposed_Terminal_extend.tcl)

#节点位置设置

\$node_0 set X_ 10.0.....	58
\$node_0 set Y_ 200.0.....	59
\$node_0 set Z_ 0.0.....	60
\$node_1 set X_ 100.0.....	61
\$node_1 set Y_ 200.0.....	62
\$node_1 set Z_ 0.0.....	63
\$node_2 set X_ 250.0.....	64
\$node_2 set Y_ 200.0.....	65
\$node_2 set Z_ 0.0.....	66
\$node_3 set X_ 350.0.....	67
\$node_3 set Y_ 200.0.....	68
\$node_3 set Z_ 0.0.....	69
\$node_4 set X_ 175.0.....	70
\$node_4 set Y_ 100.0.....	71
\$node_4 set Z_ 0.0.....	72
\$node_5 set X_ 175.0.....	73
\$node_5 set Y_ 0.0.....	74
\$node_5 set Z_ 0.0.....	75

#代理和数据流设置

set udp3 [new Agent/mUDP]	109
\$udp3 set_filename sd3.....	110
\$ns attach-agent \$node_4 \$udp3.....	111
set null3 [new Agent/mUdpSink]	112
\$null3 set_filename rd3.....	113
\$ns attach-agent \$node_5 \$null3.....	114
\$ns connect \$udp3 \$null3.....	115
set cbr3 [new Application/Traffic/CBR]	116
\$cbr3 attach-agent \$udp3.....	117
\$cbr3 set type_ CBR.....	118
\$cbr3 set packet_size_ 1000.....	119
\$cbr3 set rate_ 1Mb.....	120
\$cbr3 set random_ false.....	121
\$ns at 2.5 "\$cbr3 start".....	122
\$ns at 15.0 "\$cbr3 stop".....	123

第 51~59 行，对实例化的节点进行位置设定，注意，位置不能超过拓扑范围；

第 75~89 行，新增的第 3 条数据流，速率和数据包大小同第 1、2 条流，仿真起止时间为 2.5s~15.0s。节点 1、2、4 会出现相互的节点暴露效应。

(2) 实验步骤。

首先打开终端(命令“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在的目录；

cd WNTLab/experiments/exp7/exp7_extend

再输入以下命令运行本实验, 如图 7.12 所示;

ns Exposed_Terminal.tcl

```

wnt-ns2@WNTLab: ~/WNTLab/experiments/exp7/exp7_extend
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp7/exp7_extend/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp7/exp7_extend$ ns Exposed_Terminal.tcl
num_nodes is set 6
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 120.0
SORTING LISTS ...DONE!
  
```

图 7.12 实验步骤

(3) 运行结果。

利用 NAM 动画观察实验结果, 如图 7.13 所示。观看 NAM 动画时, 运行的步长调整为 2ms 即可。

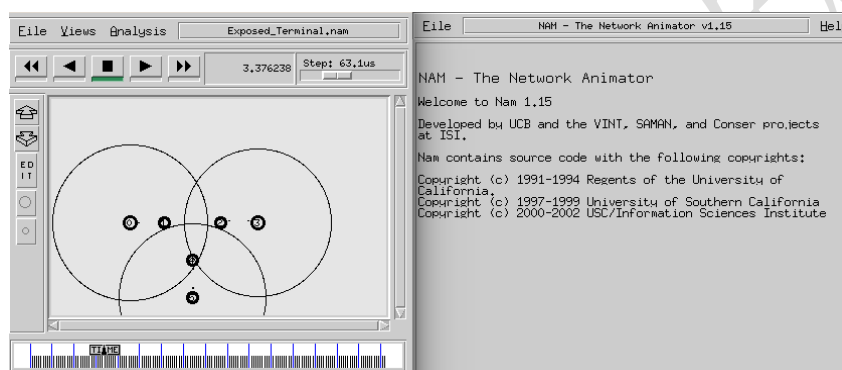


图 7.13 实验结果演示

仿真程序运行说明:

- (1) 仿真开始时(0~1s), 节点间相互广播路由(DSDV)报文, 建立路由表。
- (2) 从 1.5s 开始, 节点 1 开始向节点 0 发送数据, 需注意, 此时节点 1 需要先发送 RTS(实验中设置 802.11 的 $RTSThreshold=0$, 满足数据包大于 $RTSThreshold$, 表示开启 RTS/CTS, 解决暴露终端问题; 实现暴露终端请设置 $RTSThreshold=3000$, 满足数据包小于 $RTSThreshold$, 因为数据包不会大于 3000), 以保证信道(1→0)的顺利占用。
- (3) 在 2s 时, 节点 2 也开始向节点 3 发送数据, 同样, 节点 2 也发送 RTS 报文, 以获得信道(2→3)的顺利占用, 并正确传输数据。
- (4) 在 2.5s 时, 节点 2 也开始向节点 3 发送数据, 同样, 节点 2 也发送 RTS 报文, 以获得信道(2→3)的顺利占用, 并正确传输数据。
- (5) 仿真时间到达 15s 时, 三条流传输同时结束, 此后无数据传输, 偶尔有链路保活报文。
- (6) 在 20s 时, 仿真实验结束。

6 结果分析与讨论

隐藏节点和暴露节点是无线网络传输的经典问题, 其降低了网络吞吐量, 增加了传输时延, 而 CTS/RTS 机制可较好解决隐藏和暴露节点问题。

仿真结果可从两个方面来分析, 即仿真动画和仿真数据。在 NS2 中演示仿真动画采用 NAM, 将拓扑结构、数据包传输情况及队列丢包等以动画形式展现, 仿真结束后, 可通过分析产生的 trace 数据对仿真过程中的延迟、丢包、吞吐量等进行定量分析。

实验八 无线城域网 WiMax 仿真实验

1 实验要求与目的

- 1 人独立完成
- 理解 WiMax 的工作原理
- 利用 NS2 仿真分析 WiMax

2 实验背景与原理

2.1 实验背景

无线城域网(WMAN)和 IEEE 802.16(WiMax)的理论内容详见教材第4章。

2.2 实验原理

基础实验中包含3个节点：汇聚节点、基站节点和移动节点。数据由移动节点产生，经过基站节点转发给汇聚节点。

扩展实验中包含4个节点：汇聚节点、基站节点和两个移动节点。其中两个移动节点都产生数据，数据经过基站节点转发给汇聚节点。

3 实验环境与资源

3.1 基础实验

(1) 环境资源：

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明：

- 树莓派套件见实验一第3.1节介绍；

- 本实验主要用到仿真实验平台中的 NS2 软件，官方 NS2 版本中未包含本实验所需 WiMax 模块，实验一中已提供了实验平台搭建模块和脚本，如果已完成实验一中 wnt-ns2 仿真环境的搭建，本实验可直接运行；

- 本实验提供脚本源代码，供读者学习与二次开发，详见实验资源包 WNTLab/experiments/exp8/exp8_basic。

3.2 扩展实验

(1) 环境资源：

同3.1节。

(2) 使用说明：

- 实验提供源代码，供读者学习与二次开发，详见实验资源包 WNTLab/experiments/exp8/exp8_extend。

4 基础实验步骤与演示

下面详细描述实验步骤，读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 请将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 8.1 所示。



图 8.1 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本。本手册各不同实验在不同子文件夹中, 实验八的子文件夹为“exp8”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

`cd WNTLab/experiments/exp8/exp8_basic/`

再输入以下命令运行本实验, 如图 8.2 所示。

`bash ./run_exp.sh`

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp8/exp8_basic
File Edit View Search Terminal Help
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp8/exp8_basic/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp8/exp8_basic$ bash ./run_exp.sh
Running for profile 1 (OFDM_BPSK_1_2) and CP=0 datarate = 1.04213e+06
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.03125 datarate = 990573
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.0625 datarate = 938904
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.125 datarate = 870025
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.25 datarate = 732352
Running for profile 2 (OFDM_QPSK_1_2) and CP=0 datarate = 2.18474e+06
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.03125 datarate = 2.07645e+06
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.0625 datarate = 1.96876e+06
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.125 datarate = 1.82477e+06
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.25 datarate = 1.5382e+06
Running for profile 3 (OFDM_QPSK_3_4) and CP=0 datarate = 3.2955e+06
Running for profile 3 (OFDM_QPSK_3_4) and CP=0.03125 datarate = 3.13453e+06
Running for profile 3 (OFDM_QPSK_3_4) and CP=0.0625 datarate = 2.97301e+06
Running for profile 3 (OFDM_QPSK_3_4) and CP=0.125 datarate = 2.77946e+06
Running for profile 3 (OFDM_QPSK_3_4) and CP=0.25 datarate = 2.34302e+06
Running for profile 4 (OFDM_16QAM_1_2) and CP=0 datarate = 4.34543e+06
Running for profile 4 (OFDM_16QAM_1_2) and CP=0.03125 datarate = 4.24813e+06
Running for profile 4 (OFDM_16QAM_1_2) and CP=0.0625 datarate = 3.9912e+06
Running for profile 4 (OFDM_16QAM_1_2) and CP=0.125 datarate = 3.70265e+06
Running for profile 4 (OFDM_16QAM_1_2) and CP=0.25 datarate = 3.14633e+06
```

图 8.2 运行仿真实验

(4) 实验正确运行完成后, 打开 home/wnt-ns2/WNTLab/experiments/exp8/exp8_basic 目录下的 WiMax.eps 文件, 可得到图 8.3 所示的结果, 它表示不同循环前缀条件下, 不同调制方式的数据率。

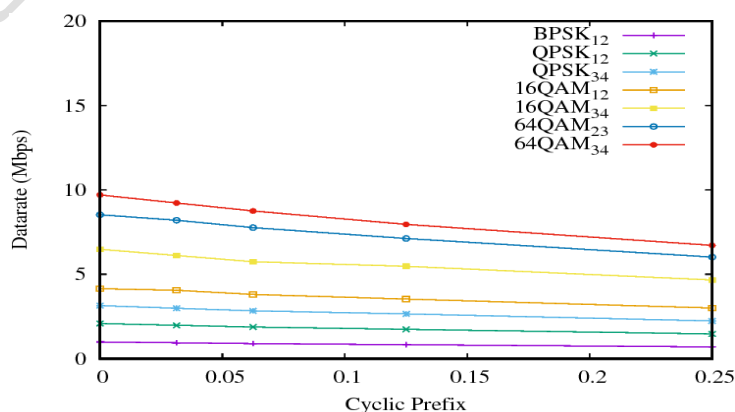


图 8.3 实验结果

5 扩展实验步骤与演示

5.1 基础实验代码分析

分析代码前, 读者需了解 OTcl 脚本语言(教材第2章)。下面给出基础仿真实验的具体代码, 并给出代码结构分析, 具体细节及含义请见实验资源包 WNTLab/experiments/exp8/exp8_basic 中的脚本文件。实验代码文件由以下三部分组成:

(1) WiMax.tcl: 主要脚本文件, 其中使用了 NS2 中的 WiMax 模块, 定义并配置了各个节点与数据流。

```

1.  #全局变量设置
2.  set traffic_start 25
3.  set traffic_stop 35
4.  set simulation_stop 50
5.  #仿真模块参数设置
6.  WimaxScheduler/BS set dlratio_0.2
7.  Mac/802_16 set debug_0
8.  Mac/802_16 set frame_duration_0.005
9.  Mac/802_16 set queue_length_500
10. Mac/802_16 set client_timeout_50
11. Phy/WirelessPhy/OFDM set g_[lindex $argv 1]
12. Phy/WirelessPhy set Pt_0.025
13. Phy/WirelessPhy set RXThresh_2.025e-12
14. Phy/WirelessPhy set CStresh_[expr 0.9*[Phy/WirelessPhy set RXThresh_]]
15. #仿真变量设置
16. set opt(chan) Channel/WirelessChannel
17. set opt(prop) Propagation/TwoRayGround
18. set opt(netif) Phy/WirelessPhy/OFDM
19. set opt(mac) Mac/802_16
20. set opt(ifq) Queue/DropTail/PriQueue
21. set opt(ll) LL
22. set opt(ant) Antenna/OmniAntenna
23. set opt(ifqlen) 50
24. set opt(adhocRouting) DSDV
25. set opt(x) 1100
26. set opt(y) 1100
27. #刷新和关闭文件的函数
28. proc finish {} {
29.   global ns tf output_dir nb_mn
30.   $ns flush-trace
31.   close $tf
32.   exit 0
33. }
34. #开始创建仿真对象
35. set ns [new Simulator]
36. $ns use-newtrace
37. #创建拓扑范围
38. set topo [new Topography]
39. $topo load_flatgrid $opt(x) $opt(y)
40. #创建out.res文件
41. set tf [open out.res w]
42. $ns trace-all $tf
43. #设置分层路由
44. $ns node-config -addressType hierarchical
45. AddrParams set domain_num_2
46. lappend cluster_num 1 1
47. AddrParams set cluster_num_ $cluster_num
48. lappend eilastlevel 1 2
49. AddrParams set nodes_num_ $eilastlevel
50. puts "Configuration of hierarchical addressing done"
51. #设置节点个数
52. create-god 2

```

53. #创建并配置汇聚节点

```
54. set sinkNode [$ns node 0.0.0]
```

```
55. puts "sink node created"
```

56. #创建并配置基站节点

```
57. $ns node-config -adhocRouting $opt(adhocRouting) \
```

```
58. -llType $opt(ll) \
```

```
59. -macType Mac/802_16/BS \
```

```
60. -ifqType $opt(ifq) \
```

```
61. -ifqLen $opt(ifqlen) \
```

```
62. -antType $opt(ant) \
```

```
63. -propType $opt(prop) \
```

```
64. -phyType $opt(netif) \
```

```
65. -channel [new $opt(chan)] \
```

```
66. -topoInstance $topo \
```

```
67. -wiredRouting ON \
```

```
68. -agentTrace ON \
```

```
69. -routerTrace ON \
```

```
70. -macTrace ON \
```

```
71. -movementTrace OFF
```

```
72. set bstation [$ns node 1.0.0]
```

```
73. $bstation random-motion 0
```

```
74. $bstation set X_ 550.0
```

```
75. $bstation set Y_ 550.0
```

```
76. $bstation set Z_ 0.0
```

```
77. [$bstation set mac_(0)] set-channel 0
```

```
78. puts "Base-Station node created"
```

79. #创建并配置移动节点

```
80. $ns node-config -macType Mac/802_16/SS \
```

```
81. -wiredRouting OFF \
```

```
82. -macTrace ON
```

```
83. set wl_node [$ns node 1.0.1]
```

```
84. $wl_node random-motion 0
```

```
85. $wl_node base-station [AddrParams addr2id [$bstation node-addr]]
```

```
86. $wl_node set X_ 400.0
```

```
87. $wl_node set Y_ 550.0
```

```
88. $wl_node set Z_ 0.0
```

```
89. puts "wireless node created ..."
```

```
90. [$wl_node set mac_(0)] set-diuc [lindex $argv 0]
```

```
91. [$wl_node set mac_(0)] set-channel 0
```

92. #代理和数据流设置

```
93. set udp [new Agent/UDP]
```

```
94. $udp set packetSize_ 1500
```

```
95. $ns attach-agent $wl_node $udp
```

```
96. set cbr [new Application/Traffic/CBR]
```

```
97. $cbr set packetSize_ 1500
```

```
98. $cbr set interval_ 0.0005
```

```
99. $cbr attach-agent $udp
```

```
100. set null [new Agent/Null]
```

```
101. $ns attach-agent $sinkNode $null
```

```
102. $ns connect $udp $null
```

```
103. $ns duplex-link $sinkNode $bstation 100Mb 1ms DropTail
```

104. #数据流开始和结束时刻

```
105. $ns at $traffic_start "$cbr start"
```

```
106. $ns at $traffic_stop "$cbr stop"
```

107. #仿真结束

```
108. $ns at $simulation_stop "finish"
```

```
109. puts "Starts simulation"
```

110. #启动仿真

```
111. puts "Simulation done."
```

(2) plot-util: 绘图脚本, 使用“plot”命令定义绘图各参数, 并且指定输出文件为“WiMax.eps”文件。

(3) run_exp.sh: 运行脚本, 执行“WiMax.tcl”文件, 并使用“gnuplot”命令调用“plot-util”文件将仿真结果绘图。

5.2 修改实验代码

第4节基础实验中包含3个节点：移动节点、基站节点和目标节点。数据由移动节点产生，经过基站节点转发给目标节点。现将基础实验中加入一个移动节点，两个移动节点都会产生数据，数据通过基站节点转发给目标节点。

根据5.1节对代码进行修改，下面给出主要修改的代码及其解释，具体请参考电子资源 WNTLab/experiments/exp8/exp8_extend 文件夹中的代码文件。

(1) WiMax.tcl：该脚本中主要增加一个移动节点和一条数据流。

1. **#创建并配置第二个移动节点**
2. set wl_node2 [\$ns node 1.1.0]
3. \$wl_node2 random-motion 0
4. \$wl_node2 base-station [AddrParams addr2id [\$bstation node-addr]]
5. \$wl_node2 set X_ 600.0
6. \$wl_node2 set Y_ 550.0
7. \$wl_node2 set Z_ 0.0
8. puts "wireless node2 created ..."
9. [\$wl_node2 set mac_(0)] set-diuc [lindex \$argv 0]
10. [\$wl_node2 set mac_(0)] set-channel 0
11. **#新增代理和数据流设置**
12. set udp1 [new Agent/UDP]
13. \$udp1 set packetSize_ 1500
14. \$ns attach-agent \$wl_node2 \$udp1
15. set cbr1 [new Application/Traffic/CBR]
16. \$cbr1 set packetSize_ 1500
17. \$cbr1 set interval_ 0.0005
18. \$cbr1 attach-agent \$udp1
19. set null1 [new Agent/Null]
20. \$ns attach-agent \$sinkNode \$null1
21. \$ns connect \$udp1 \$null1
22. \$ns duplex-link \$sinkNode \$bstation 100Mb 1ms DropTail

(2) plot-util：将增加一个plot-util脚本，分别将两条数据流的仿真结果都绘图。

(3) run_exp.sh：增加一条过滤条件。

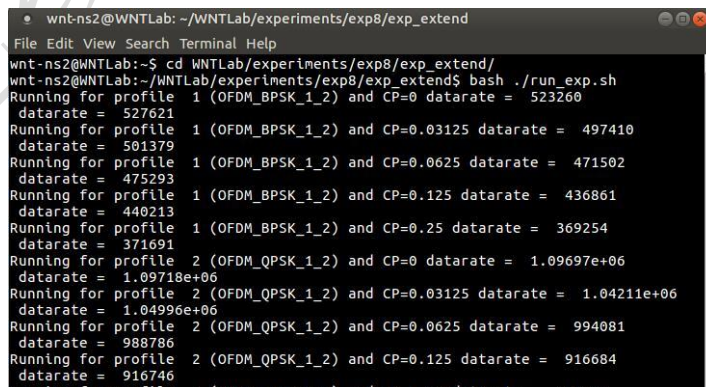
5.3 运行实验

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”)，输入以下命令进入本实验脚本所在文件夹：

```
cd WNTLab/experiments/exp8/exp8_extend/
```

再输入以下命令运行本实验，如图8.4所示。

```
bash ./run_exp.sh
```



```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp8/exp8_extend
File Edit View Search Terminal Help
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp8/exp8_extend/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp8/exp8_extend$ bash ./run_exp.sh
Running for profile 1 (OFDM_BPSK_1_2) and CP=0 datarate = 523260
datarate = 527621
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.03125 datarate = 497410
datarate = 501379
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.0625 datarate = 471502
datarate = 475293
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.125 datarate = 436861
datarate = 440213
Running for profile 1 (OFDM_BPSK_1_2) and CP=0.25 datarate = 369254
datarate = 371691
Running for profile 2 (OFDM_QPSK_1_2) and CP=0 datarate = 1.09697e+06
datarate = 1.09718e+06
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.03125 datarate = 1.04211e+06
datarate = 1.04996e+06
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.0625 datarate = 994081
datarate = 988786
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.125 datarate = 916684
datarate = 916746
Running for profile 2 (OFDM_QPSK_1_2) and CP=0.25 datarate = 772533
```

图8.4 运行仿真实验

(4) 实验正确运行完成后，打开 home/wnt-ns2/WNTLab/experiments/exp8/exp8_extend 目录下的 WiMax1.eps 和 WiMax2.eps 文件，可得到图8.5所示的结果，分别表示两条流中不同循环前缀条件下，不同调制方式的数据率。

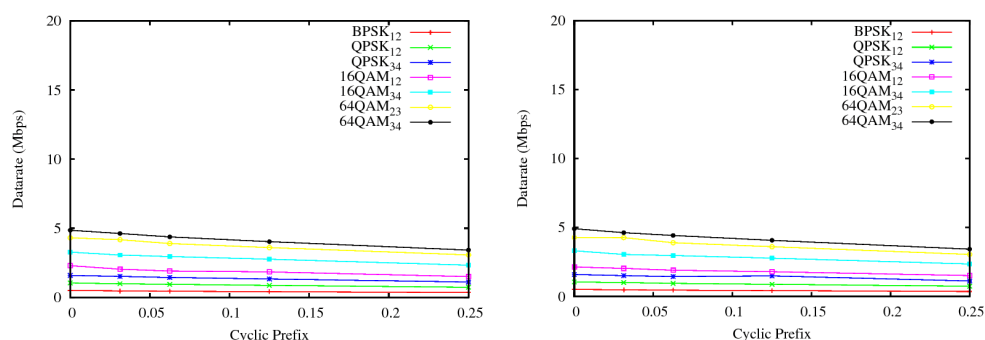


图 8.5 实验结果

6 结果分析与讨论

无线城域网应用广泛, 本实验主要对 WiMax 进行仿真。在仿真分析过程中, 需注意以下两个问题: (1)WiMax 在传输过程中存在哪些特点和不足? (2)无线城域网技术和其它同类技术相比有何不同?

实验九 蜂窝移动网络的数据传输实验

1 实验要求与目的

- 2人共同完成
- 了解蜂窝移动网络和蜂窝通信基本概念
- 掌握 AT 指令实现移动终端与主机通信

2 实验背景与原理

2.1 实验背景

蜂窝移动网络：即常见的移动通信网络，2G/3G/4G/5G 等具体内容知识见教材 4.5 节。

AT 指令：AT 即 Attention，一般用于终端设备与 PC 应用间的连接与通信。其对所传输包大小有限制：即对 AT 指令的发送，除 AT 两个字符外，最多可接收 1056 字符长(包括最后空字符)。每个 AT 命令行中只包含一条 AT 指令；对由终端设备主动向主机报告的 URC 指示或 response 响应，也要求 1 行最多包含 1 个，不允许上报 1 行中有多条指示或响应。

SIM800C：由芯讯通无线科技推出的紧凑型产品，属于四频 GSM/GPRS 模块。SIM800C 采用工业标准接口，工作频率为 GSM/GPRS 的 850/900/1800/1900MHz，可以低功耗实现语音、SMS 和数据信息的传输。

2.2 实验原理

本实验在 SSCOM 串口调试助手中利用 AT 指令对 SIM800C 进行控制，以实现蜂窝移动网络的基础通信功能。

(1) 基础实验中实现蜂窝移动网络的基础功能：SIM800C 模块与手机之间进行语音通话以及短信通信；

(2) 扩展实验中实现 GPRS 功能的应用：使用 SIM800C 访问某一网站，访问成功后利用 SIM800C 的 HTTP GET 功能获取该网站的 html 代码。

3 实验环境与资源

(1) 环境资源

SIM800C 模块一个、5V/2A 电源适配器、USB 转串口线一根、Nano/Micro 转换卡、已开通 GPRS 功能的 SIM 卡一张、手机一部、耳机一副、PC 一台、SSCOM 串口调试助手。

(2) 使用说明

- 各类硬件如图 9.1 至 9.4 所示。
- SSCOM 串口调试助手：作者使用版本为 5.12 版，压缩包大小约 400KB，下载方法请见实验资源包 WNTLab/experiments/exp9，官方下载地址为：<http://www.daxia.com/bibis/>。
- 本实验使用到 AT 指令操作 SIM800C 模块，实验手册中给出了所用指令的解释，实验资源中给出了更多关于 SIM800C 技术文档的下载方式，详见实验资源包 WNTLab/experiments/exp9。



图 9.1 SIM800C 模块



图 9.2 5V/2A 电源适配器



图 9.3 USB 转串口线



图 9.4 Nano/Micro 转换卡

4 基础实验步骤与演示

基础实验内容为使用 SIM800C 模块的语音通话与短信功能；实验要求 2 人共同完成，其中其中 1 人操作智能手机，另 1 人用 AT 指令控制 SIM800C 模块。

实验准备如下所示，准备工作完成后将使用 SIM800C 模块进行语音通话和短信收发。

(1) 将 5V/2A 电源适配器连接至 SIM800C 模块，并将 USB 转串口线的串口插在模块上，将 USB 一头连接在 PC 的 USB 接口并安装 USB 转串口驱动，将耳机(带耳麦)插入耳机口，如图 9.5 所示；

(2) 按照模块背面所示方向，将 SIM 卡装在卡槽内（需注意，SIM800C 模块上卡槽尺寸是 Micro SIM 卡的尺寸，若 SIM 卡是 Nano SIM 卡，需要使用 Nano/Micro 转换卡。）；

(3) 按下 SIM800C 模块的电源开关，此时模块通电，网络状态指示灯亮起并快闪(每秒一次)，表示模块已开机；等待数秒指示灯会慢闪(3 秒 1 次)，表示模块已搜索到蜂窝网络；

(4) 在 PC 中打开串口调试助手，选择波特率为 115200，选择相应串口号并打开串口；

(5) 一部可进行正常语音通话和收发短信的手机备用。



图 9.5 硬件连接

4.1 拨打/接听电话

4.1.1 模块配置

(1) 按步骤输入以下指令

在串口调试助手发送区中输入指令 **AT+COPS?** 并发送;
待模块返回“+COPS: ...” (此处不同的手机卡返回内容不同) 再发送指令 **ATE1**;
待模块返回“OK”后发送指令 **AT+COLP=1**, 模块再次返回“OK”, 如图 9.6 所示。

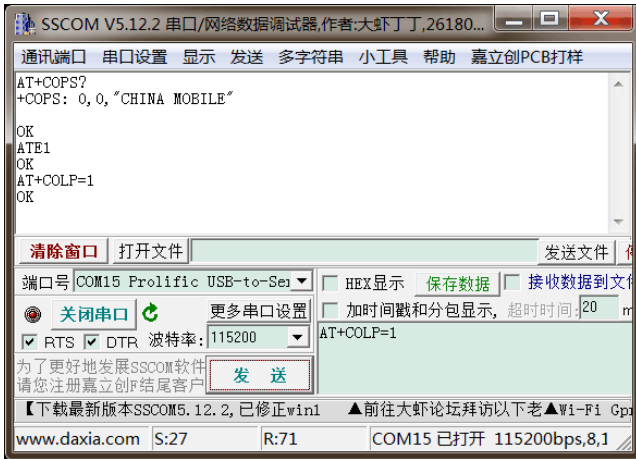


图 9.6 拨打电话前的配置工作

(1) 指令解释如下:

AT+COPS? 用于查询当前运营商, 该指令只有连上网络后, 才返回运营商信息, 否则返回空。如返回 **+COPS:0,0, "CHINA MOBILE"**, 表示当前选择运营商是中国移动。

ATE1 用于设置回显模式 (默认开启), 即模块将收到的 **AT** 指令完整返回发送端, 启用该功能有利于调试模块。如不需要开启回显模式, 则发送 **ATE0** 指令即可关闭, 这样收到指令将不再返回发送端, 这样方便程序控制。

AT+COLP 用于设置被叫号码显示, 这里通过发送 **AT+COLP=1**, 开启被叫号码显示。当成功拨通时 (被叫接听电话), 模块会返回被叫号码。

4.1.2 拨打电话

(1) 按步骤输入以下指令:

如图 9.7 所示, 发送该指令 **ATD150574...;** (数字为要拨打的目标手机号码, 注意指令最后要加上分号“;”)。等待数秒后目标手机接到来电, 如图 9.8 所示。

最后发送指令 **ATH**, 此时通话挂断。

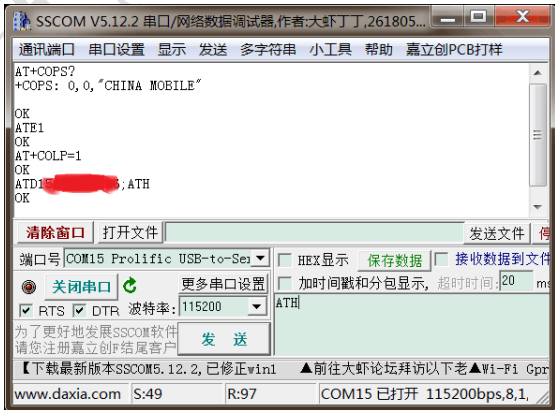


图 9.7 拨打电话并挂断



图 9.8 目标手机接到来电

(2) 指令解释如下:

ATD 用于拨打任意电话号码, 格式为 ATD+号码+;, 末尾的';'一定要加上, 否则不能成功拨号, 如发送 ATD10086;, 即可实现拨打 10086。

ATH 用于挂断电话, 要结束正在进行的通话, 只需给模块发送“ATH”, 即可挂断。

4.1.3 接听来电

(1) 按步骤输入以下指令:

输入指令 **AT+CLIP=1** 并发送;

利用备用手机拨打 SIM800C 模块上的 SIM 卡, 模块收到来电后会连续返回返回指令“RING”和“+CLIP: ...”;

发送指令 **ATA**, 模块返回 OK 后接通来电, 开始通话;

在备用手机上挂断通话, 模块返回指令“NO CARRIER”并结束此次通话, 也可通过发送指令 **ATH** 主动结束通话。



图 9.9 手机拨打电话

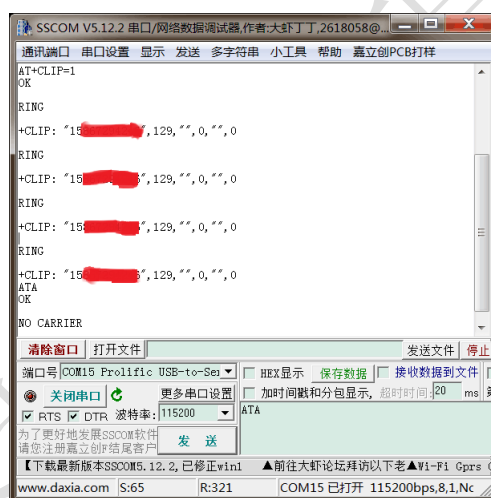


图 9.10 模块接到来电

(2) 指令解释如下:

AT+CLIP 用于设置来电显示, 发送 AT+CLIP=1, 可实现设置来电显示功能, 模块接收到来电时, 会返回来电号码。

ATA 用于应答电话, 当收到来电时, 给模块发送 ATA, 即可接听来电。

4.2 收发短信

4.2.1 发送短信

(1) 按步骤输入以下指令:

如图 9.11 所示, 发送指令 **AT+CSCS="GSM"**;

收到模块回复 OK 之后发送 **AT+CMGF=1** 指令, 至此完成短信发送前的配置。

再发送 **AT+CMGS="150..."** (数字为目标手机号码) 指令, 此时模块会返回指令“>”, 在该指令后输入要发送内容并发送。

发送完成后, 勾选“HEX 发送”再发送指令“**1A**”, 模块返回“+CMGS: ...”和“OK”。至此, 短信发送完成, 在目标手机查看收到的短信, 如图 9.12 所示。



图 9.11 向目标手机号发送短信



图 9.12 目标手机收到短信

(2) 指令解释如下:

AT+CSCS 用于设置 TE 字符集, 默认为 GSM 7 位缺省字符集, 在发送纯英文短信时候, 发送: AT+CSCS="GSM", 设置为缺省字符集即可。在发送中英文短信时, 需发送: AT+CSCS="UCS2", 设置为 16 位通用 8 字节倍数编码字符集。

AT+CMGF 用于设置短消息发送模式, SIM800C 支持 PDU 模式和文本模式等, 发送: AT+CMGF=1, 即设置为文本模式。

AT+CMGS="..." 用于发送短信, 在"GSM"字符集下, 最多可发送 180 字节的英文字符, 在"UCS2"字符集下, 最多可发送 70 个汉字(包括字符/数字)。

0X1A 用于告诉 SIM800C, 要执行发送操作。还可发送: 0X1B, 即“ESC”的键值, 用于告诉 SIM800C, 取消本次操作, 不执行发送。

4.2.2 接收短信

(1) 按步骤输入以下指令:

在串口调试助手发送指令 AT+CNMI=2,1, 模块返回“OK”;

然后, 利用备用手机向 SIM800C 模块发送一条短信, 如图 9.13 所示;

待串口调试助手收到+CMTI: "SM", 3 指令后, 表示接收到的短信存储在 SIM 卡的位置 3, 再发送指令 AT+CMGR=<Pram>。其中<Pram>代表短信存储在 SIM 卡位置的编号, 此处为 3。如图 9.14, 串口调试助手会显示出收到的短信内容、时间以及发送者号码。(需注意, 收到的短信可能会存储在 SIM 卡不同的位置, 此处的<Pram>参数需要根据上一步的返回情况而定。)

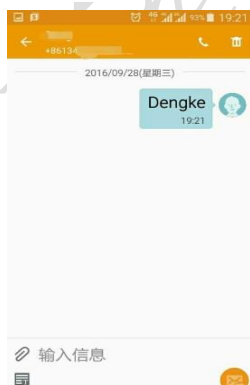


图 9.13 向 SIM800C 模块发短信

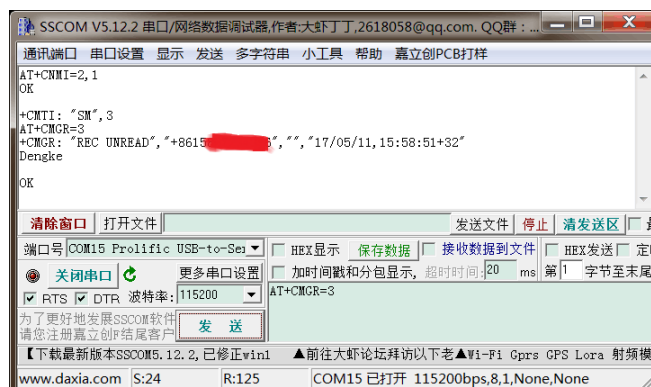


图 9.14 SIM800C 模块收到短信

(2) 指令解释如下:

AT+CNMI 用于设置新消息指示。发送 AT+CNMI=2,1, 设置新消息的提示, 当收到新消息, 且 SIM 卡未滿时, SIM800C 模块会返回数据给串口, 如+CMTI: "SM",2 表示

收到接收到新消息, 存储在 SIM 卡的位置 2。

AT+CMGR 用于读取短信内容, 如发送 AT+CMGR=1, 则可读取 SIM 卡存储在位置 1 的短信。

5 扩展实验步骤与演示

扩展实验内容为应用 SIM800C 模块的 GPRS 功能; 实验要求 1 人独立完成。

5.1 访问 HTTP

5.1.1 初始化设置

(1) 按步骤输入以下指令:

在发送区输入命令 **AT+CGATT?** 并发送;

待模块返回 “+CGATT: 1” 和 “OK” 以后, 发送指令 **AT+SAPBR=3,1,"CONTYPE","GPRS"**;

待模块返回 “OK” 后, 发送指令 **AT+SAPBR=3,1,"APN","CMNET"**;

待模块返回 “OK” 后, 发送指令 **AT+SAPBR=1,1**;

待模块返回 “OK” 后, 发送指令 **AT+HTTPINIT**;

待模块返回 “OK” 后, 读者可输入任意可访问的网址, 发送指令 **AT+HTTPPARA="URL","www.thinkmesh.net/wireless/"**;

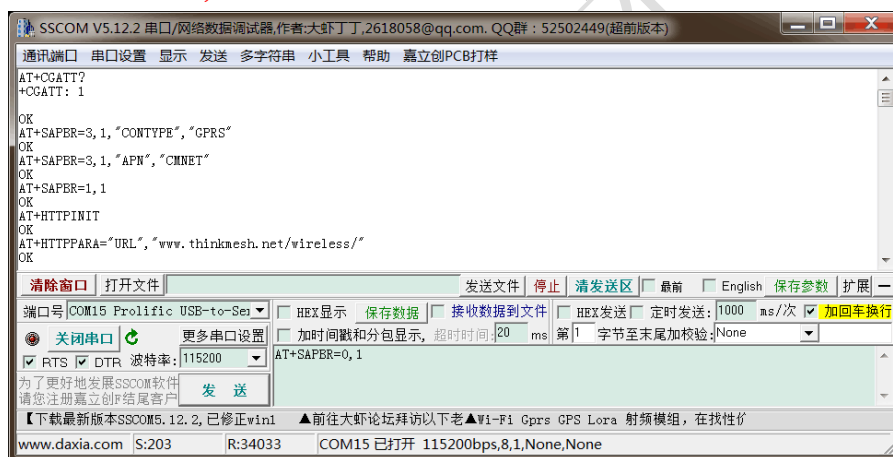


图 9.15 初始化设置

(2) 指令解释如下:

AT+CGATT? 用于查询 GPRS 业务处于附着还是分离状态, 如果返回 “+CGATT: 1” 表示以附着 GPRS 业务, 否则需要使用指令 AT+CGATT=1 设置附着 GPRS 业务;

AT+SAPBR=3,1,"CONTYPE","GPRS" 用于配置承载场景;

AT+SAPBR=3,1,"APN","CMNET" 用于设置接入点名称;

AT+SAPBR=1,1 用于激活一个 GPRS 上下文;

AT+HTTPINIT 用于初始化 HTTP 服务;

AT+HTTPPARA="URL","www.thinkmesh.net/wireless/" 用于设置 HTTP 会话参数, 即设置需要访问的网站。

5.1.2 开始访问

(1) 按步骤输入以下指令:

输入指令 **AT+HTTPACTION=0** 并发送, 模块会返回 “OK”, 数秒后模块返回 “+HTTPACTION:0,200,33719” 字样 (**需注意, 在返回该数据之前请勿进行任何操作**);

输入指令 **AT+HTTPREAD** 并发送, 模块会返回 “+HTTPREAD: 33719”, 并且开始读

取 HTTP 服务器的参数, 如图 9.16 所示。数秒后读取完成, 返回“OK”, 如图 9.17 所示。

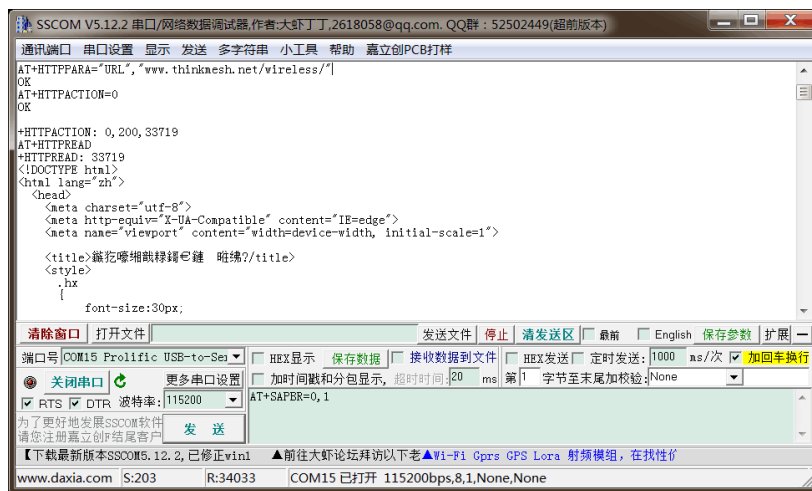


图 9.16 读取 HTTP 服务器的参数

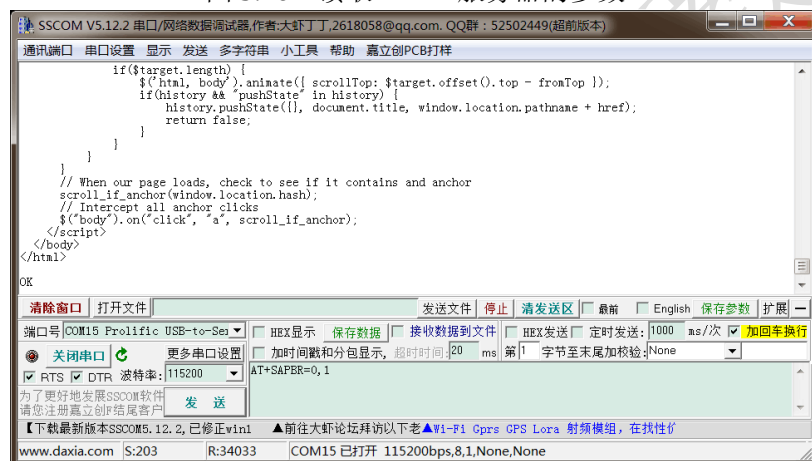


图 9.17 读取完成

(2) 指令解释如下:

AT+HTTTPACTION=0 表示访问 HTTP 会话开始;

AT+HTTPREAD 用于读取 HTTP 会话参数。

5.1.3 结束访问

(1) 按步骤输入以下指令:

输入指令 **AT+HTTTPTERM** 并发送, 模块返回“OK”;

输入指令 **AT+SAPBR=0,1** 并发送, 模块返回“OK”, 如图 9.18 所示。

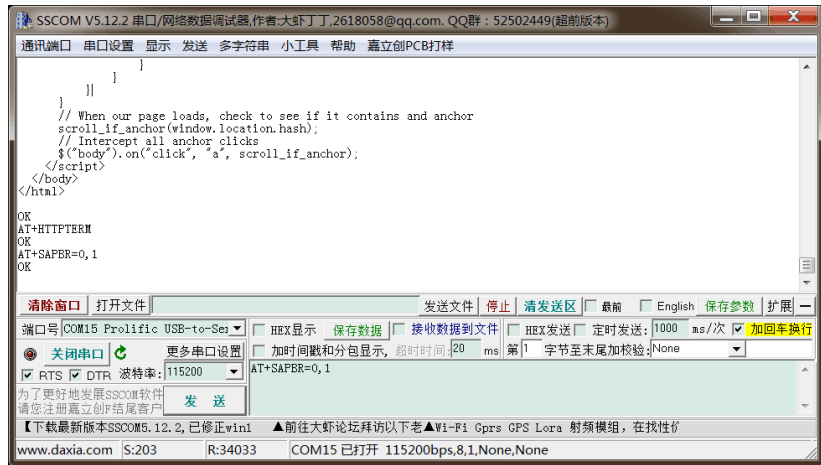


图 9.18 结束访问

(2) 指令解释如下:

AT+HTTPTERM 用于结束 HTTP 服务;

AT+SAPBR=0,1 用于关闭 GPRS 上下文。

6 结果分析与讨论

本实验学习了蜂窝移动网络与蜂窝移动通信的基本概念, 并利用 AT 指令控制 SIM800C 模块实现了语音、短信、GPRS 功能。

7 注意事项

实验中输入指令时, 请务必使用英文输入法输入, 如果发送指令后没有得到正确的返回结果, 请关闭模块电源后并重新打开进行实验。

实验十 卫星网络系统仿真实验

1 实验要求与目的

- 1人独立完成
- 理解 Iridium 和 Teledesic 卫星网络的基本原理
- 利用 NS2 仿真实现 Iridium 和 Teledesic 卫星系统

2 实验背景与原理

2.1 实验背景

Iridium 和 Teledesic 等卫星网络知识内容, 读者可阅读教材第5章。

2.2 实验原理

卫星网络为三维结构, 在 NAM 动画中难以演示, 因此本实验以数据分析形式予以呈现。

实验中设置了5条数据流进行对比, 起点均为北京, 终点分别为上海、香港、悉尼、纽约和里约热内卢。为便于分析和展示实验效果, 实验采用 CBR 作为数据源, 并利用 UDP 进行数据传输。分析 trace 文件(NS2 运行仿真时产生的数据文件, 需要在仿真中设置)来对比时延的变化和通信距离的关系, 实际数据传输路径中卫星节点个数不同, 也会造成端到端的时延不同。

3 实验环境与资源

(1) 环境资源:

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第3.1节介绍;
- 本实验主要用到仿真实验平台中的 NS2 软件, 实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;
- 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp10。

4 实验步骤与演示

4.1 Iridium 仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第4.1节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 10.1 所示。



图 10.1 树莓派连接电源和显示器

(2) “home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本。本手册各不同实验在不同子文件夹中，实验十的子文件夹为“exp10”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

`cd WNTLab/experiments/exp10/Iridium /`

再输入以下命令运行本实验，如图 10.2 所示。

`ns sat-Iridium.tcl`

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp10/Iridium
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp10/Iridium/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp10/Iridium$ ns sat-Iridium.tcl
wnt-ns2@WNTLab:~/WNTLab/experiments/exp10/Iridium$
```

图 10.2 运行仿真实验

(4) 利用 awk 脚本分析 trace 文件 (awk 编程相关知识请参考：<http://www.gnu.org/software/gawk/manual/gawk.html>), 主要性能指标有延迟、抖动、丢包和吞吐率。下面以北京到上海的 Iridium 延迟为例进行分析。注意需将 awk 脚本和 trace 文件放置在相同目录下或正确指明文件路径。图 10.3 所示为北京至上海延迟分析的 awk 脚本，可采用命令：`awk -f delay.awk sat-Iridium.tr > delay.txt` 进行分析，其中 delay.awk 为图 10.3 所示的脚本，sat-Iridium.tr 为 trace 文件，>表示重定向输出，delay.txt 为分析的结果文件。

```
BEGIN{
    highest_packet_id=0;
}
{
    action=$1;
    time=$2;
    from=$3;
    to=$4;
    type=$5;
    size=$6;
    flow_id=$8;
    src=$9;
    dst=$10;
    seq_no=$11;
    packet_id=$12;
    if (packet_id > highest_packet_id)
        highest_packet_id = packet_id;
    if (start_time[packet_id]==0)
        start_time[packet_id] = time;
    if(flow_id == 0 && action != "d"){
        if(action=="r" && to=="67"){
            end_time[packet_id]=time;
```

```

    }else{
        end_time[packet_id]=-1;
    }
}
}
END{
    for (packet_id=0;packet_id <= highest_packet_id;packet_id++){
        start=start_time[packet_id];
        end=end_time[packet_id];
        delay=end-start;
        if(start<end)printf("%f %f\n",start,delay)
    }
}

```

图 10.3 北京至上海延迟分析的 awk 脚本

(4) 分析完 trace 文件后, 获得结果可进行展示, 以进行直观对比。本实验采用 gnuplot 程序, gnuplot 的更多知识可参见其官方网站。具体步骤为: 在终端中执行 gnuplot 命令, 接着执行 plot “delay.txt” using 1:2 w lp 命令, 可得到图 10.4 所示的结果。

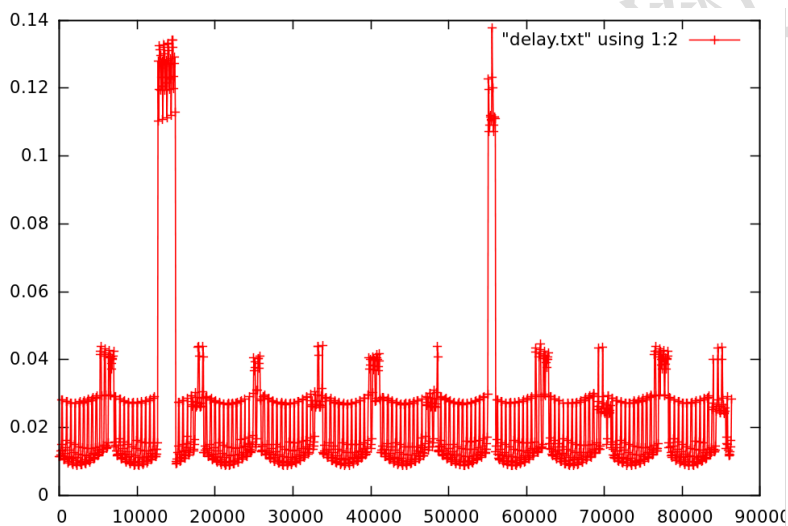


图 10.4 北京至上海延迟的数据展示图形

4.2 Teledesic 仿真分析

(1) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

```
cd WNTLab/experiments/exp10/Teledesic/
```

再输入以下命令运行本实验, 如图 10.5 所示。

```
ns sat-Iridium.tcl
```

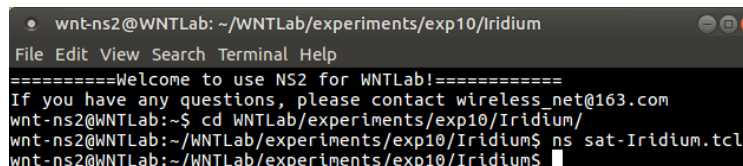


图 10.5 运行仿真实验

```

BEGIN{
    highest_packet_id=0;
}
{
    action=$1;
    time=$2;
    from=$3;
    to=$4;
    type=$5;
    size=$6;
    flow_id=$8;
    src=$9;
    dst=$10;
    seq_no=$11;
    packet_id=$12;
    if (packet_id > highest_packet_id)
        highest_packet_id = packet_id;
    if (start_time[packet_id]==0)
        start_time[packet_id] = time;
    if (flow_id == 0 && action != "d"){
        if (action == "r" && to == "67"){
            end_time[packet_id] = time;
        } else {
            end_time[packet_id] = -1;
        }
    }
}
}
END{
    for (packet_id=0; packet_id <= highest_packet_id; packet_id++){
        start=start_time[packet_id];
        end=end_time[packet_id];
        delay=end-start;
        if (start<end) printf("%f %f\n", start, delay)
    }
}

```

图 10.6 北京至上海延迟分析的 awk 脚本

(2) 利用 awk 脚本分析 trace 文件 (awk 相关知识请参考：<http://www.gnu.org/software/gawk/manual/gawk.html>), 主要性能指标有延迟、抖动、丢包和吞吐率。下面以北京到上海的 Teledesic 延迟为例进行分析。注意需将 awk 脚本和 trace 文件放置在相同目录下或正确指明文件路径。图 10.6 为北京至上海延迟分析的 awk 脚本, 可采用命令: `awk -f delay.awk sat-teledesic.tr > delay.txt`, 其中 delay.awk 为图 10.6 所示的脚本, sat-teledesic.tr 为 trace 文件, > 表示重定向输出, delay.txt 为分析结果文件。

(3) 分析完 trace 文件后, 继续采用 gnuplot, 在终端中执行 gnuplot 命令, 接着执行 plot “delay.txt” using 1:2 w lp 命令, 即可得到图 10.7 所示的结果。

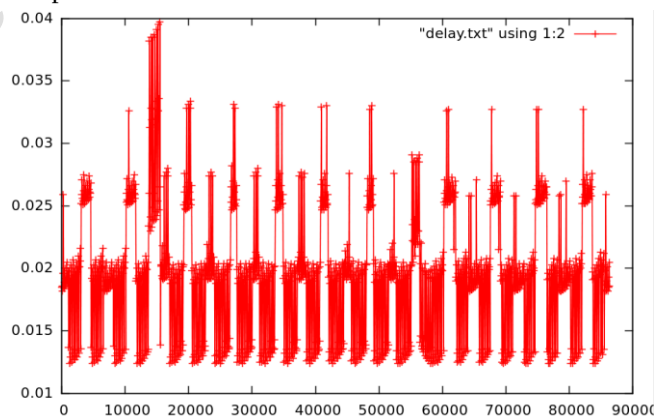


图 10.7 北京至上海延迟的数据展示

5 实验扩展与分析

在本实验基础上,读者可自己动手设计不同网络场景下的仿真,分析 Iridium 和 Teledesic 卫星网络的性能,分析解读实验相关的程序代码。

下面给出对卫星网络 trace 文件的简要解释:

```
+ 421.0000 288 169 cbr 210 ----- 0 288.0 289.0 7 7 39.54 116.28 44.57 108.48
- 421.0000 288 169 cbr 210 ----- 0 288.0 289.0 7 7 39.54 116.28 44.57 108.48
r 421.0068 288 169 cbr 210 ----- 0 288.0 289.0 7 7 39.54 116.28 44.57 108.48
+ 421.0068 169 193 cbr 210 ----- 0 288.0 289.0 7 7 44.57 108.48 37.12 122.27
- 421.0068 169 193 cbr 210 ----- 0 288.0 289.0 7 7 44.57 108.48 37.12 122.27
r 421.0125 169 193 cbr 210 ----- 0 288.0 289.0 7 7 44.57 108.48 37.12 122.27
+ 421.0125 193 289 cbr 210 ----- 0 288.0 289.0 7 7 37.12 122.27 31.12 121.26
- 421.0125 193 289 cbr 210 ----- 0 288.0 289.0 7 7 37.12 122.27 31.12 121.26
r 421.0189 193 289 cbr 210 ----- 0 288.0 289.0 7 7 37.12 122.27 31.12 121.26
+ 481.0000 288 169 cbr 210 ----- 0 288.0 289.0 8 8 39.54 116.28 47.72 108.85
- 481.0000 288 169 cbr 210 ----- 0 288.0 289.0 8 8 39.54 116.28 47.72 108.85
r 481.0073 288 169 cbr 210 ----- 0 288.0 289.0 8 8 39.54 116.28 47.72 108.85
+ 481.0073 169 193 cbr 210 ----- 0 288.0 289.0 8 8 47.72 108.85 40.28 122.50
- 481.0073 169 193 cbr 210 ----- 0 288.0 289.0 8 8 47.72 108.85 40.28 122.50
r 481.0128 169 193 cbr 210 ----- 0 288.0 289.0 8 8 47.72 108.85 40.28 122.50
+ 481.0128 193 289 cbr 210 ----- 0 288.0 289.0 8 8 40.28 122.50 31.12 121.26
- 481.0128 193 289 cbr 210 ----- 0 288.0 289.0 8 8 40.28 122.50 31.12 121.26
```

Trace 文件每条记录共 16 列, 每列之间用空格隔开, 各列的解释如下:

第 1 列: 事件(r: Receive 接收;d: Drop 丢弃;e: Error 错误;+: Enqueue 入队;-: Dequeue 出队);
第 2 列: 产生事件的时间;第 3 列: 产生事件的源节点;第 4 列: 产生事件的目标节点;第
5 列: 数据包类型;第 6 列: 数据包大小;第 7 列: 标志位;第 8 列: 流 ID;第 9 列: 源
节点:端口;第 10 列: 目标节点:端口;第 11 列: 序列号;第 12 列: 报文唯一标识;第 13
列: 源纬度;第 14 列: 源经度;第 15 列: 目标纬度;第 16 列: 目标经度。

6 结果分析与讨论

相关的分析脚本详见电子资源。通过分析 trace 文件的数据,可获知仿真过程中的延迟、丢包、吞吐量等指标。

有关本实验的结果分析和代码说明详见教材 5.7 节。

7 注意事项

实验利用 awk 脚本分析了 Iridium 卫星网络的 5 个场景,分别为北京到上海、香港、悉尼、纽约和里约热内卢,所有实验场景的分析脚本位于电子资源中实验源代码的 exp10 目录下。对于每一个场景,主要分析了其延迟、抖动、丢包率和吞吐量。

实验十一 GPS 室外定位实验

1 实验要求与目的

- 1 人独立完成
- 了解 Arduino 单片机的使用
- 实现 Arduino 控制 WiFi 模块传输位置信息

2 实验背景与原理

2.1 实验背景

GPS、WiFi 的背景知识请参考教材第 3 章、第 5 章。

2.2 实验原理

本实验为 GPS 实测实验, 位于室外开阔区域, 作者实验场地如图 11.1 蓝色圆点处所示, 地图显示此处坐标为: 北纬 $29^{\circ} 54' 32.88''$, 东经 $121^{\circ} 37' 54.18''$ 。实验结束后, 对比实验结果与该坐标。



图 11.1 实验场地

实验内容分为两部分:

(1) 基础实验: 使用 GPS 模块收集当前位置的经度、纬度坐标和海拔高度, 由 Arduino Mega 2560 单片机模块处理数据, 最后通过串口通信将处理后的数据传输到 PC 端, 如图 11.2 所示。

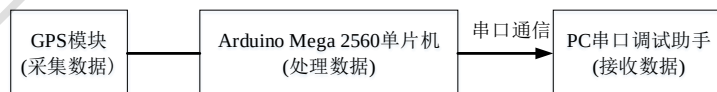


图 11.2 基础实验原理

(2) 扩展实验: 结合 WiFi 网络与 GPS 进行开发性实验, 同样使用 GPS 模块收集当前位置的经度、纬度坐标和海拔高度, 由 Arduino Mega 2560 单片机模块处理数据, 通过 WiFi 模块建立 TCP 连接将数据发送到服务器, 在服务器端显示位置坐标, 如图 11.3 所示。

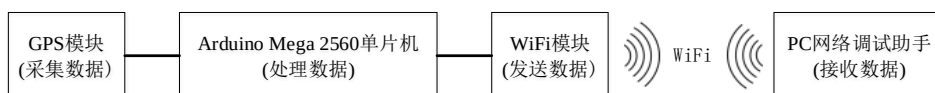


图 11.3 扩展实验原理

3 实验环境与资源

(1) 环境资源:

PC 一台、Arduino Mega 2560 单片机模块一个、GPS 模块一个、WiFi 模块一个、USB 转串口模块一个、USB A 型转 B 型线一根、电池一组、公对母杜邦线若干、公对公杜邦线若干、Arduino IDE、网络调试助手、SSCOM 串口调试助手。

(2) 使用说明

- 硬件如图 11.4 至 11.5, 使用方法如实验步骤;

- Arduino IDE 是 Arduino 系列单片机的开发环境, 考虑到稳定性, 推荐读者使用 1.0.x 系列版本, 官网下载连接为 <https://www.arduino.cc/en/Main/OldSoftwareReleases#previous>, 详细的下载方法及说明请见实验资源包 WNTLab/experiments/exp11。

- 网络调试助手: 作者使用版本为 4.2.5, 压缩包大小约 500KB, 下载方法请见实验资源包 WNTLab/experiments/exp11, 官方下载地址为: <http://www.cmssoft.cn/>。

- SSCOM 串口调试助手: 作者使用版本为 5.12 版, 压缩包大小约 400KB, 下载方法请见实验资源包 WNTLab/experiments/exp11, 官方下载地址为: <http://www.daxia.com/bibis/>。

- 本实验开放所有源码, 供读者学习和二次开发, 详见实验资源包 WNTLab/experiments/exp11。



图 11.4 NEO-7N GPS 模块



图 11.5 ESP8266-1 WiFi 模块

4 基础实验步骤与演示

基础实验内容为使用 Arduino Mega 2560 单片机控制 GPS 模块进行室外定位, 要求 1 人独立完成。

4.1 程序烧录

参考实验十六 4.1 节 Arduino 代码烧录过程, 打开 Arduino IDE, 编译本实验代码, 并烧录到 Arduino Mega 2560 单片机中。(需注意, 烧录代码时, Arduino 上不要连接任何模块, 否则会导致烧录失败。)

本程序功能是利用 Arduino Mega 2560 单片机控制 GPS 模块进行室外定位, 输出当前位置的经度、纬度以及海拔数据。程序流程为: GPS 模块获取 GPS 原始数据, 并将之存放到缓冲区; 单片机解析 GPS 数据, 并存放到相应数组中; 通过串口输出经度、纬度、海拔数据。

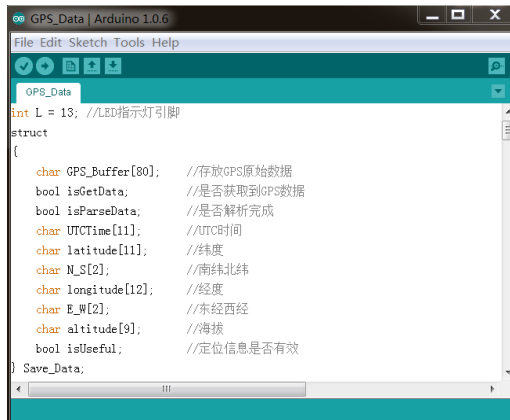


图 11.6 基础实验程序

4.2 硬件配置及连线

(1) 本实验所用 GPS 模块如图 11.7 所示, 该模块附有外接天线接口和板载陶瓷天线。由于 GPS 设备只能在室外接收卫星信号, 所以采用板载陶瓷天线接收信号时需要将该模块置于室外, 若读者自行外接天线时可将天线置于室外, 模块置于室内使用。本实验以使用板载陶瓷天线为例。

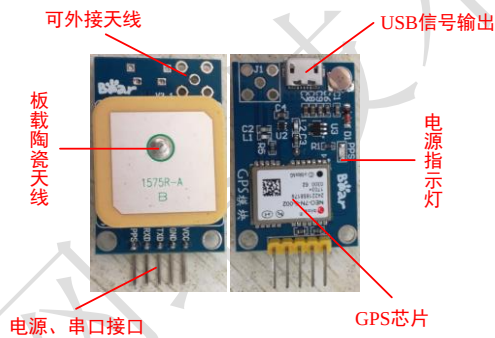


图 11.7 GPS 模块简介

(2) 使用公对母杜邦线, 将 GPS 模块与 Arduino Mega 2560 单片机模块按照图 11.8 所示连接。(**需注意**, GPS 模块的 RX 和 TX 引脚需要和单片机的 RX 和 TX 引脚交叉连接。)

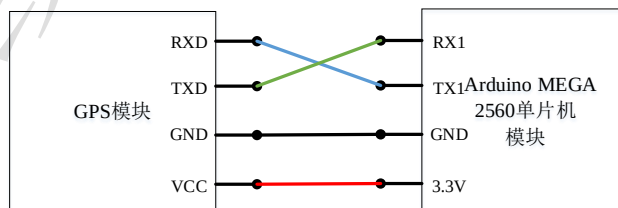


图 11.8 GPS 模块与单片机模块连接示意图

(3) 将 GPS 模块置于室外, 通过 USB A 型转 B 型线连接 Arduino Mega 2560 单片机和 PC (读者可选择较开阔的窗台、阳台等场景)。

4.3 室外定位

(1) 打开 SSCOM 串口调试助手, 设置波特率为 9600, 并打开相应的端口号, 如图 11.9 所示。

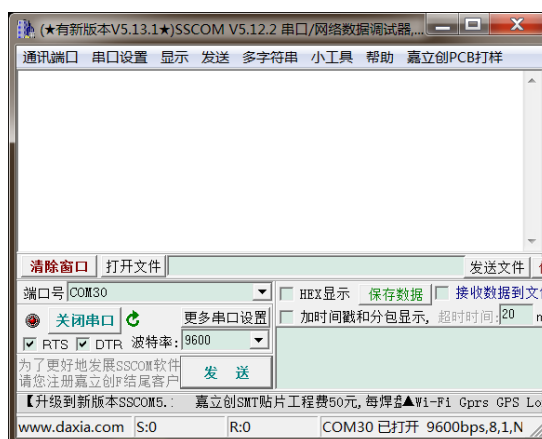


图 11.9 设置 SSSCOM 串口调试助手

(2) 观察串口输出数据: GPS 模块刚连通电源后, 电源指示灯常亮, 此时模块已开始工作, 但未实现定位, 串口助手中显示 GPS 数据不可用字样, 如图 11.10 所示; 等待 1 分钟左右, GPS 模块电源指示灯开始闪烁, 此时定位成功, 串口助手中输出定位信息, 如图 11.11 所示。

定位信息为北纬 2954.54462, 东经 12137.91063, 转换为度分秒表示形式的方法如下:

纬度: $2954.54462/100=29^{\circ}$
 $2954.54462\%100=54'$
 $0.54462*60=32.6772''$

同理, 经度为 $121^{\circ} 37' 54.6378''$, 即该定位结果与地图显示数据 (北纬 $29^{\circ} 54' 32.88''$, 东经 $121^{\circ} 37' 54.18''$) 基本一致, 定位成功。

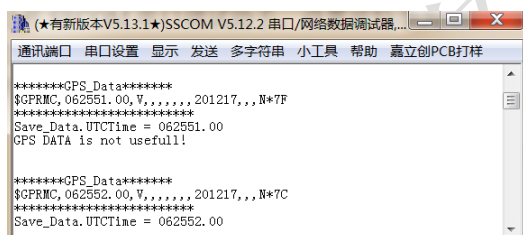


图 11.10 等待 GPS 信号

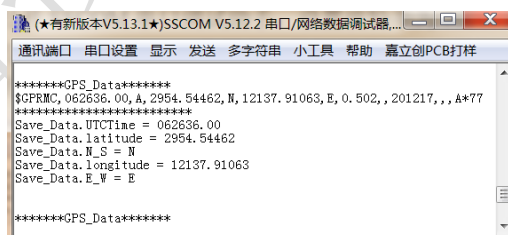


图 11.11 定位成功

5 扩展实验步骤与演示

实验三中已经进行了无线局域网的组网实验, 此处将结合 WiFi 进行开发性实验: 首先在 PC 端建立 TCP 服务器, 利用单片机控制 GPS 模块进行室外定位, 再将定位内容通过 WiFi 模块发送到 TCP 服务器端。(**需注意, 扩展实验中 PC 和 WiFi 模块均需要连接到 WiFi 网络, 读者可选用实验三中组建的网络或者身边可用的网络。**)

5.1 实验准备

首先需构建无线局域网环境, 步骤如下:

(1) 根据实验三第 4.1 节步骤, 组建无线局域网, 可以选择不接入 Internet, 实验在本地局域网下进行即可。(读者也可选择已有的无线局域网进行实验)

(2) 在 PC 端连接组建好的 WiFi 网络。(**需注意, 如果 PC 连接了有线网络, 请在 PC 的网络连接中禁用有线连接, 以确保 PC 使用的是该无线局域网。**)

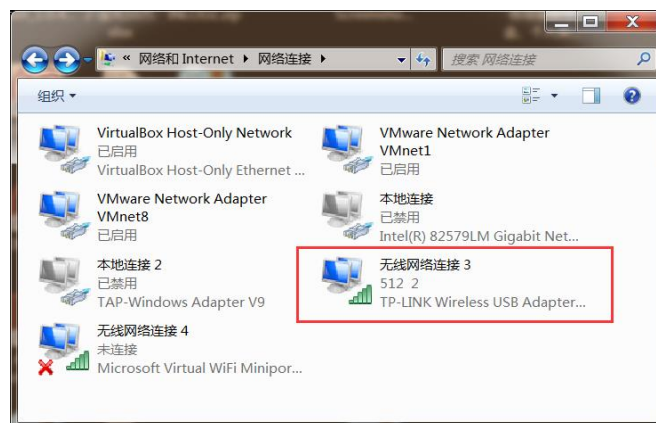


图 11.12 设置 SSCOM 串口调试助手

(3) 在 PC 端打开网络调试助手, 选择“协议类型”为 TCP Server, 设置“本地 IP 地址”为 PC 获得的无线局域网 IP 地址, 设置端口号, 并点击“开始监听”按钮, 建立一个 TCP 服务器开始监听数据, 如图 11.13 所示。(需注意, 读者可在 Windows 系统命令提示符中使用 `ipconfig` 命令查看 PC 获得的局域网 IP 地址。)

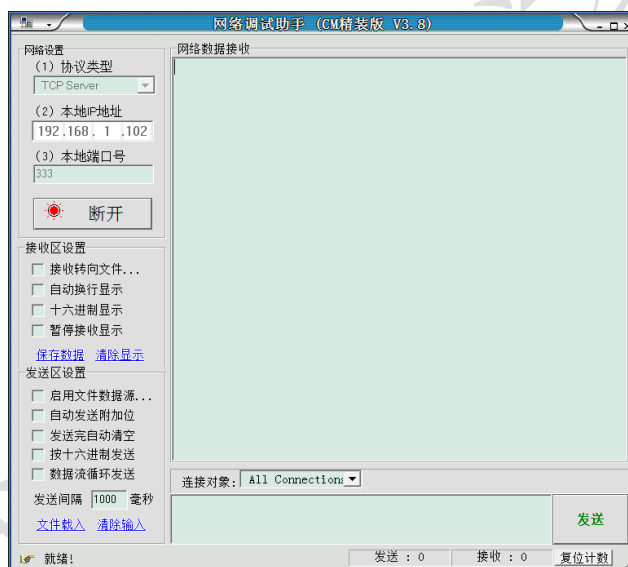
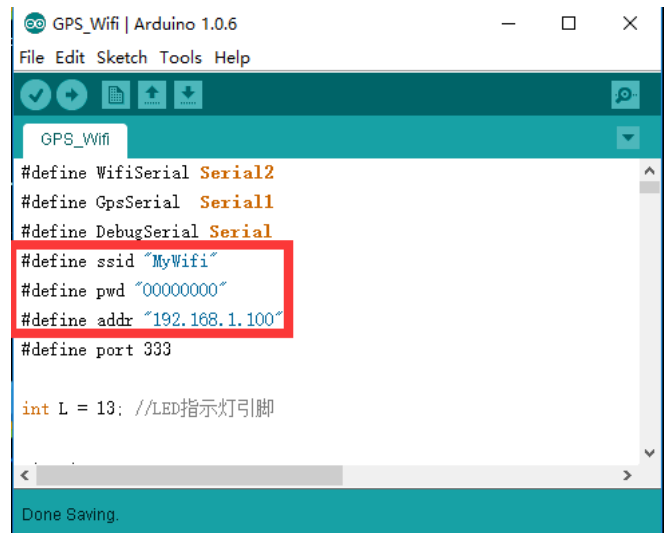


图 11.13 设置网络调试助手

(4) 修改实验程序。修改如图 11.14 红色框处定义的“ssid”、“pwd”、“addr”三个字段的具体内容, 三个字段内容分别为步骤(2)和(3)中所连接 WiFi 的名称、密码以及 PC 获得的局域网 IP 地址。

(5) 烧录实验程序。参考实验十六 4.1 节 Arduino 代码烧录过程, 打开 Arduino IDE, 编译本实验代码, 并烧录到 Arduino Mega 2560 单片机中。(需注意, 烧录代码时, Arduino 上不要连接任何模块, 否则会导致烧录失败。)

本程序利用 Arduino Mega 2560 单片机控制 GPS 模块进行室外定位, 再利用 WiFi 模块与指定 IP 地址的 TCP 服务器建立连接, 并将定位得到的经度和纬度数据传输到该服务器。



```
GPS_Wifi | Arduino 1.0.6
File Edit Sketch Tools Help

GPS_Wifi

#define WifiSerial Serial2
#define GpsSerial Serial1
#define DebugSerial Serial
#define ssid "MyWifi"
#define pwd "00000000"
#define addr "192.168.1.100"
#define port 333

int L = 13; //LED指示灯引脚

Done Saving.
```

图 11.14 扩展实验程序

5.2 硬件配置及连线

实验中需要先检查测试 WiFi 模块，测试完成后将 WiFi 模块、GPS 模块及 Arduino Mega 2560 单片机连接在一起，具体步骤如下：

- (1) 使用母对母杜邦线，按图 11.15 所示方式，连接 WiFi 模块和 USB 转串口模块。

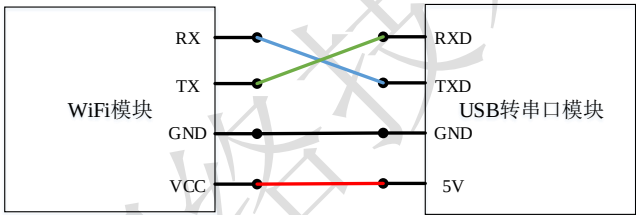


图 11.15 WiFi 模块连接 USB 转串口模块

- (2) 模块连接好之后，打开 SSCOM 串口调试助手，选择对应端口号并将波特率设为 9600，发送指令“AT”，若收到“OK”说明模块正常工作。读者可自行测试其它 AT 指令，可参考实验资源文件夹 WNTLab/experiments/exp11 中的 AT 指令集示例。

- (3) 测试完成后，拔掉 WiFi 模块与 USB 转串口模块的连线。并使用公对母杜邦线，按图 11.16 所示方式，连接 WiFi 模块、GPS 模块以及 Arduino Mega 2560 单片机模块。

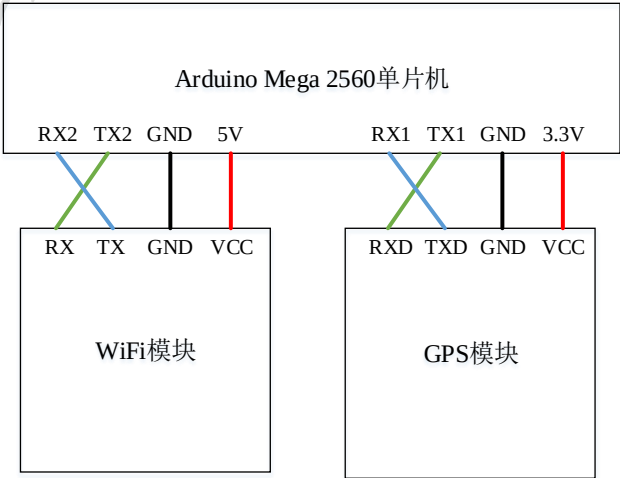


图 11.16 硬件连接原理图

5.3 测试实验结果

(1) 将电池连接在单机电源口进行供电, 如图 11.17 所示, 并将连接好的系统放置于室外(也可选择较开阔的窗台、阳台等场景)。

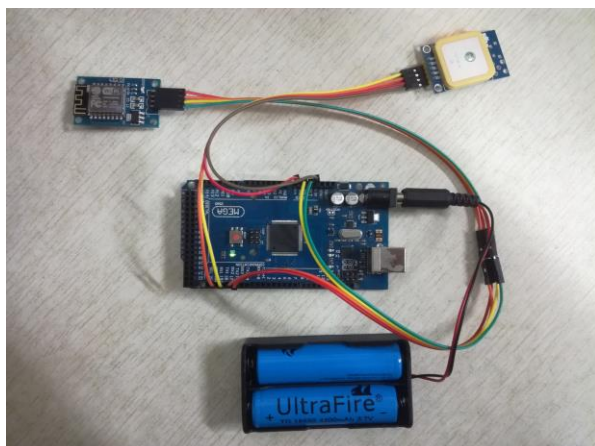


图 11.17 系统供电

(2) 观察实验结果: 观察 5.1 节中建立的 TCP 服务器, 如果系统成功运行, 则 PC 的网络调试助手中可以收到 WiFi 模块发送来的定位结果, 如图 11.18 所示。

同 4.3 节第(2)步所述方法转换定位结果为度分秒表示形式, 可知定位成功。



图 11.18 网络调试助手收到定位结果

6 结果分析与讨论

本实验通过 GPS 模块采集 GPS 数据, 由 WiFi 模块发送数据, 在 Arduino 单片机上实现整个过程, 包括处理 GPS 坐标以及对 WiFi 模块进行控制, 读者应当首先阅读源代码来理解程序的运行过程, 然后修改程序源代码来改变具体功能, 实现个性化, 并对模块、单片机进行进一步开发使用。

7 注意事项

烧录程序时应保证 Arduino Mega 2560 单片机的 RX0、TX0 引脚没有被占用。

使用 WiFi 模块时, 首先测试模块的 AT 指令有无错误来确认模块可以正常工作。

实验十二 AODV 和 DSR 协议仿真实验

1 实验要求与目的

- 1 人独立完成
- 理解 AODV 和 DSR 协议的工作过程
- 掌握 NS2 仿真分析 AODV 和 DSR 协议

2 实验背景与原理

2.1 实验背景

AODV 是应用最广泛的按需路由协议之一, 中间节点不需事先维护路由。AODV 中节点移动可能会导致原来路由不可用, 采用逐跳路由的方式转发分组, 并利用广播请求查找路由。整个通信过程对称, 路由可逆, 具体原理参考教材第 6 章。

DSR 路由协议应用也较多, 具体实现过程和原理参考教材第 6 章。

2.2 实验原理

基础实验: 分别对 AODV 和 DSR 路由协议进行仿真, 采用同一拓扑如图 12.1 所示, 其由 13 节点组成, 分别是节点 0~12。其中, 节点 8 是源节点, 节点 2 是目标节点。

扩展实验: 从 OpenStreetMap 中导出某地的地图, 并使用 SUMO 工具将该地图转换为 NS2 可读取的移动场景, 最后利用该场景中的所有节点构成网络拓扑, 对 AODV 和 DSR 协议进行仿真。

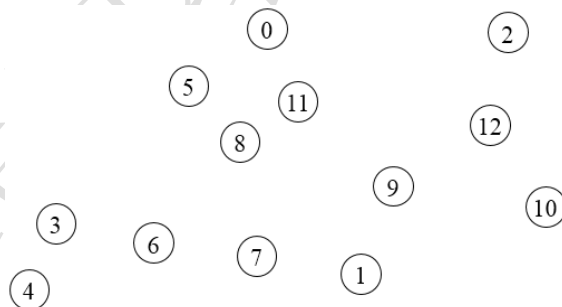


图 12.1 AODV 和 DSR 协议仿真拓扑

3 实验环境与资源

3.1 基础实验

(1) 环境资源:

树莓派套件、实验一中搭建的仿真实验平台、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;

- 本实验主要用到仿真实验平台中的 NS2 软件, 实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;

● 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp12/exp12_basic。

3.2 扩展实验

(1) 环境资源:

同 3.1 节

(2) 使用说明:

● 本实验需要用到道路交通仿真软件 SUMO, 该软件已集成在仿真实验平台中, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;

● 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp12/exp12_extend。

4 基础实验步骤与演示

4.1 AODV 仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 12.2 所示。



图 12.2 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本, 本实验的子文件夹为“exp12”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

```
cd WNTLab/experiments/exp12/exp12_basic/AODV
```

再输入以下命令运行本实验, 如图 12.3 所示。

```
ns AODV.tcl
```

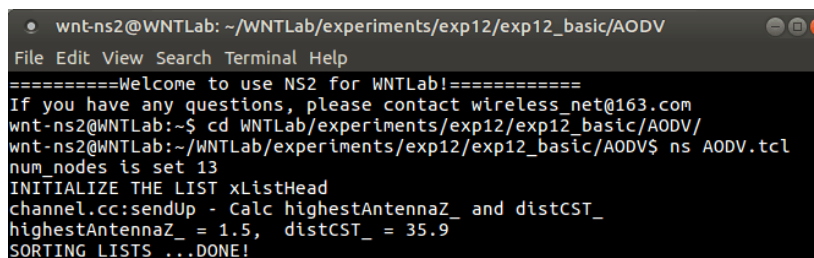


图 12.3 运行仿真实验

(4) 利用 NAM 动画观察实验结果, 如图 12.4 所示。在观看 NAM 动画时, 运行的步长调整为 2ms 即可。

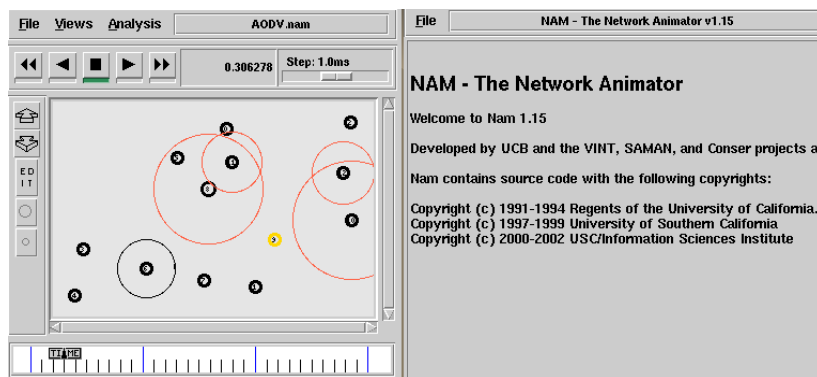


图 12.4 实验结果演示

程序运行说明: 仿真开始时(0s), 节点间未交换路由信息, 路由表尚未建立(AODV 的按需机制决定)。0.02s 时, 节点 8 开始向节点 2 发送数据, 注意, 发送数据前, 节点 8 需通过 RREQ/RREP 与节点 2 建立路由。进行到 0.25s 时, 节点 4 开始向节点 1 发送数据, 同样, 节点 4 需通过 RREQ/RREP 与节点 1 建立路由。在仿真过程(0.25~3s)中, 两条传输的数据流会出现断断续续的情况, 其反映 TCP 的拥塞控制。在 3s 时, 仿真实验结束。

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析, 并绘制图形对比。

4.2 DSR 仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

```
cd WNTLab/experiments/exp12/exp12_basic/DSR
```

再输入以下命令运行本实验, 如图 12.5 所示。

```
ns DSR.tcl
```

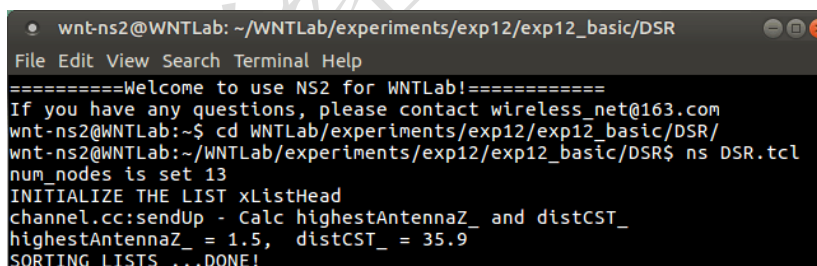


图 12.5 运行仿真实验

(2) 利用 NAM 动画观察实验结果, 如图 12.6 所示。观看 NAM 动画时, 运行的步长调整为 2ms 即可。

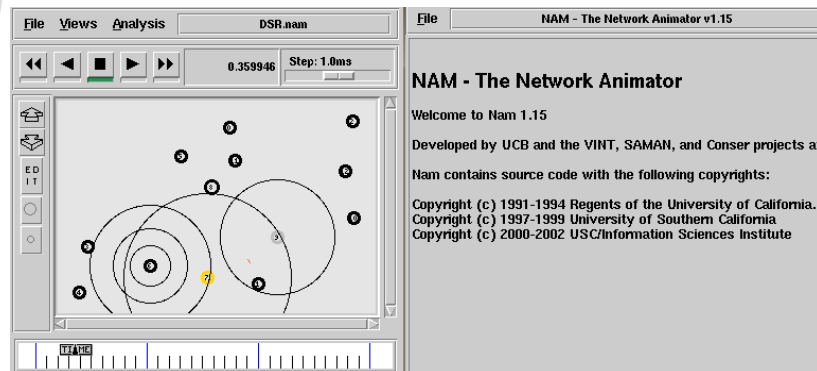


图 12.6 实验结果演示

程序运行说明：仿真开始时(0s)，节点间未交换路由信息，路由表尚未建立(DSR 的源路由决定)。到 0.02s 时，节点 8 开始向节点 2 发送数据，注意，发送数据前，节点 8 需通过 RREQ/RREP 与节点 2 建立路由。进行到 0.25s 时，节点 4 开始向节点 1 发送数据，同样节点 4 需通过 RREQ/RREP 与节点 1 建立路由。仿真过程(0.25~3s)中，两条传输的数据流会出现断断续续的情况，反映 TCP 的拥塞控制。在 3s 时，仿真实验结束。

如果读者想进一步分析仿真的深层次结果，可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析，并绘制图形对比。

5 扩展实验步骤与演示

为便于读者理解，下面对基础实验中的代码进行简单分析，并根据分析分别对 AODV 和 DSR 协议仿真进行修改与扩展。

5.1 基础实验代码分析

分析代码前，读者需要了解 OTcl 脚本语言。下面以基础实验中 AODV 协议仿真为例，给出具体代码及结构分析，具体代码细节及含义请见实验资源包 WNTLab/experiments/exp12/exp12_basic 中的脚本文件。

```

1.  #仿真变量设置
2.  set val(chan)           Channel/WirelessChannel
3.  set val(prop)           Propagation/TwoRayGround
4.  set val(netif)          Phy/WirelessPhy/802_15_4
5.  set val(mac)            Mac/802_15_4
6.  set val(ifq)            Queue/DropTail/PriQueue
7.  set val(ll)             LL
8.  set val(ant)            Antenna/OmniAntenna
9.  set val(ifqlen)         50
10. set val(nn)             13
11. set val(rp)             AODV
12. set val(x)              30
13. set val(y)              30
14. set val(traffic)        ftp
15. set Stime1              0.02
16. set Stime2              0.25
17. set End                 3
18. #ns实例化和trace文件设置
19. set ns_                  [new Simulator]
20. set tracefd              [open AODV.tr w]
21. $ns_ trace-all $tracefd
22. set namtrace              [open AODV.nam w]
23. $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
24. $ns_ puts-nam-traceall {# nam4wpan #}
25. Mac/802_15_4 wpanNam namStatus on
26. #仿真拓扑设置
27. set dist(15m) 8.54570e-07
28. Phy/WirelessPhy set CStresh_ $dist(15m)
29. Phy/WirelessPhy set RXThresh_ $dist(15m)
30. set topo                 [new Topography]
31. $topo load_flatgrid $val(x) $val(y)
32. set god_                 [create-god $val(nn)]
33. set chan_1_              [new $val(chan)]
34. #节点配置
35. $ns_ node-config -adhocRouting $val(rp) \
36. -llType $val(ll) \
37. -macType $val(mac) \
38. -ifqType $val(ifq) \
39. -ifqLen $val(ifqlen) \
40. -antType $val(ant) \
41. -propType $val(prop) \
42. -phyType $val(netif) \

```



```

43. -topoInstance $topo \
44. -agentTrace OFF \
45. -routerTrace OFF \
46. -macTrace ON \
47. -movementTrace OFF \
48. -channel $chan_1_
49. #节点实例化
50. for {set i 0} {$i < $val(nn)} {incr i} {
51.   set node_($i) [$ns_ node]
52.   $node_($i) random-motion 0
53. }
54. source ./AODV_topo.scn
55. #代理和数据流设置
56. proc ftppro { src dst starttime } {
57.   global ns_ node_
58.   set tcp($src) [new Agent/TCP]
59.   eval $tcp($src) set packetSize_ 60
60.   set sink($dst) [new Agent/TCPSink]
61.   eval $ns_ attach-agent $node_($src) $tcp($src)
62.   eval $ns_ attach-agent $node_($dst) $sink($dst)
63.   eval $ns_ connect $tcp($src) $sink($dst)
64.   set ftp($src) [new Application/FTP]
65.   eval $ftp($src) attach-agent $tcp($src)
66.   $ns_ at $starttime "$ftp($src) start"
67. }
68. set lowSpeed 0.20ms
69. set highSpeed 1.4ms
70. Mac/802_15_4 wpanNam PlaybackRate $lowSpeed
71. ftppro 8 2 $Stime1
72. ftppro 4 1 $Stime2
73. Mac/802_15_4 wpanNam FlowClr -p AODV -c tomato
74. Mac/802_15_4 wpanNam FlowClr -p tcp -s 8 -d 2 -c blue
75. Mac/802_15_4 wpanNam FlowClr -p ack -s 2 -d 8 -c blue
76. Mac/802_15_4 wpanNam FlowClr -p tcp -s 4 -d 1 -c green
77. Mac/802_15_4 wpanNam FlowClr -p ack -s 1 -d 4 -c green
78. #节点初始化
79. for {set i 0} {$i < $val(nn)} {incr i} {
80.   $ns_ initial_node_pos $node_($i) 2
81. }
82. for {set i 0} {$i < $val(nn)} {incr i} {
83.   $ns_ at $End "$node_($i) reset";
84. }
85. $ns_ at $End "stop"
86. $ns_ at $End "$ns_ halt"
87. #仿真完成处理函数
88. proc stop {} {
89.   global ns_ tracefd namtrace
90.   $ns_ flush-trace
91.   close $tracefd
92.   close $namtrace
93.   exec nam AODV.nam &
94.   exit 0
95. }
96. #启动仿真
97. $ns_ run

```

第 1~17 行, 对仿真中需要用到变量进行设置, 该部分代码仅为便于修改(多处采用相同模块时仅需修改变量即可), 并未增加任何功能;

第 18~25 行, 实例化 ns 变量, 同时设置仿真过程中的 trace 和 nam 输出文件, 以便对实验结果进行分析和演示;

第 26~33 行, 对仿真实验的拓扑范围和全局节点记录进行设置, 以便控制节点位置和运动范围;

第 34~48 行, 在节点实例化前对参数进行配置, 如路由协议、天线模型、物理层模块等,

实现正确有效的隐藏节点实验分析:

第 49~54 行, 对节点进行实例化, 调用了 AODV_topo.scn 脚本中对节点的配置;

第 55~77 行, 建立源节点到目的节点的 TCP 连接, 建议一条 FTP 数据流, 并且设置实验开始时间;

第 78~86 行, 对节点大小进行设置, 用于 nam 演示, 同时为实验结束时的节点进行 reset 事件设置, 并且设置实验结束时间;

第 87~95 行, 对仿真结束时的后续处理函数(stop)功能定义, 包括关闭打开的文件、启动演示动画等, 读者可在其基础上自行扩展;

第 96~97 行, 启动 ns 运行仿真。

5.2 实验扩展

本实验将从 OpenStreetMap 中导出某城市地图, 并使用 SUMO 工具将该地图转换为 NS2 可读取的移动场景, 最后利用该场景中的所有节点构成网络拓扑, 对 AODV 和 DSR 协议进行仿真。下面以 AODV 为例介绍实验步骤。

(1) 修改代码。具体代码请参考脚本文件 WNTLab/experiments/exp12/exp12_extend/

AODV.tcl, 下面给出主要修改部分:

```
1. source ./AODV_topo.scn
2. for {set i 0} {$i < 6} {incr i 2} {
3.   set j [expr $i+1]
4.   ftppro $i $j $Stime1+($i*0.05)
5. }
```

其中第1行修改了节点的来源文件, 对应5.1节中第54行; 第3~6行将ftp流定向发送修改为互相发送, 防止路由节点过多时距离太远引起实验效果较差, 对应5.1节中第71、72行。

(2) 导出地图。从 OpenStreetMap 中导出任一地区的地图, 步骤如下:

a. 打开浏览器, 进入网站 www.openstreetmap.org;

b. 进入 Open Street 主页后, 在左上角的搜索框中搜索指定位置的区域, 并单击顶部的“Export”按钮, 如图 12.7 和 12.8 所示;

c. 在左侧选择“手动选择不同的区域”, 并在右侧选定需导出的地图区域, 然后单击左侧“Export”按钮导出, 默认文件 map.osm.xml。本实验将该文件重命名为: test.osm.xml, 并将该文件剪切至/home/wnt-ns2/WNTLab/experiments/exp12/exp12_extend/AODV 目录下

(3) 产生移动场景。依次在 Terminal 中执行以下命令:

```
a. netconvert --osm-files test.osm.xml -o test.net.xml --remove-edges.by-type "highway.service"
b. polyconvert --osm-files test.osm.xml --net-file test.net.xml --type-file osmPolyconvert.typ.xml -o test.poly.xml
c. python $SUMO_HOME/tools/randomTrips.py -n test.net.xml -r test.rou.xml -e 100 -l
```

(4) 获取可视化地图。在 Terminal 中执行命令:

```
sumo-gui test.sumo.cfg
```

(5) 将地图数据导出为 NS2 可识别的场景文件, 依次在 Terminal 中执行以下命令:

```
a. sumo -c test.sumo.cfg --fcd-output test.sumo.xml
b. python $SUMO_HOME/tools/traceExporter.py --fcd-input test.sumo.xml --ns2config-output ctest.tcl --ns2activity-output activity.tcl --ns2mobility-output mobility.tcl
```

(6) 经过上述步骤, 可得到一个关键的文件: mobility.tcl。该文件为移动场景, 将之重命名为 AODV_mobility.tcl 即可。

(7) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

```
cd WNTLab/experiments/exp12/exp12_extend/AODV
```

再输入以下命令运行本实验, 如图 10.7 所示。

```
ns AODV.tcl
```

```

wnt-ns2@WNTLab: ~/WNTLab/experiments/exp12/exp12_extend/AODV
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp12/exp12_extend/AODV/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp12/exp12_extend/AODV$ ns AODV.tcl
num_nodes is set 61
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 35.9
SORTING LISTS ...DONE!

```

图 12.7 运行仿真实验

(2) 利用 NAM 动画观察实验结果, 如图 12.8 所示。观看 NAM 动画时, 运行的步长调整为 2ms 即可。

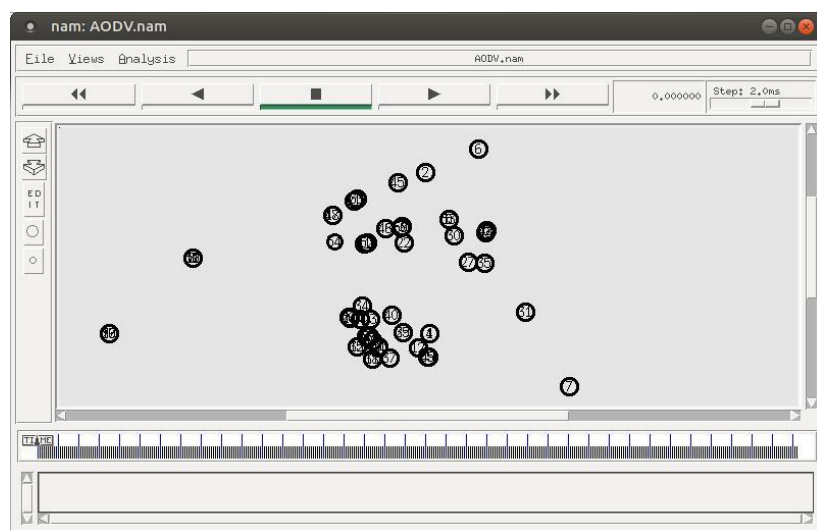


图 12.8 实验结果演示

6 结果分析与讨论

AODV 和 DSR 是自组网路由的代表, 受到广泛的研究和应用。本实验的结果分析和代码说明详见教材第 6 章, 同时在仿真分析过程中还需要注意以下两个问题:

- (1) AODV 和 DSR 协议如何进行路由发现和路由建立?
- (2) AODV 和 DSR 与其它自组网路由协议相比有何不同?

7 注意事项

仿真主要涉及相关的仿真脚本, 无需修改仿真模块, 具体代码参见电子资源中的实验源代码。

实验十三 DD 和 S-MAC 协议仿真实验

1 实验要求和目的

- 1 人独立完成
- 了解定向扩散路由和 S-MAC 协议原理
- 掌握 NS2 分析定向扩散和 S-MAC 协议

2 实验背景与原理

2.1 实验背景

定向扩散(Directed Diffusion, DD)是以数据为中心、查询驱动的 WSN 经典路由协议。有关工作原理详见教材第 7 章。

Sensor MAC(S-MAC)是争用型的 MAC 层协议, 详细内容请参考教材第 7 章。

2.2 实验原理

DD 协议实验定义了一个大小为 670*670 的仿真拓扑, 如图 13.1 所示, 设置了 10 个无线传感器节点, 并配置了 DD 协议。S-MAC 协议实验定义了一个含 14 个 WSN 节点的仿真拓扑, 如图 13.2 所示。

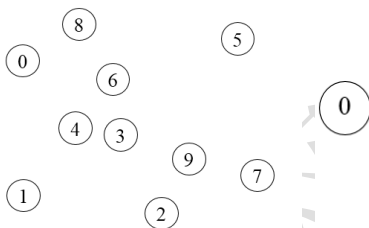


图 13.1 DD 协议仿真拓扑

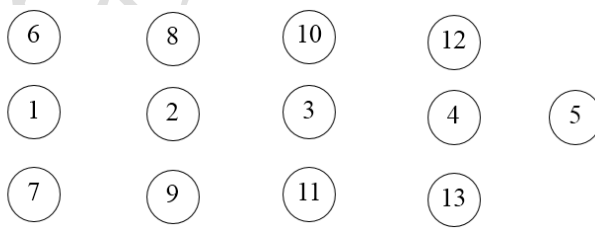


图 13.2 S-MAC 协议仿真拓扑

3 实验环境与资源

3.1 基础实验

(1) 环境资源:

树莓派套件、实验一中搭建的仿真实验平台、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;
- 本实验主要用到仿真实验平台中的 NS2 软件, 实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;
- 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp13/exp13_basic。

3.2 扩展实验

(1) 环境资源:

同 3.1 节

(2) 使用说明:

● 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp13/exp13_extend。

4 基础实验步骤与演示

4.1 DD 协议仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一之 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 13.3 所示。



图 13.3 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本。本实验的子文件夹为“exp13”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

cd WNTLab/experiments/exp13/exp13_basic/DD

再输入以下命令运行本实验, 如图 13.4 所示。

ns DD.tcl

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp13/exp13_basic/DD
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp13/exp13_basic/DD/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp13/exp13_basic/DD$ ns DD.tcl
num_nodes is set 10
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
Loading connection pattern...
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 550.0
SORTING LISTS ...DONE!
```

图 13.4 运行仿真实验

(4) 利用 NAM 动画观察实验结果, 如图 13.5 所示。在观看 NAM 动画时, 运行的步长调整为 2ms 即可。

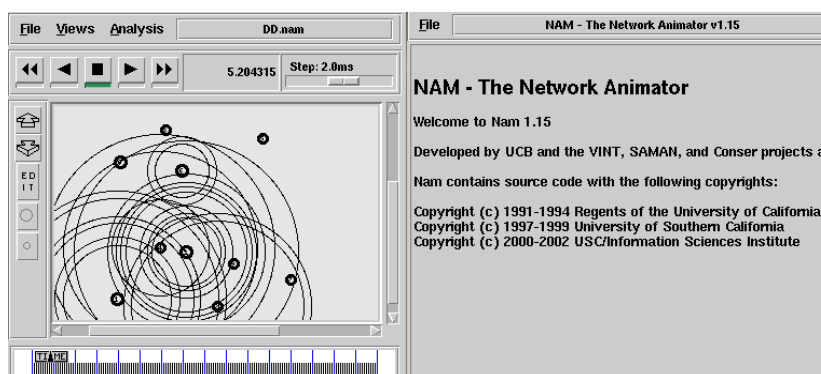


图 13.5 实验结果演示

程序运行说明: 仿真开始时, 节点之间未交换路由信息, 此时无数据传输。在 0.12s 时, 节点 2 的 Ping 发送者进行 publish 操作, 并通过 DD 协议进行兴趣扩散。在 1.15s 时, 节点 9 的 Ping 接收者进行 subscribe 操作; 2.30s 时, 节点 8 的 Ping 接收者进行 subscribe 操作; 3.45s 时, 节点 7 的 Ping 接收者进行 subscribe 操作; 4.6s 时, 节点 6 的 Ping 接收者进行 subscribe 操作。100s 时, 仿真实验结束。

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件 (.tr) 进行分析, 并绘制图形对比。

4.2 S-MAC 协议仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

```
cd WNTLab/experiments/exp13/exp13_basic/S-MAC
```

再输入以下命令运行本实验, 如图 13.6 所示。

```
ns SMAC.tcl
```

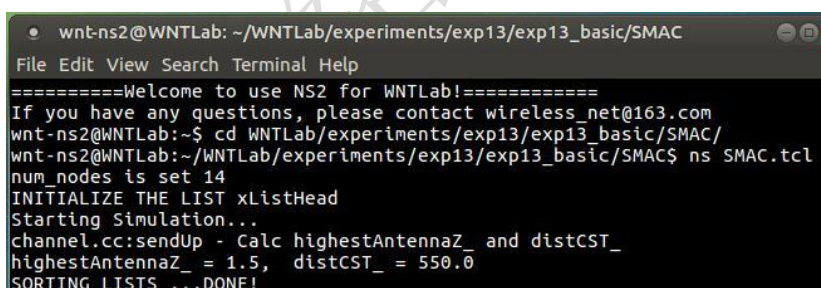


图 13.6 运行仿真实验

(2) 利用 NAM 动画观察实验结果, 如图 13.7 所示。观看 NAM 动画时, 运行的步长调整为 2ms 为宜。

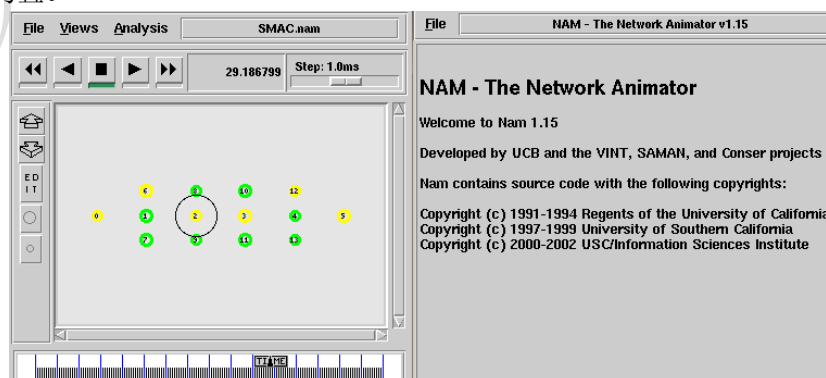


图 13.7 实验结果演示

程序运行说明: 仿真开始时, 节点之间未建立路由, 此时无数据传输, 所有节点为绿色。从 1s 开始, 节点 0 与节点 5 建立 UDP/CBR 的数据传输。需注意, 在传输数据前, 节点 0 先要通过 RREQ/RREP 建立到节点 5 的路由。数据传输过程中, 由于节点能量消耗, 较高负载的节点会出现颜色变化(如 21s 左右, 节点 0、6、2、3、12、5 开始变黄)。仿真时间到达 100s 时, UDP/CBR 流的传输结束, 同时, 仿真实验结束。

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析, 并绘制图形对比。

5 扩展实验步骤与演示

为便于读者理解, 下面对基础实验中的代码进行简单分析, 并根据分析分别对 AODV 和 DSR 协议仿真进行修改与扩展。

5.1 基础实验代码分析

分析代码前, 读者需要了解 OTcl 脚本语言。下面以基础实验中 DD 协议仿真为例, 给出具体代码及结构分析, 具体代码细节及含义请见实验资源包 WNTLab/experiments/exp13/exp13_basic 中的脚本文件。

```

1.  #仿真变量设置
2.  set opt(chan)          Channel/WirelessChannel
3.  set opt(prop)          Propagation/TwoRayGround
4.  set opt(netif)         Phy/WirelessPhy
5.  set opt(mac)           Mac/802_11
6.  set opt(ifq)           Queue/DropTail/PriQueue
7.  set opt(ll)            LL
8.  set opt(ant)           Antenna/OmniAntenna
9.  set opt(filters)       GradientFilter
10. set opt(x)             670
11. set opt(y)             670
12. set opt(ifqlen)        50
13. set opt(nn)            10
14. set opt(sndr)           1
15. set opt(rcvr)          4
16. set opt(seed)          0.0
17. set opt(stop)          100
18. set opt(tr)            "DD.tr"
19. set opt(nam)           "DD.nam"
20. set opt(adhocRouting)   Directed_Diffusion
21. #仿真模块参数设置
22. LL set mindelay_        50us
23. LL set delay_          25us
24. LL set bandwidth_       0
25. Queue/DropTail/PriQueue set Prefer_Routing_Protocols 1
26. Antenna/OmniAntenna set X_ 0
27. Antenna/OmniAntenna set Y_ 0
28. Antenna/OmniAntenna set Z_ 1.5
29. Antenna/OmniAntenna set Gt_ 1.0
30. Antenna/OmniAntenna set Gr_ 1.0
31. Phy/WirelessPhy set CPTresh_ 10.0
32. Phy/WirelessPhy set CSTresh_ 1.559e-11
33. Phy/WirelessPhy set RXThresh_ 3.652e-10
34. Phy/WirelessPhy set Rb_ 2*1e6
35. Phy/WirelessPhy set Pt_ 0.2818
36. Phy/WirelessPhy set freq_ 914e+6
37. Phy/WirelessPhy set L_ 1.0
38. #ns实例化、trace文件设置及仿真拓扑设置
39. set ns_ [new Simulator]
40. set topo [new Topography]
41. set tracefd [open $opt(tr) w]
42. $ns_ trace-all $tracefd

```



```

43. set nf [open $opt(nam) w]
44. $ns_ namtrace-all-wireless $nf $opt(x) $opt(y)
45. $topo load_flatgrid $opt(x) $opt(y)
46. set god_ [create-god $opt(nn)]
47. #节点配置
48. $ns_ node-config -adhocRouting $opt(adhocRouting) \
49. -llType $opt(ll) \
50. -macType $opt(mac) \
51. -ifqType $opt(ifq) \
52. -ifqLen $opt(ifqlen) \
53. -antType $opt(ant) \
54. -propType $opt(prop) \
55. -phyType $opt(netif) \
56. -channelType $opt(chan) \
57. -topoInstance $topo \
58. -diffusionFilter $opt(filters) \
59. -agentTrace ON \
60. -routerTrace ON \
61. -macTrace ON
62. #节点实例化
63. for {set i 0} {$i < $opt(nn)} {incr i} {
64. set node_($i) [$ns_ node $i]
65. $node_($i) color black
66. $node_($i) random-motion 0
67. $god_ new_node $node_($i)
68. }
69. puts "Loading connection pattern..."
70. #节点位置设置
71. $node_(0) set X_ 18
72. $node_(0) set Y_ 331
73. $node_(0) set Z_ 0
74. $node_(1) set X_ 11
75. $node_(1) set Y_ 36
76. $node_(1) set Z_ 0
77. $node_(2) set X_ 224
78. $node_(2) set Y_ 20
79. $node_(2) set Z_ 0
80. $node_(3) set X_ 158
81. $node_(3) set Y_ 139
82. $node_(3) set Z_ 0
83. $node_(4) set X_ 101
84. $node_(4) set Y_ 147
85. $node_(4) set Z_ 0
86. $node_(5) set X_ 321
87. $node_(5) set Y_ 382
88. $node_(5) set Z_ 0
89. $node_(6) set X_ 149
90. $node_(6) set Y_ 314
91. $node_(6) set Z_ 0
92. $node_(7) set X_ 381
93. $node_(7) set Y_ 78
94. $node_(7) set Z_ 0
95. $node_(8) set X_ 113
96. $node_(8) set Y_ 400
97. $node_(8) set Z_ 0
98. $node_(9) set X_ 258
99. $node_(9) set Y_ 113
100. $node_(9) set Z_ 0
101. #节点初始化
102. for {set i 0} {$i < $opt(sndr)} {incr i} {
103. set src_($i) [new Application/DiffApp/PingSender/TPP]
104. $ns_ attach-diffapp $node_([expr $i + 2]) $src_($i)
105. $ns_ at [expr 0.12 * [expr 1+$i]] "$src_($i) publish"
106. }
107. for {set i 0} {$i < $opt(rcvr)} {incr i} {
108. set snk_($i) [new Application/DiffApp/PingReceiver/TPP]

```

```

109. $ns_ attach-diffapp $node_([expr $opt(nn)-1-$i]) $snk_($i)
110. $ns_ at [expr 1.15*[expr 1+$i]] "$snk_($i) subscribe"
111. }
112. for {set i 0} {$i < $opt(nn)} {incr i} {
113. $ns_ initial_node_pos $node_($i) 20
114. }
115. for {set i 0} {$i < $opt(nn)} {incr i} {
116. $ns_ at $opt(stop).1 "$node_($i) reset";
117. }
118. #仿真完成处理函数
119. proc finish {} {
120. global ns_ tracefd nf
121. $ns_ flush-trace
122. close $tracefd
123. close $nf
124. exec nam DD.nam &
125. exit 0
126. }
127. #仿真结束
128. $ns_ at $opt(stop) "finish"
129. $ns_ at $opt(stop) "$ns_ nam-end-wireless $opt(stop)"
130. $ns_ at $opt(stop).01 "puts \"NS EXITING...\"; $ns_ halt"
131. puts "Starting Simulation..."
132. #启动仿真
133. $ns_ run

```

第 1~20 行, 对仿真中需用到的变量进行设置, 该部分代码仅为便于修改(多处采用相同模块时仅需修改变量即可), 并未增加任何功能;

第 21~37 行, 对仿真中需用到的模块参数进行设置, 需注意, 此处并未给出所有参数的设置, 仅对必要的参数进行修改, 若未给出则意味采用 NS2 中的默认值;

第 38~46 行, 实例化 ns 变量, 同时设置仿真过程中的 trace 和 nam 输出文件, 以便对实验结果进行分析和演示; 对仿真实验的拓扑范围和全局节点记录进行设置, 以便控制节点位置和运动范围;

第 47~61 行, 在节点实例化前对参数进行配置, 如路由协议、天线模型、物理层模块等, 实现正确有效的隐藏节点实验分析;

第 62~69 行, 所有节点具有相同配置, 如需更改节点配置, 需在实例化节点前重新用 node-config 进行配置;

第 70~100 行, 对实例化的节点进行位置的设定, 注意, 位置不能超过拓扑范围;

第 101~117 行, 对节点大小进行初始化设置;

第 118~126 行, 对仿真结束时的后续处理函数(finish)功能定义, 主要有关闭打开的文件、启动演示动画等, 读者可在其基础上自行扩展;

第 127~131 行, 仿真结束事件的设置, 此处调用了之前定义的 finish 函数, 同时输出提示信息, 表明仿真实验的结束并停机;

第132~133行, 启动ns运行仿真。

5.2 实验扩展

读者在理解 5.1 节的基础上, 可自行修改脚本代码, 下面以 DD 协议仿真为例给出修改案例以供参考。

(1) 修改代码。具体代码请参考脚本文件 WNTLab/experiments/exp13/exp13_extend/DD/DD.tcl, 下面给出主要修改部分:

```

1. set opt(nn)      12
2. set opt(sndr)    2
3. set opt(rcvr)    5
4. $node_(10) set   X_ 500
5. $node_(10) set   Y_ 133

```

```

6. $node_(10) set      Z_ 0
7. $node_(11) set      X_ 369
8. $node_(11) set      Y_ 233
9. $node_(11) set      Z_ 0

```

其中第1~3行修改了节点数量, 对应5.1节中第13~15行; 第4~9行为新增的节点定义位置, 对应5.1节中第70~100行。

(2) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

cd WNTLab/experiments/exp13/exp13_basic/DD

再输入以下命令运行本实验, 如图 13.8 所示。

ns DD.tcl

```

wnt-ns2@WNTLab: ~/WNTLab/experiments/exp13/exp13_extend/DD
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp13/exp13_extend/DD/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp13/exp13_extend/DD$ ns DD.tcl
num_nodes is set 12
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
Loading connection pattern...
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 550.0
SORTING LISTS ...DONE!

```

图13.8 运行仿真实验

(3) 利用 NAM 动画观察实验结果, 如图 13.9 所示。在观看 NAM 动画时, 运行的步长调整为 2ms 为宜。

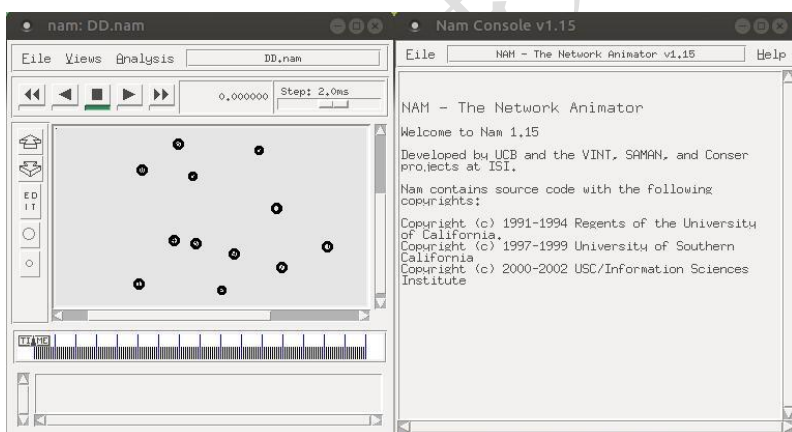


图13.9 实验结果演示

6 结果分析与讨论

有关本实验的原理、结果分析详见教材第7章, 仿真过程中还需注意以下问题:

- (1) DD 路由协议如何进行路由建立?
- (2) DD 路由协议存在哪些问题? 如何解决?
- (3) SMAC 如何主要为解决什么问题设计的? 为什么不能直接采用 802.11 协议?
- (4) SMAC 和其它同类协议的性能比较有何不同?

7 注意事项

仿真主要涉及相关脚本, 无需修改仿真模块, 具体代码参见电子资源中的实验源代码。

实验十四 水下无线传感网协议仿真实验

1 实验要求与目的

- 1人独立完成
- 了解水下无线 WSN 的基本概念
- 掌握水下无线 WSN 协议的仿真

2 实验背景与原理

2.1 实验背景

水下 WSN(UWSN)的工作原理详见教材 7.6 节。本实验采用康涅狄格大学 UWSN 实验室开发的实验平台(包括有关物理层、MAC 层和网络层协议), 仿真分析传输性能, MAC 协议为 Broadcast MAC, 路由为 VBF 协议, 数据包的发送速率为 1packet/s。

基于向量的转发(Vector-Based Forwarding, VBF)路由协议的理论知识详见教材第 7.6 节。

2.2 实验原理

实验设置了 6 个节点, 每个节点的 MAC 和路由协议分别为 Broadcast MAC 和 VBF, 拓扑如图 14.1 所示。

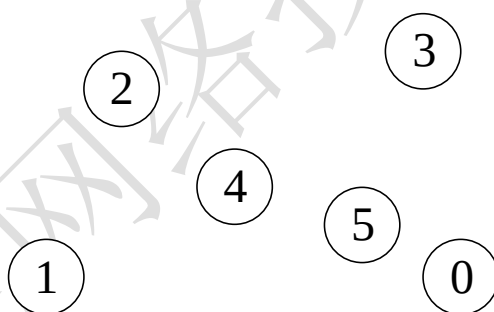


图 14.1 水下传感网仿真拓扑

3 实验环境与资源

(1) 环境资源:

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;
- 本实验主要用到仿真实验平台中的 NS2 软件, 官方 NS2 版本中未包含本实验所需模块。实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;
- 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp14。

4 基础实验步骤与演示

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 14.2 所示。



图 14.2 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本。本实验的子文件夹为“exp14”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

```
cd WNTLab/experiments/exp14/exp14_basic
```

再输入以下命令运行本实验, 如图 14.3 所示。

```
ns VBF.tcl
```

```
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp14/exp14_basic/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp14/exp14_basic$ ns VBF.tcl
```

图 14.3 运行仿真实验

(4) 实验结果如图 14.4 所示, 可观察到各个节点间数据包传输情况。

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp14
File Edit View Search Terminal Help
underwater: the packet is interfered at node 0
underwater: the packet is interfered at node 0
underwater: the packet is interfered at node 3
underwater: the packet is interfered at node 1
underwater: the packet is interfered at node 2
underwater: the packet is interfered at node 3
UWSink (id:0): I get the packet data no.7 from 1
SK 0: Num_Recv 36, InterArrival 8.011716
UWSink (id:0): I get the packet data no.7 from 2
SK 0: Num_Recv 37, InterArrival 0.507140
UWSink (id:0): I get the packet data no.7 from 3
SK 0: Num_Recv 38, InterArrival 0.499526
UWSink (id:0): I get the packet data no.7 from 4
SK 0: Num_Recv 39, InterArrival 0.484603
UWSink (id:0): I get the packet data no.7 from 5
```

图 14.4 实验结果演示

5 扩展实验步骤与演示

基础实验对水下载感网中 VBF 路由协议进行简单仿真, 读者可从仿真结果观察 6 个节点间数据包的传输情况。扩展实验中将从能耗这一角度对 VBF 路由协议深入分析。

5.1 基础实验代码分析

分析代码前, 读者需了解 OTcl 脚本语言。下面给出基础仿真实验的具体代码, 并且给

出代码结构分析, 具体代码细节及含义请见实验资源包 WNTLab/experiments/exp14/exp14_basic 中的脚本文件。

```

1.  #仿真变量设置
2.  set opt(chan)          Channel/UnderwaterChannel
3.  set opt(prop)          Propagation/UnderwaterPropagation
4.  set opt(netif)         Phy/UnderwaterPhy
5.  set opt(mac)           Mac/UnderwaterMac/BroadcastMac
6.  set opt(ifq)           Queue/DropTail/PriQueue
7.  set opt(ll)            LL
8.  set opt(energy)        EnergyModel
9.  set opt(txpower)       2.0
10. set opt(rxpower)       0.75
11. set opt(initialenergy) 10000
12. set opt(idlepower)     0.008
13. set opt(ant)           Antenna/OmniAntenna
14. set opt(filters)       GradientFilter
15. set opt(minspeed)      0
16. set opt(maxspeed)      3
17. set opt(speed)         0.5
18. set opt(position_update_interval) 0.3
19. set opt(packet_size) 50
20. set opt(routing_control_packet_size) 20
21. set opt(ifqlen)        50
22. set opt(nn)            6
23. set opt(x)             1000
24. set opt(y)             10
25. set opt(z)             10
26. set opt(seed)          11
27. set opt(stop)          500
28. set opt(prestop)       90
29. set opt(tr)            "VBF.tr"
30. set opt(datafile)      "VBF.data"
31. set opt(nam)           "VBF.nam"
32. set opt(adhocRouting)  Vectorbasedforward
33. set opt(width)         100
34. set opt(interval)      10.0
35. set opt(range)         120
36. if { $argc > 0 } {
37. set opt(seed) [lindex $argv 0]
38. set opt(nn) [lindex $argv 1]
39. set opt(datafile) [lindex $argv 2]
40. }
41. puts "the file name is $opt(datafile)"
42. puts "the sending interval is $opt(interval)"
43. #仿真模块参数设置
44. LL set mindelay_       50us
45. LL set delay_          25us
46. LL set bandwidth_      0
47. Queue/DropTail/PriQueue set Prefer_Routing_Protocols 1
48. Antenna/OmniAntenna set X_ 0
49. Antenna/OmniAntenna set Y_ 0
50. Antenna/OmniAntenna set Z_ 0.05
51. Antenna/OmniAntenna set Gt_ 1.0
52. Antenna/OmniAntenna set Gr_ 1.0
53. Agent/Vectorbasedforward set hop_by_hop_ 0
54. Mac/UnderwaterMac set bit_rate_ 1.0e4
55. Mac/UnderwaterMac set encoding_efficiency_ 1
56. Mac/UnderwaterMac/BroadcastMac set packetheader_size_ 0
57. Phy/UnderwaterPhy set CPTthresh_ 10
58. Phy/UnderwaterPhy set CSTthresh_ 0
59. Phy/UnderwaterPhy set RXTthresh_ 0
60. Phy/UnderwaterPhy set Pt_ 0.2818
61. Phy/UnderwaterPhy set freq_ 25
62. Phy/UnderwaterPhy set K_ 2.0
63. #ns实例化、trace文件设置及仿真拓扑设置

```



```

64. set ns_ [new Simulator]
65. set topo [new Topography]
66. $topo load_cubicgrid $opt(x) $opt(y) $opt(z)
67. set tracefd [open $opt(tr) w]
68. $ns_ trace-all $tracefd
69. set nf [open $opt(nam) w]
70. $ns_ namtrace-all-wireless $nf $opt(x) $opt(y)
71. set data [open $opt(datafile) a]
72. set total_number [expr $opt(nn)-1]
73. set god_ [create-god $opt(nn)]
74. $ns_ at 0.0 "$god_ set_filename $opt(datafile)"
75. set chan_1_ [new $opt(chan)]
76. global defaultRNG
77. $defaultRNG seed $opt(seed)
78. $ns_ node-config -adhocRouting $opt(adhocRouting) \
79. -llType $opt(ll) \
80. -macType $opt(mac) \
81. -ifqType $opt(ifq) \
82. -ifqLen $opt(ifqlen) \
83. -antType $opt(ant) \
84. -propType $opt(prop) \
85. -phyType $opt(netif) \
86. -agentTrace OFF \
87. -routerTrace OFF \
88. -macTrace ON \
89. -topoInstance $topo \
90. -energyModel $opt(energy) \
91. -txPower $opt(txpower) \
92. -rxPower $opt(rxpower) \
93. -initialEnergy $opt(initialenergy) \
94. -idlePower $opt(idlepower) \
95. -channel $chan_1_
96. puts "Width=$opt(width)"
97. #节点设置
98. set node_(0) [ $ns_ node 0]
99. $node_(0) set sinkStatus_ 1
100. $god_ new_node $node_(0)
101. $node_(0) set X_ 500
102. $node_(0) set Y_ 0
103. $node_(0) set Z_ 0
104. $node_(0) set passive 1
105. set rt [$node_(0) set ragent_]
106. $rt set control_packet_size $opt(routing_control_packet_size)
107. set a_(0) [new Agent/UWSink]
108. $ns_ attach-agent $node_(0) $a_(0)
109. $a_(0) attach-vectorbasedforward $opt(width)
110. $a_(0) cmd set-range $opt(range)
111. $a_(0) cmd set-target-x -20
112. $a_(0) cmd set-target-y -10
113. $a_(0) cmd set-target-z -20
114. $a_(0) cmd set-filename $opt(datafile)
115. $a_(0) cmd set-packetsize $opt(packet_size)
116. set node_(1) [ $ns_ node 1]
117. $node_(1) set sinkStatus_ 1
118. $god_ new_node $node_(1)
119. $node_(1) set X_ 440
120. $node_(1) set Y_ 0
121. $node_(1) set Z_ 0
122. $node_(1) set passive 1
123. set rt [$node_(1) set ragent_]
124. $rt set control_packet_size $opt(routing_control_packet_size)
125. $node_(1) set max_speed $opt(maxspeed)
126. $node_(1) set min_speed $opt(minspeed)
127. $node_(1) set position_update_interval_ $opt(position_update_interval)
128. set a_(1) [new Agent/UWSink]
129. $ns_ attach-agent $node_(1) $a_(1)

```

```

130. $a_1 attach-vectorbasedforward $opt(width)
131. $a_1 cmd set-range $opt(range)
132. $a_1 cmd set-target-x -20
133. $a_1 cmd set-target-y -10
134. $a_1 cmd set-target-z -20
135. $a_1 cmd set-filename $opt(datafile)
136. $a_1 cmd set-packetsize $opt(packet_size)
137. set node_2 [ $ns_ node 2]
138. $node_2 set sinkStatus_ 1
139. $node_2 random-motion 1
140. $node_2 set max_speed $opt(maxspeed)
141. $node_2 set min_speed $opt(minspeed)
142. $node_2 set position_update_interval_ $opt(position_update_interval)
143. $god_new_node $node_2
144. $node_2 set X_ 450
145. $node_2 set Y_ 50
146. $node_2 set Z_ 0
147. $node_2 set passive 1
148. set rt [$node_2 set ragent_]
149. $rt set control_packet_size $opt(routing_control_packet_size)
150. set a_2 [new Agent/UWSink]
151. $ns_ attach-agent $node_2 $a_2
152. $a_2 attach-vectorbasedforward $opt(width)
153. $a_2 cmd set-range $opt(range)
154. $a_2 cmd set-target-x -20
155. $a_2 cmd set-target-y -10
156. $a_2 cmd set-target-z -20
157. $a_2 cmd set-filename $opt(datafile)
158. $a_2 cmd set-packetsize $opt(packet_size)
159. set node_3 [ $ns_ node 3]
160. $node_3 set sinkStatus_ 1
161. $node_3 random-motion 1
162. $node_3 set max_speed $opt(maxspeed)
163. $node_3 set min_speed $opt(minspeed)
164. $node_3 set position_update_interval_ $opt(position_update_interval)
165. $god_new_node $node_3
166. $node_3 set X_ 500
167. $node_3 set Y_ 70
168. $node_3 set Z_ 0
169. $node_3 set passive 1
170. set rt [$node_3 set ragent_]
171. $rt set control_packet_size $opt(routing_control_packet_size)
172. set a_3 [new Agent/UWSink]
173. $ns_ attach-agent $node_3 $a_3
174. $a_3 attach-vectorbasedforward $opt(width)
175. $a_3 cmd set-range $opt(range)
176. $a_3 cmd set-target-x -20
177. $a_3 cmd set-target-y -10
178. $a_3 cmd set-target-z -20
179. $a_3 cmd set-filename $opt(datafile)
180. $a_3 cmd set-packetsize $opt(packet_size)
181. set node_4 [ $ns_ node 4]
182. $node_4 set sinkStatus_ 1
183. $node_4 random-motion 1
184. $node_4 set max_speed $opt(maxspeed)
185. $node_4 set min_speed $opt(minspeed)
186. $node_4 set position_update_interval_ $opt(position_update_interval)
187. $god_new_node $node_4
188. $node_4 set X_ 470
189. $node_4 set Y_ 20
190. $node_4 set Z_ 30
191. $node_4 set passive 1
192. set rt [$node_4 set ragent_]
193. $rt set control_packet_size $opt(routing_control_packet_size)
194. set a_4 [new Agent/UWSink]
195. $ns_ attach-agent $node_4 $a_4
196. $a_4 attach-vectorbasedforward $opt(width)

```

```

197. $a_(4) cmd set-range $opt(range)
198. $a_(4) cmd set-target-x -20
199. $a_(4) cmd set-target-y -10
200. $a_(4) cmd set-target-z -20
201. $a_(4) cmd set-filename $opt(datafile)
202. $a_(4) cmd set-packetsize $opt(packet_size)
203. set node_($total_number) [$ns_ node $total_number]
204. $god_new_node $node_($total_number)
205. $node_($total_number) set sinkStatus_ 1
206. $node_($total_number) set X_ 490
207. $node_($total_number) set Y_ 10
208. $node_($total_number) set Z_ 40
209. $node_($total_number) set-cx 200
210. $node_($total_number) set-cy 0
211. $node_($total_number) set-cz 0
212. set rt [$node_($total_number) set ragent_]
213. $rt set control_packet_size $opt(routing_control_packet_size)
214. set a_($total_number) [new Agent/UWSink]
215. $ns_ attach-agent $node_($total_number) $a_($total_number)
216. $a_($total_number) attach-vectorbasedforward $opt(width)
217. $a_($total_number) cmd set-range $opt(range)
218. $a_($total_number) cmd set-target-x 500
219. $a_($total_number) cmd set-target-y 0
220. $a_($total_number) cmd set-target-z 0
221. $a_($total_number) cmd set-filename $opt(datafile)
222. $a_($total_number) cmd set-packetsize $opt(packet_size)
223. #节点初始化
224. set start_time 1.33
225. for { set i 1 } { $i<$opt(nn) } { incr i } {
226. $a_($i) set data_rate_ [expr 1.0/$opt(interval)]
227. $ns_ at $start_time "$a_($i) cbr-start"
228. set start_time [expr $start_time+0.5 ]
229. $ns_ at $opt(stop).001 "$a_($i) terminate"
230. }
231. set node_size 10
232. for {set k 0} { $k<$opt(nn)} {incr k} {
233. $ns_ initial_node_pos $node_($k) $node_size
234. }
235. set opt(stop2) [expr $opt(stop)+200]
236. $ns_ at $opt(stop2).002 "$a_(0) terminate"
237. $ns_ at $opt(stop2).003 "$god_compute_energy"
238. $ns_ at $opt(stop2).004 "$ns_ nam-end-wireless $opt(stop)"
239. $ns_ at $opt(stop2).005 "puts \"NS EXISTING...\"; $ns_ halt"
240. puts $data "New simulation..."
241. puts $data "nodes = $opt(nn), maxspeed = $opt(maxspeed), minspeed = $opt(minspeed), random_seed =
$opt(seed), sending_interval_=$opt(interval), width=$opt(width)"
242. puts $data "x= $opt(x) y= $opt(y) z= $opt(z)"
243. close $data
244. puts "starting Simulation..."
245. #启动仿真
246. $ns_ run

```

第 1~42 行, 对仿真中需要用到变量进行设置, 该部分代码仅为便于修改(多处采用相同模块时仅需修改变量即可), 并未增加任何功能;

第 43~62 行, 对仿真中需用到的模块参数进行设置, 需注意, 此处并未给出所有参数的设置, 仅对必要的参数进行修改, 若未给出则意味采用 NS2 中的默认值;

第 63~96 行, 实例化 ns 变量, 同时设置仿真过程中的 trace 和 nam 输出文件, 以便对实验结果进行分析和演示; 对仿真实验的拓扑范围和全局节点记录进行设置, 以便控制节点位置和运动范围;

第 97~222 行, 配置仿真拓扑中的所有节点及其位置;

第 223~244 行, 对节点大小进行初始化设置;

第 245~246 行, 启动 ns 运行仿真。

5.2 实验扩展

(1) 修改代码。分析 VBF 协议的能量消耗需要修改实验脚本, 具体代码请参考 WNTLab/experiments/exp14/exp14_extend, 下面给出主要修改部分:

```

1. #VBF.tcl脚本内容
2. $ns_ at $opt(stop7).003 "$god_ compute_energy"
3. $ns_ at $opt(stop6).003 "$god_ compute_energy"
4. $ns_ at $opt(stop5).003 "$god_ compute_energy"
5. $ns_ at $opt(stop4).003 "$god_ compute_energy"
6. $ns_ at $opt(stop3).003 "$god_ compute_energy"
7. $ns_ at $opt(stop2).003 "$god_ compute_energy"
8. #plot-util脚本内容
9. set terminal postscript eps color solid linewidth 2 enhanced 'Times' 22
10. set terminal post eps
11. set output "VBF.eps"
12. set nogrid
13. set xlabel "Time(s)"
14. set ylabel "Consumed_Energy(KJ)"
15. set xrange [0:700]
16. set yrange [0:210]
17. plot "VBF-plot.data" using 1:2 title "VBF" with lp
18. #run_exp.sh脚本内容
19. ns VBF.tcl
20. sed -i "1s/God : consumed_energy =/100/g" VBF-plot.data
21. sed -i "2s/God : consumed_energy =/200/g" VBF-plot.data
22. sed -i "3s/God : consumed_energy =/300/g" VBF-plot.data
23. sed -i "4s/God : consumed_energy =/400/g" VBF-plot.data
24. sed -i "5s/God : consumed_energy =/500/g" VBF-plot.data
25. sed -i "6s/God : consumed_energy =/600/g" VBF-plot.data
26. sed -i "7s/God : consumed_energy =/700/g" VBF-plot.data
27. gnuplot plot-util

```

第 1~7 行, VBF.tcl 脚本中新增内容, 用于获取不同时间内能量的消耗情况;

第 8~17 行, 新增绘图脚本, 使用“plot”命令定义绘图各参数, 并且指定输出文件为“VBF.eps”文件;

第 18~27 行, 新增运行脚本, 执行“VBF.tcl”文件, 并使用“gnuplot”命令调用“plot-util”文件将仿真结果绘图。

(2) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

```
cd WNTLab/experiments/exp14/exp14_extend/
```

再输入以下命令运行本实验, 如图 14.5 所示。

```
bash ./run_exp.sh
```

```

=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp14/exp14_extend/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp14/exp14_extend$ bash ./run_exp.sh

```

图 14.5 运行仿真实验

(3) 实验正确运行完成后, 打开 home/wnt-ns2/WNTLab/experiments/exp14/exp14_extend 目录下的 VBF.eps 文件, 可得到图 14.6 所示的结果, 它表示在 100~700s 时间内 VBF 协议能量的消耗情况。

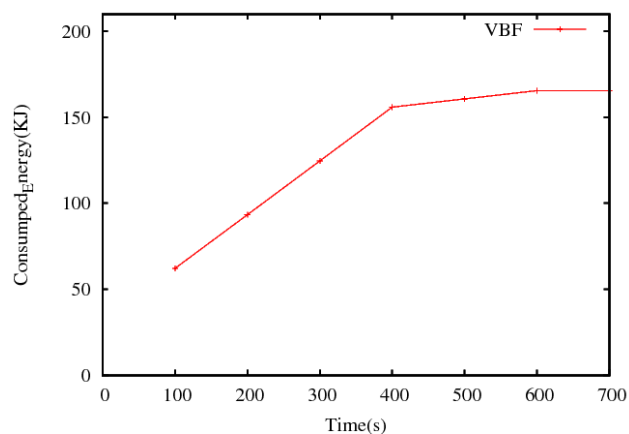


图 14.6 实验结果

6 结果分析与讨论

仿真分析过程中需要注意不同节点位置对协议的影响。

7 注意事项

仿真主要涉及相关脚本, 无需修改仿真模块, 具体代码参见电子资源中的实验源代码。

实验十五 ZigBee 节点安装和组网基础

1 实验要求与目的

- 1 人独立完成
- 掌握 ZigBee 网络实测平台搭建
- 掌握简单 ZigBee 网络组建
- 掌握简单 ZigBee 网络分析
- 掌握 ZigBee 节点结构和组建

2 实验背景与原理

2.1 实验背景

(1) WSN 和 ZigBee 原理知识见教材第 7 和第 8 章。

(2) Z-Stack(ZigBee 协议栈): IEEE802.15.4 标准定义了 PHY 层和 MAC 层, Z-Stack 在此基础上建立。ZigBee 设备应包括 IEEE802.15.4 定义的 PHY 和 MAC 层, 以及 ZigBee 堆栈层: 网络层、应用层和安全服务层。

(3) ZigBee 设备类型:

协调器(ZC)用于启动和控制网络, 存储网络信息, 包括作为认证中心和存储安全密钥; 路由器(ZR)用于扩展网络覆盖面, 实现动态路由, 提供备份路由以防网络拥塞和设备失败。ZR 可联系到协调器和其他路由器, 并支持子设备;

终端设备(ZE)用于可发送或接收数据包, 但不能执行任何路由操作。它们必须被连接到协调器或路由器, 且不支持子设备。

(4) ZigBee 组网流程: 组建一个完整的 ZigBee 网状网络流程如图 15.2 所示。

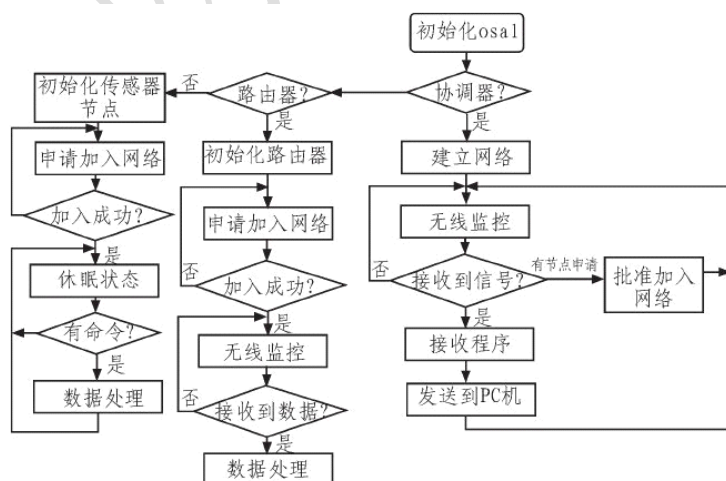


图15.2 ZigBee 网络组网流程

(5) IAR 开发套件: IAR Embedded Workbench(简称 EW)是一套包括编辑器、C/C++交叉编译器、连接器和调试器的集成开发工具, 和各种仿真器、调试器紧密结合, 提供集成的用户界面的开发环境, 用以完成多种微控制器开发。

(6) CC2530 节点: CC2530 是用于 2.4GHz IEEE 802.15.4、ZigBee 和 RF4CE 应用的一个

片上系统解决方案,能以较低成本建立较完备的 WSN 节点。CC2530 结合 RF 收发器、标准增强型 8051CPU、可编程闪存、8KB RAM 和许多功能等。有四种不同闪存版本: F32/64/128/256, 分别有 32/64/128/256KB 的闪存。CC2530 有不同运行模式,使其适用超低功耗要求的系统。本实验中采用的 CC2530 节点如图 15.3 所示,该节点主要参数如下:

- CC2530 芯片: 增强型 8051 处理器 + 射频前端
- 芯片内存: 256k Flash、8K 内部 SRAM
- 电源电压: 5V
- 工作频率: 2.4GHz
- 休眠电流: <10mA
- 传输速率: 250kbps
- 节点开关: 1 个电源选择开关, 1 个复位按键, 2 个普通按键
- 供电方式: Mini USB 接口供电(开关拨到 USB 端)、电池供电(开关拨到 BAT 端)
- 接口: Mini USB 接口, 程序下载接口, 通用 I/O 口, 天线接口, 各类传感器模块接口
- 程序下载: 使用 CC Debugger 调试器烧录程序, 调试器如图 15.4 所示。

2.2 实验原理

本实验为 ZigBee 网络实测实验的基础,搭建 ZigBee 网络实测平台,完成 Z-Stack、IAR EW8051、CC Debugger、Smart Flash Programmer 等环境的安装,进行所有用到 ZigBee 网络、CC2530 传感器节点的实验前需要先完成本实验。

3 实验环境与资源

(1) 环境资源:

CC2530 传感器节点套件 (CC2530 节点、CC Debugger 调试器、USB A 型转 B 型线、下载线、Mini USB 线、天线)、电池、PC 一台、Z-Stack 协议栈、IAR Embedded Workbench 软件、Smart Flash Programmer 软件、ZigBee_Packet_Sniffer 软件、ZigBee_Sensor_Monitor 软件。

(2) 使用说明:

- CC2530 节点:
- CC Debugger 调试器:
- USB A 型转 B 型线: 用于连接 CC Debugger 调试器和 PC 的 USB 接口, 如图 15.5 所示;
- 下载线: 10 针口, 用于连接 CC Debugger 调试器和 CC2530 节点, 如图 15.6 所示;
- Mini USB 线: 用于 CC2530 节点供电、串口数据传输, 如图 15.7 所示;
- 天线: 连接在 CC2530 节点的天线接口, 增大信号传输距离, 如图 15.8 所示。
- Z-Stack 协议栈是 TI 公司提供的半开源实验代码软件包, 官网下载链接为:

<http://www.ti.com/tool/z-stack>;

- IAR Embedded Workbench 是瑞典 IAR Systems 公司开发的集成开发环境, 其支持 CC2530 的版本为 IAR Embedded Workbench for 8051, 通常简称为 IAR EW8051。官网下载链接为: www.iar.com;

- Smart Flash Programmer 是一款通用的程序烧录软件, 主要用于将实验代码烧录到终端节点中。官网下载链接为: www.ti.com/tool/flash-programmer;

- ZigBee_Packet_Sniffer 主要用于抓取 ZigBee 网络中传输的数据包, 类似实验 4 介绍的 Wireshark。官方下载链接: www.ti.com/tool/packet-sniffer;

- ZigBee_Sensor_Monitor 是 TI 公司提供的查看 ZigBee 网络拓扑的一款工具。官方下载链接: <http://www.ti.com/lit/sw/swrc096e/swrc096e.zip>;

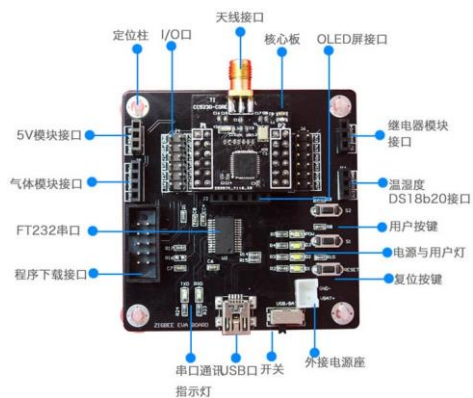


图15.3 CC2530节点产品外观



图15.4 CC Debugger调试器



图15.5 USB A型转B型线



图15.6 下载线



图15.7 Mini USB线



图15.8 天线

4 实验步骤与演示

本实验内容：4.1-4.6 节进行各类环境及工具的安装；4.7-4.8 以 CC2530 模块为基础进行了程序烧录、ZigBee 网络数据包分析实验。

4.1 Z-Stack 的安装

从官网下载完成 Z-Stack 安装包后，按以下步骤即可安装成功。

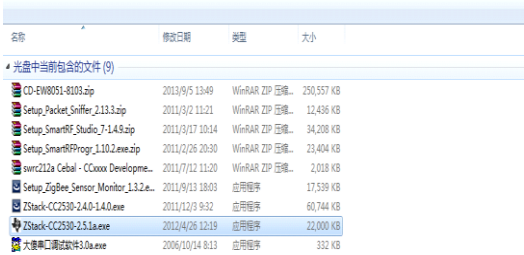


图15.9 Z-Stack安装包(1)



图15.10 Z-Stack安装(2)

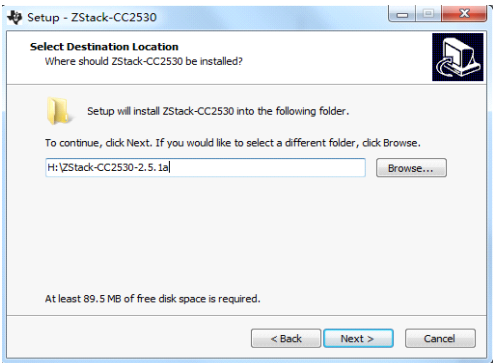


图15.11 Z-Stack安装(3)



图15.12 Z-Stack安装(4)

4.2 IAR EW8051 的安装

下载完成 IAR Embedded Workbench for 8051 安装包后, 如图 15.13 和 15.14 所示。解压缩后双击运行图 15.15 中的可执行安装程序, 安装界面如图 15.16 所示, 紧接着根据图 15.17 至图 15.26 的提示一步步进行, 直至安装成功。(需注意, 本软件非免费软件, 如图 15.19, 软件安装过程中需要输入序列号, 请读者从官方渠道获取序列号。)

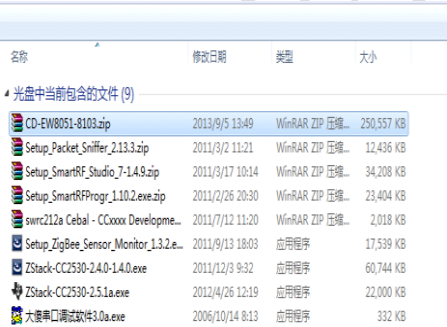


图15.13 IAR EW安装包



图15.14 解压缩文件

名称	修改日期	类型	大小
doc	2011/9/30 9:14	文件夹	
dongle	2011/9/30 9:14	文件夹	
ew8051	2011/9/30 9:14	文件夹	
license-init	2011/9/30 9:14	文件夹	
autorun.exe	2011/9/30 9:14	应用程序	574 KB
autorun.inf	2011/9/30 9:14	安装信息	1 KB
autorun.ini	2011/9/30 9:14	配置设置	3 KB

图15.15 IAR EW可执行文件



图15.16 IAR EW安装(1)



图15.17 IAR EW安装(2)

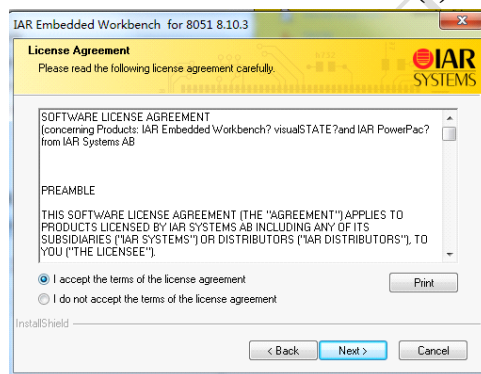


图15.18 IAR EW安装(3)

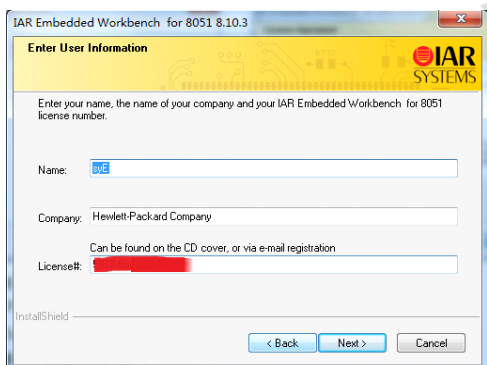


图15.19 IAR EW安装(4)

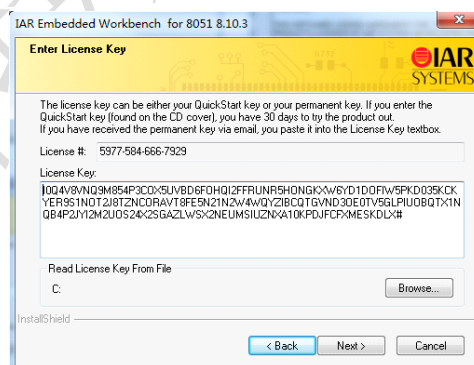


图15.20 IAR EW安装(5)

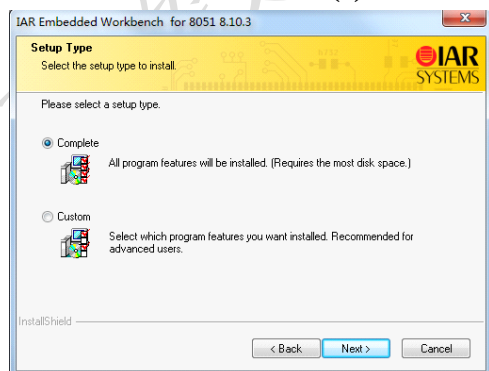


图15.21 IAR EW安装(6)

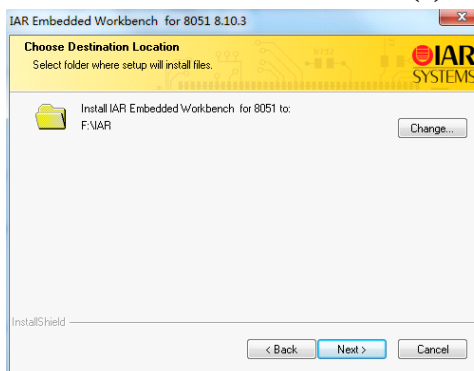


图15.22 IAR EW安装(7)

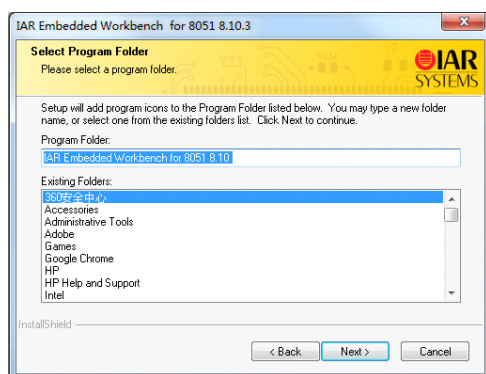


图15.23 IAR EW安装(8)

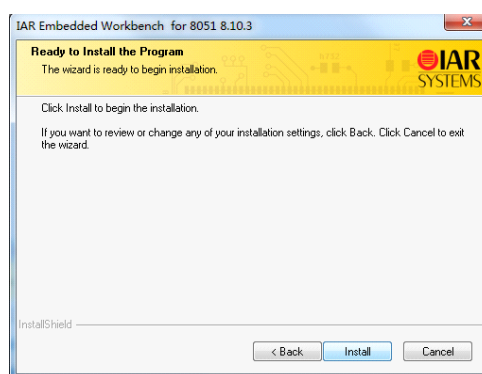


图15.24 IAR EW安装(9)



图15.25 IAR EW安装(10)

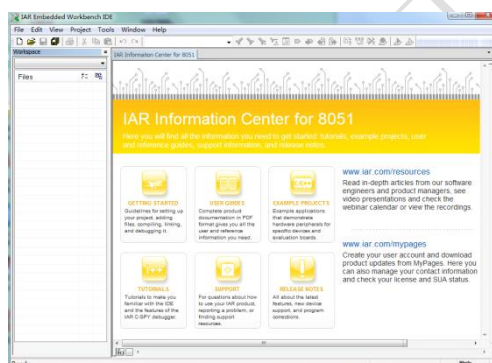


图15.26 IAR EW安装(11)

4.3 CC Debugger 的驱动安装

将仿真器连接好数据线, 连接已联网的 PC, Windows 会自动为仿真器安装设备驱动程序。读者也可自行下载相应驱动程序, 自行安装调试。

4.4 Smart Flash Programmer 的安装

获取 Smart Flash Programmer 安装包后解压如图 15.27。具体安装步骤如图 15.28 至图 15.32 所示。



图 15.27 Smart Flash Programmer 安装包



图 15.28 Smart Flash Programmer 安装(1)

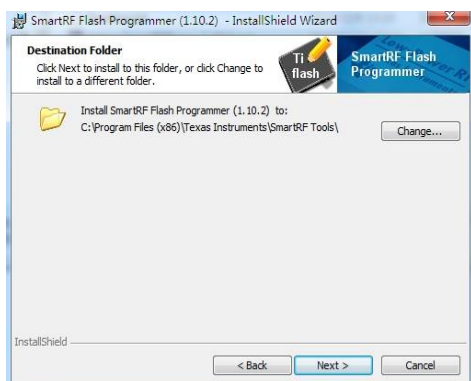


图 15.29 Smart Flash Programmer 安装(2)



图 15.30 Smart Flash Programmer 安装(3)

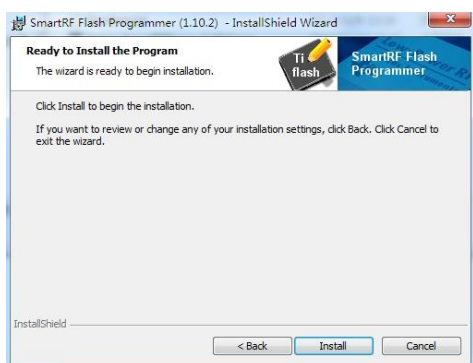


图 15.31 Smart Flash Programmer 安装(4)

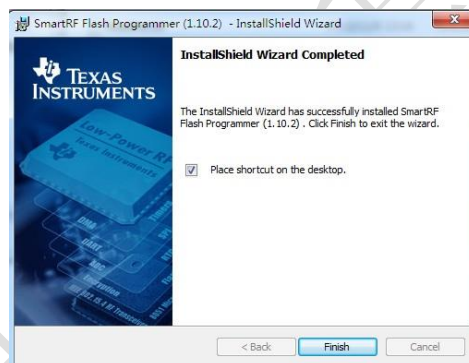


图 15.32 Smart Flash Programmer 安装(5)

4.5 ZigBee_Packet_Sniffer 的安装

获取安装包后解压如图 15.33。具体安装步骤如图 15.34 至图 15.38 所示。

名称	修改日期	类型	大小
Readme.txt	2014/6/25 13:24	文本文档	12 KB
Setup_SmartRF_Packet_Sniffer_2.18.0.exe	2014/6/25 13:25	应用程序	10,685 KB
Setup_SmartRF_Packet_Sniffer_2.18.1.exe	2014/6/26 11:54	应用程序	12,540 KB

图 15.33 ZigBee_Packet_Sniffer安装包

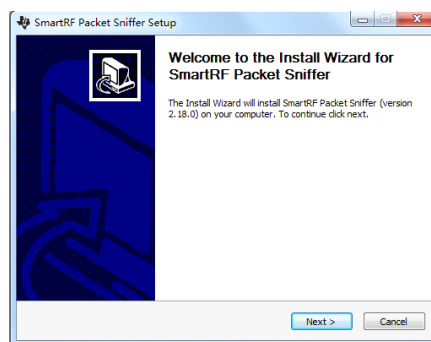


图 15.34 ZigBee_Packet_Sniffer安装(1)

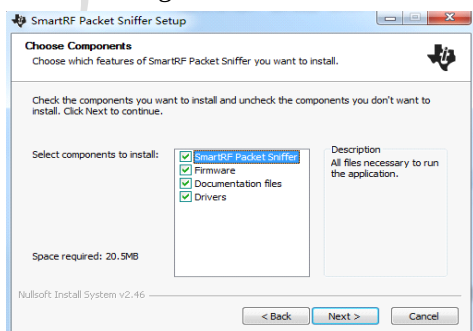


图 15.35 ZigBee_Packet_Sniffer安装(2)

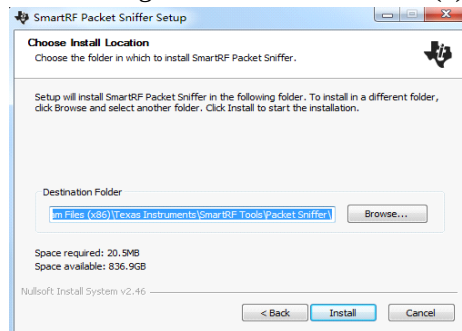


图 15.36 ZigBee_Packet_Sniffer安装(3)

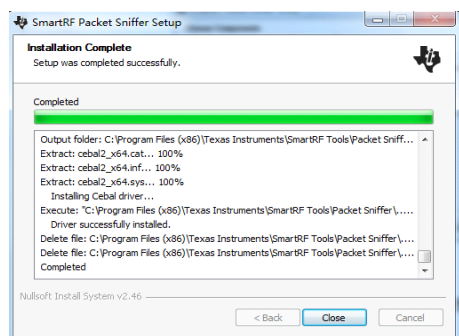


图15.37 ZigBee_Packet_Sniffer安装(4)

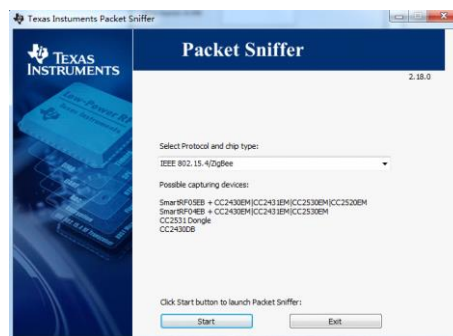


图15.38 ZigBee_Packet_Sniffer安装(5)

4.6 ZigBee_Sensor_Monitor 的安装

下载解压后按下列步骤即可成功安装。

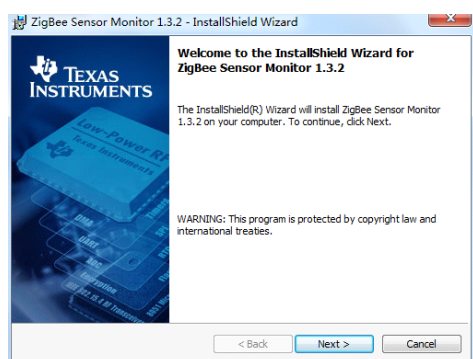


图15.39 ZigBee_Sensor_Monitor安装(1)

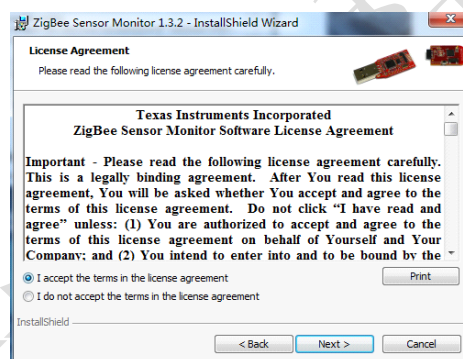


图15.40 ZigBee_Sensor_Monitor安装(2)

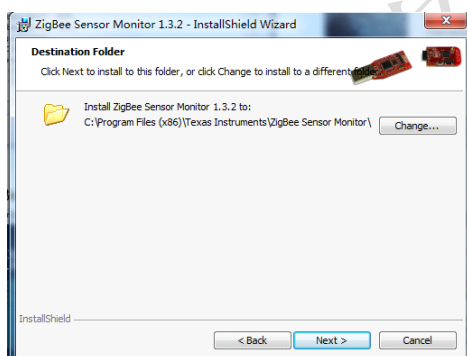


图15.41 ZigBee_Sensor_Monitor安装(3)

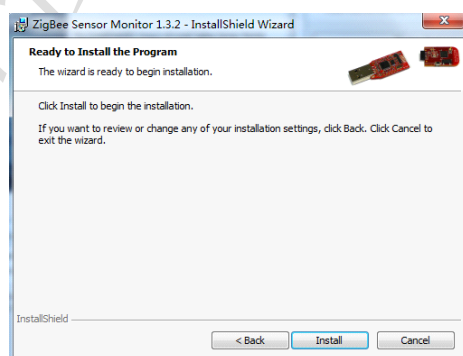


图15.42 ZigBee_Sensor_Monitor安装(4)

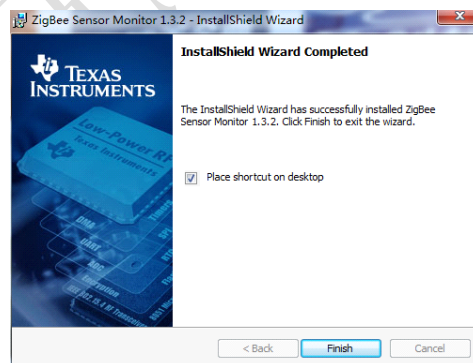


图15.43 ZigBee_Sensor_Monitor安装(5)

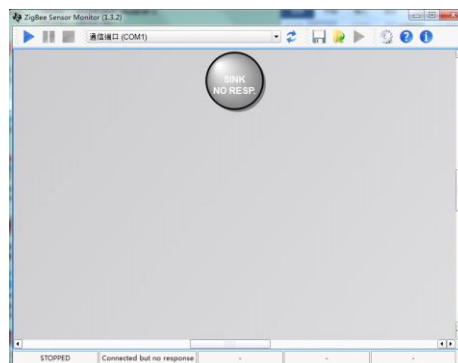


图15.44 ZigBee_Sensor_Monitor安装(6)

4.7 传感器节点的程序烧录方法

实验平台安装成功以后,就可以进行代码编辑、编译、调试和下载。本节对程序烧录方法进行简介,后续实验实际使用时会详细介绍。

下面介绍两种程序烧录方式:

1: EW 自带烧录

具体过程如下:

- (1)用 CC debugger 仿真器连接 PC 机和传感器节点。仿真器电源接通,指示灯显示红色;
- (2)在放置于 ZStack 协议栈下的项目文件夹中找到后缀名为 eww 的工程文件并双击打开,即可在 EW 中打开整个工程;
- (3)点击右上角的  图标,编译整个工程;
- (4)编译成功后,再按下仿真器复位键,此时指示灯显示为绿色;
- (5)在 workspace 列表框中选择对应的烧录对象,点击  图标,即可自动下载程序;
- (6)程序烧录完成后请点击红色叉号按钮,停止调试。

2: SmartRF flash Programmer 烧录

具体过程如下:

- (1)用 CC debugger 仿真器连接 PC 机和传感器节点,仿真器电源接通,指示灯显示红色;
- (2)双击桌面图标  打开烧录软件,界面如图 15.45 所示;
- (3)传感器节点装上两节 1.5v 电池,开启传感器节点开关,再按下仿真器复位键,若 CC debugger 仿真器驱动安装成功,则仿真器指示灯为绿色,即连接正常。若指示灯依旧为红色,则需将 PC 联网,再重新插拔 CC debugger 仿真器,Windows 系统会自动安装驱动程序。成功连接后,SmartRF flash Programmer 工具界面显示如图 15.46 所示;

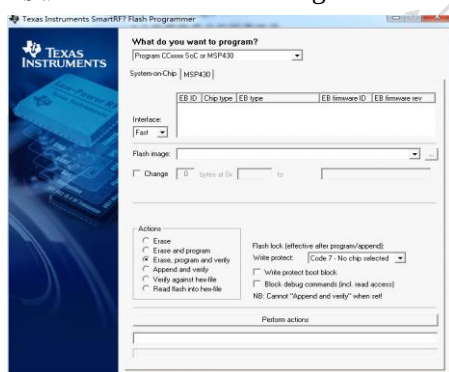


图15.45 SmartRF flash Programmer启动界面

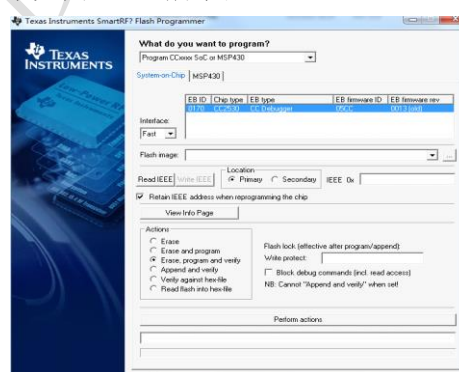


图15.46 成功连接CC2530节点

- (4)在 Flash image 一行中选择省略号按钮,在目录中选择协调器对应的烧录文件。完成后如图 15.47 所示;

- (5)点击按钮“Perform actions”即可将对应文件烧录到对应的传感器节点中,即完成传感器的代码烧录工作。效果如图 15.48 所示。

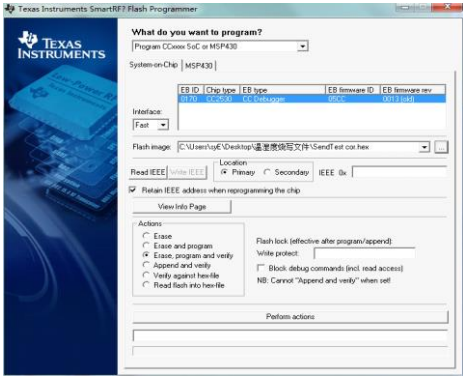


图15.47 选择烧录文件

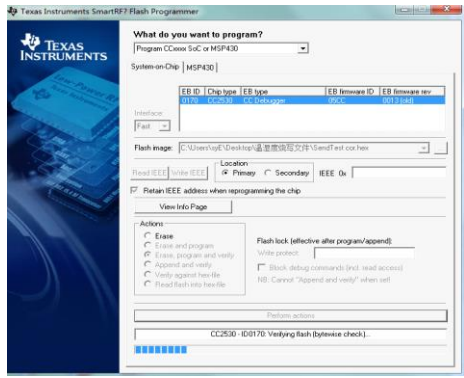


图15.48 正在烧录文件

4.8 ZigBee 网络数据包分析

目前,有多种方式来分析 ZigBee 网络,这里介绍一种采用 TI 公司提供的无线网络分析仪 ZigBee_Packet_Sniffer 来实现 WSN 的数据包嗅探分析。本节对该软件的使用进行简介,仅供读者简单了解,后续实验实际使用时会详细介绍。

- (1) 根据 4.7 节介绍,在协调器节点烧录协调器程序;在待测节点烧录终端节点程序;
- (2) 待传感器节点代码完成后,用仿真器连接协调器和 PC 机;
- (3) 然后打开 Texas Instruction Packet Sniffer 软件,如图 15.49 所示:

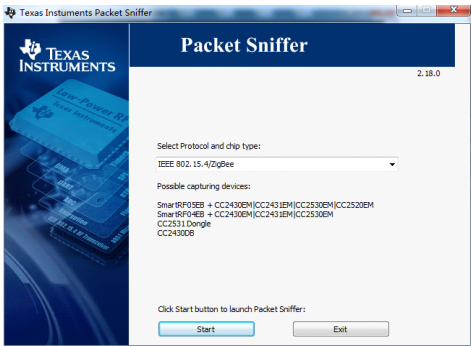


图 15.49 Texas Instruction Packet Sniffer 启动界面

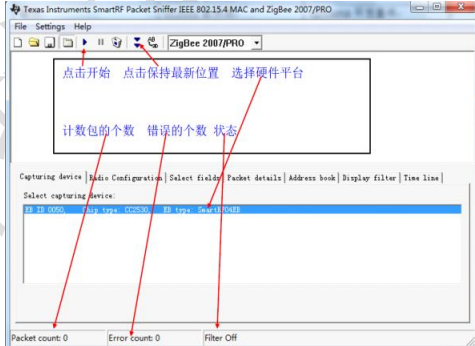


图 15.50 Packet Sniffer 工作界面

- (4) 然后选择 IEEE802.15.4/ZigBee, 启动。Packet Sniffer 工作界面如图 15.50。
- (5) 依次开启协调器和终端节点的电源;
- (6) 等网络稳定后, 点击无线网络分析仪开始按钮, 即可抓取数据包, 如图 15.51 所示:

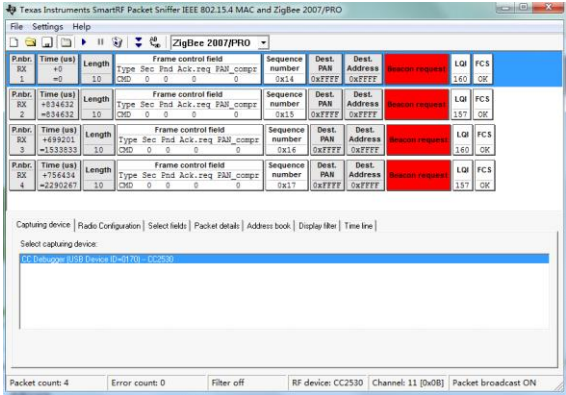


图 15.51 ZigBee 网络抓包实验效果图

(7) 可选择“Select fields”设置数据包数据域, 选择“Packet details”查看数据包具体内容。读者可参考相关手册熟悉无线网络分析仪其他用法, 此处不再赘述。

5 结果分析与讨论

为减少实验复杂性,实验中仅采取 3 个传感器节点进行实验(1 个协调器+2 个终端节点)。本实验中的 ZigBee 网络数据包分析和 ZigBee 网络拓扑分析实验,均基于 ZStack 提供的样本实验程序。读者可自行添加传感器节点,实现较复杂网络的网络拓扑分析和数据包分析,也可自行修改 TI 公司提供的代码。

6 注意事项

本实验平台由于安装软件较多,具体操作时由于各项配置的不同,可能会有不同的解决方案,如有问题,可考虑选择替换软件和不同的实现方式,或者联系我们,联系邮箱和读者服务 QQ 群见本手册前言。

实验十六 Arduino 节点安装与开发基础

1 实验要求与目的

- 1 人独立完成
- 了解 Arduino 单片机的使用
- 实现 Arduino 控制 Bluetooth 通信方法

2 实验背景与原理

2.1 实验背景

Arduino、Bluetooth 和 ZigBee 的背景知识见本书第 8、9 章。

2.2 实验原理

本章实验由 3 部分组成：

(1) Arduino 单片机开发环境的配置和代码烧录过程；

(2) Arduino 单片机控制蓝牙 4.0 模块，以实现 Arduino 节点与安卓手机之间的数据通信，如图 16.1 所示；

(3) Arduino 单片机控制 Zigbee 模块，以实现 Arduino 节点与主机之间的无线通信，如图 16.2 所示。

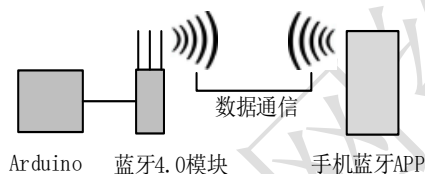


图 16.1 BLE 通信原理图

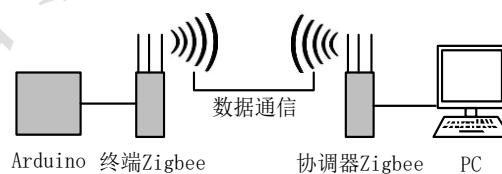


图 16.2 Zigbee 通信原理图

3 实验环境与资源

3.1 Arduino 代码烧写

(1) 环境资源：

PC 一台、Arduino MEGA 2560 单片机模块一个、USB A 型转 B 型线一根、Arduino IDE。

(2) 使用说明：

● 各类硬件如图 16.3 和 16.4 所示，详细的使用方法见实验步骤；

● Arduino IDE 是 Arduino 系列单片机的开发环境，考虑到稳定性，推荐读者使用 1.0.x 系列版本，官网下载连接为 <https://www.arduino.cc/en/Main/OldSoftwareReleases#previous>，详细的下载方法及说明请见实验资源包 WNTLab/experiments/exp16。

● 本实验开放所有源码，供读者学习与二次开发，详见实验资源包 WNTLab/experiments/exp16。

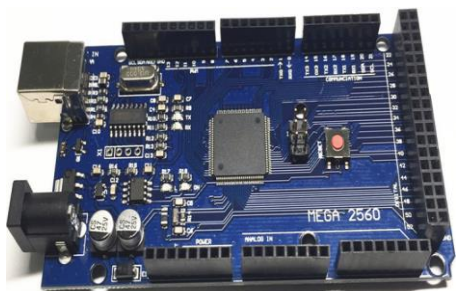


图 16.3 Arduino MEGA 2560 单片机模块



图 16.4 USB A 型转 B 型线

3.2 蓝牙 4.0 模块通信

(1) 环境资源:

在 3.1 的基础上增加: 蓝牙 4.0 模块两个、电池一组、USB 转串口模块、公对母杜邦线若干、母对母杜邦线若干、SSCOM 串口调试助手

(2) 使用说明:

- 各类硬件如图 16.5~16.8 所示, 详细的使用方法见实验步骤;
- 本实验开放所有源码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp16;
- SSCOM 串口调试助手: 作者使用版本为 5.12 版, 压缩包大小约 400KB, 下载方法请见实验资源包 WNTLab/experiments/exp16, 官方下载地址为: <http://www.daxia.com/bibis/>。

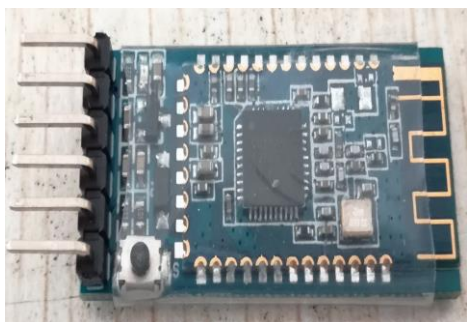


图 16.5 蓝牙 4.0 模块



图 16.6 电池



图 16.7 USB 转串口模块



图 16.8 公对母杜邦线

3.3 ZigBee 模块通信

(1) 环境资源:

在 3.1 基础上增加: PC 一台、CC2530 传感器节点套件两套及软件环境(详见实验十五第 3 节)、电池一组、USB 转串口模块、公对母杜邦线若干、SSCOM 串口调试助手。

(2) 使用说明:

- CC2530 传感器节点套件介绍见实验十五;

●SSCOM 串口调试助手的官方下载地址为: <http://www.daxia.com/bibis/>, 详细下载方法及软件简介请见实验资源包 WNTLab/experiments/exp16;

●本实验开放所有源码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp16。

4 基础实验步骤与演示

4.1 Arduino 代码烧录

本节实验内容为向 Arduino 单片机中烧录代码, 具体步骤如下:

(1) 根据第 3.1 节介绍, 下载 Arduino IDE 并解压。进入解压后的文件夹, 运行 arduino.exe 即可。Arduino IDE 界面如图 16.9 所示。

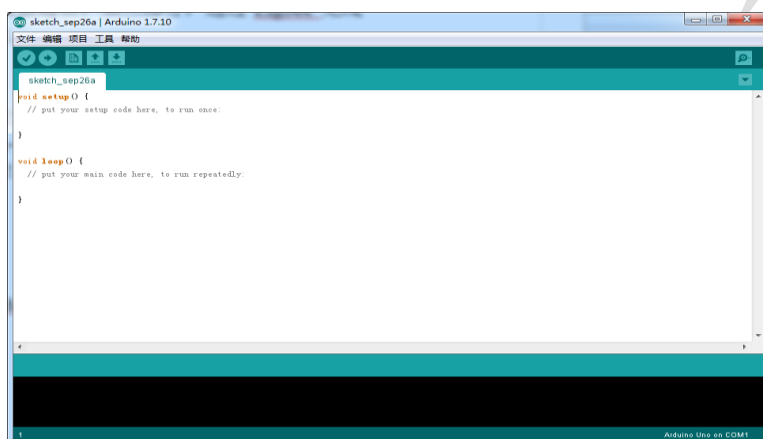


图 16.9 Arduino IDE 界面

(2) 用 USB 串口线连接 Arduino MEGA 2560 单片机模块和 PC, 打开“设备管理器”, 查看连接所占用端口号, 如图 16.10 所示。

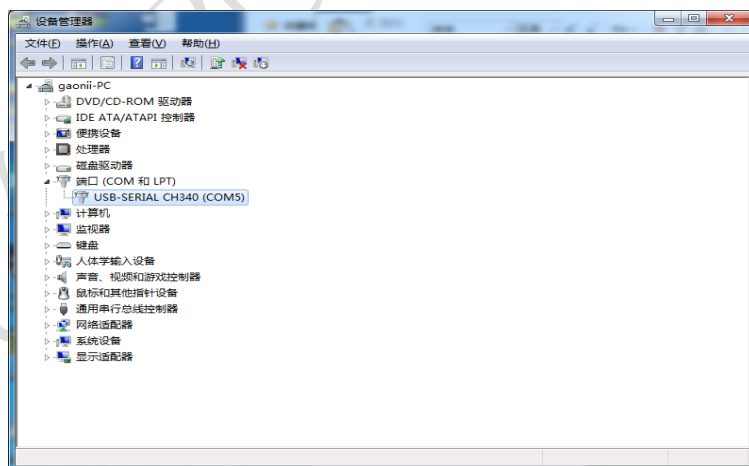


图 16.10 查看端口号

(3) 正确连接后, 打开 Arduino IDE 的“工具”菜单, 选择“板”→“Arduino Mega 2560”, 如图 16.11 所示。紧接着, 打开“工具”菜单, 选择“端口”→“COM5”, 如图 16.12 所示。注意, 此处的端口号与图 16.10 查看的端口号相同。

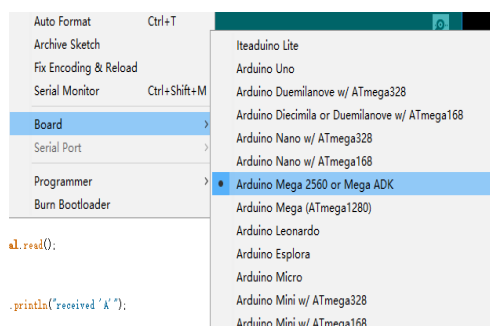


图 16.11 单片机型号选择

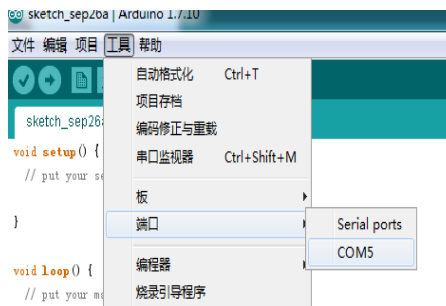


图 16.12 连接端口选择

(4) 所有连接和配置完成后, 接着烧录示例代码进行测试。该示例代码中(图 16.13)仅实现了串口打印字符串的功能。图 16.14 至图 16.17 为代码编译、烧录和结果观测过程, 其中红色方框为需点击的按钮。注意设置波特率为 9600。



图 16.13 测试代码编写



图 16.14 测试代码编译



图 16.15 烧录测试代码



图 16.16 打开串口监视器

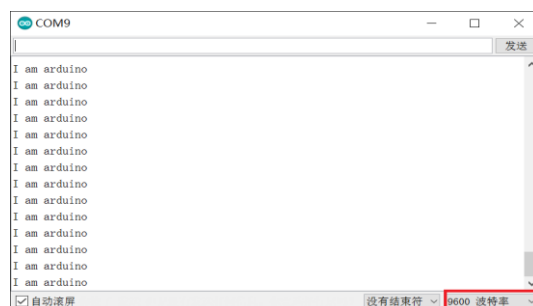


图 16.17 串口结果输出

4.2 蓝牙 4.0 模块通信

本节实验内容为利用 Arduino 单片机控制蓝牙 4.0 模块通信, 具体步骤如下:

(1) 环境配置和代码烧录过程同 4.1 节。具体到本实验, 实现的 Arduino 代码功能为: 接收到‘A’字符后, 回复“received ‘A’”, 如图 16.18 所示。



图 16.18 BLE 测试代码

(2) 代码烧录完成后, 利用母对母杜邦线按照图 16.19 所示方式, 连接 Arduino MEGA 2560 单片机模块和蓝牙 4.0 模块; 利用公对母杜邦线按照图 16.20 所示方式, 连接 USB 转串口模块和蓝牙 4.0 模块。

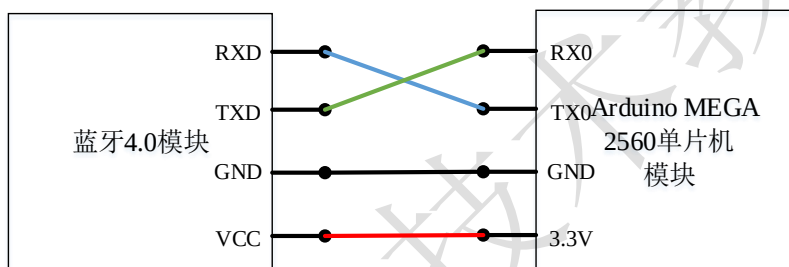


图 16.19 蓝牙模块与 Arduino 连接原理图

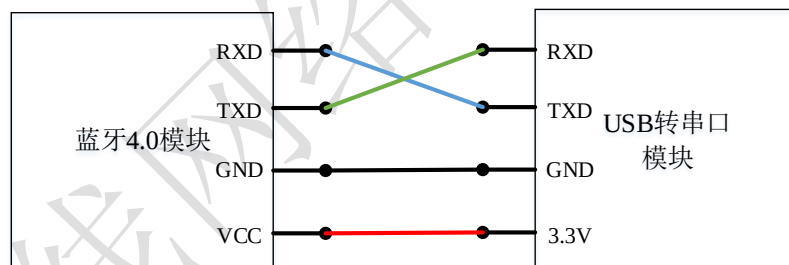


图 16.20 蓝牙模块与 USB 转串口模块连接原理图

(3) 连接完成后, 将 USB 转串口模块连接至 PC 的 USB 接口, 如图 16.21 所示; 将电池连接至 Arduino MEGA 2560 单片机模块的 DC 电源口, 如图 16.22 所示。



图 16.21 USB 转串口模块连接至 PC



图 16.22 Arduino 连接至电池

(4) 在 PC 中打开 SSCOM 串口调试助手, 选择相应串口号, 并设置波特率为 9600, 然

后打开串口, 如图 16.23 所示。

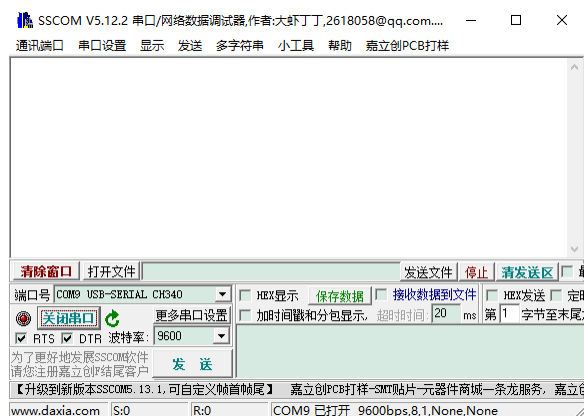


图 16.23 打开 SSSCOM 串口调试助手并配置

(5) 按步骤输入以下指令:

输入并发送指令 **AT+ROLE1**, 设置蓝牙 4.0 模块为主设备模式, 成功后会返回 “+ROLE=1” 和 “OK” 信息;

输入并发送指令 **AT+INQ**, 开始搜索蓝牙信号, 并返回搜索到的信号, 如图 16.24 所示;

输入并发送指令 **AT+CONN<Param>**, 连接指定蓝牙信号, 其中<Param>代表上一步中搜索到的蓝牙信号的序号, 比如此处为 AT+CONN1。



图 16.24 输入并发送 AT 指令

(6) 当第(5)步中连接成功后, 观察两个蓝牙 4.0 模块的指示灯由闪烁状态变为常亮状态。此时在 SSSCOM 串口调试助手中输入并发送 “A”, 返回 “received ‘A’” 如图 16.25 所示。至此两个蓝牙 4.0 模块通信成功, 实验完成。

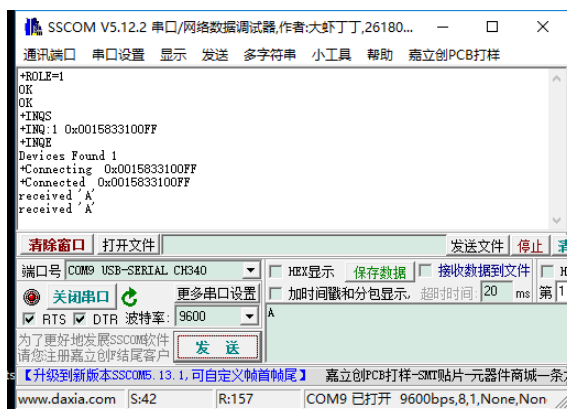


图 16.25 蓝牙模块通信成功

4.3 ZigBee 模块通信

本节实验内容为利用 Arduino 单片机控制 ZigBee 模块通信。(需注意, 进行该实验之前, 读者需先完成实验十五, 即配置好 ZigBee 网络的实测环境。)

4.3.1 ZigBee 模块配置

本实验将用到两个 CC2530 节点, 需要分别将它们配置为协调器和终端, 具体步骤如下:

(1) 用 CC Debugger 烧录线连接 PC 和一个 CC2530 节点, 使用 Mini USB 线连接相应电源接口, 同时按下 CC Debugger 的 reset 键, 显示灯为绿色, 连接完成后如图 16.26 所示;



图 16.26 节点与电脑间的连接

(2) 在主机上打开实验源码的工程文件, 在电子资源 exp16 实验目录下, Z-Stack 透传 \Projects\zstack\Utilities\SerialApp\CC2530DB\ 文件夹里。找到 SerialApp.eww 文件, 双击打开, IAR EW 软件并自动导入工程项目;

(3) 打开项目, 在左上角选择 CoordinatorEB (协调器节点选择该项, 如果是终端节点或路由节点, 请选择 EndDeviceEB 或 RouterEB, 下述过程以协调器代码的编译和烧录为例), 并在项目上单击右键, 进行 Rebuild All 操作, 完成烧录节点的代码编译, 如图 16.27 所示。注意, 编译完成后要确保无错误, 可查看软件下方的输出提示;

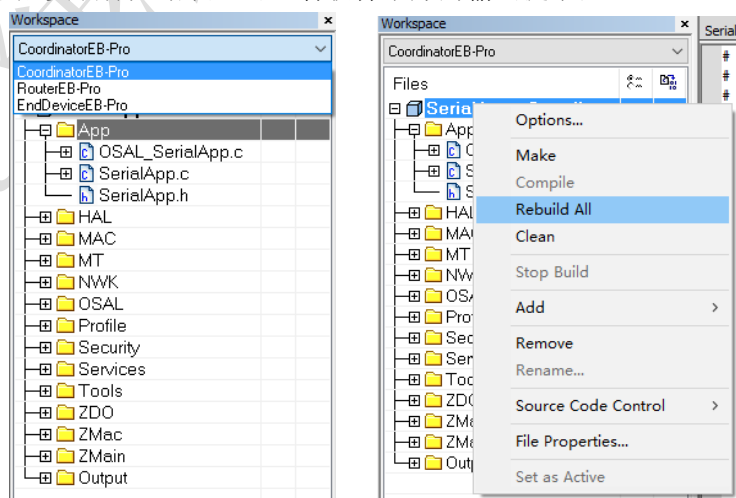


图 16.27 CC2530 节点烧录代码编译

(4) 正确编译后, 选择工具栏中的下载按钮(图 16.28 中黑色方框)进行节点烧录。

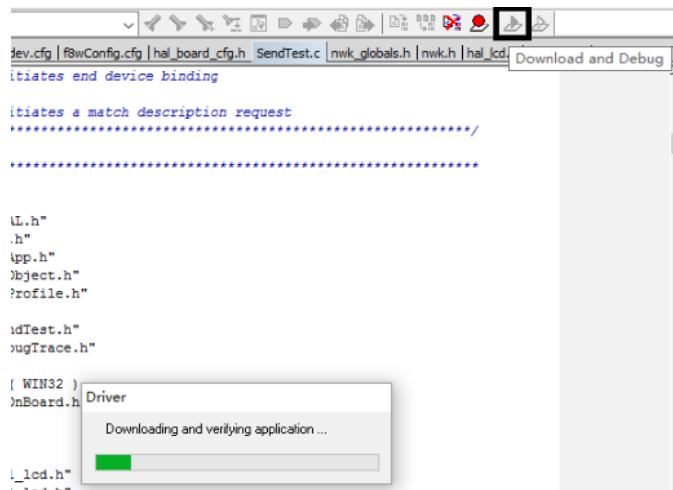


图 16.28 CC2530 节点烧录

(5) 此外本节实验还需一个终端节点。因此使用另一个 CC2530 节点, 重复上述步骤, 在第(3)步中选择 EndDeviceEB, 编译并烧录终端节点代码。(需注意, 代码烧录过程中如出现错误请参考注意事项。)

4.3.2 ZigBee 模块通信

配置 ZigBee 模块完成后, 按以下步骤进行实验:

(1) 如图 16.29 所示, 将 4.3.1 中配置好的 ZigBee 协调器和终端节点的核心板取下, 得到如图 16.30 所示的两个模块, 其中 1 个为协调器, 1 个为终端。

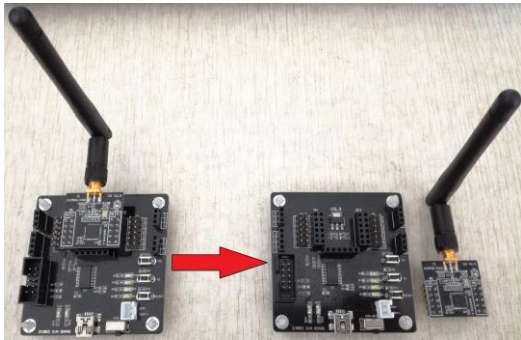


图 16.29 取下核心板



图 16.30 协调器和终端

(2) Arduino 的环境配置和代码烧录过程同 4.1。具体到本实验, 实现的 Arduino 代码功能为: 一直发送“<http://www.thinkmesh.net/wireless/>”字符串, 如图 16.31 所示。

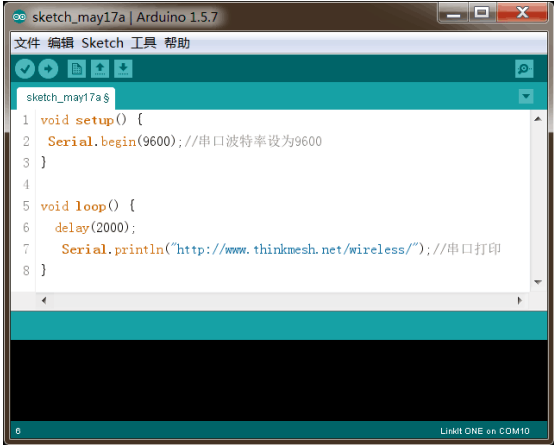


图 16.31 Arduino 单片机代码

(3) 将步骤(1)中获得的协调器与 USB 转串口模块按照图 16.32 所示方式连接, 将终端与 Arduino MEGA 2560 单片机模块按照图 16.33 所示方式连接。

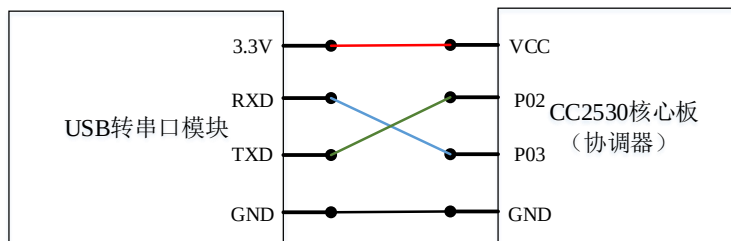


图 16.32 协调器与 USB 转串口模块连接

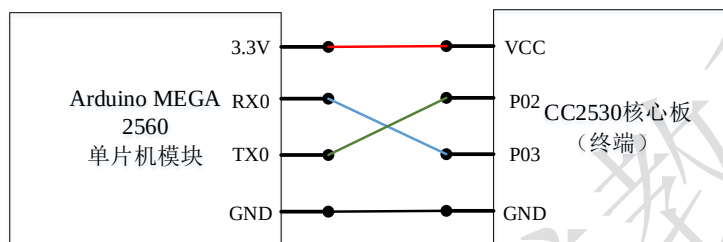


图 16.33 终端与 Arduino MEGA 2560 单片机模块连接

(4) 完成步骤(3)后, 将协调器端的 USB 接口连接到 PC, 如图 16.34 所示; 并在 PC 中打开 SSCOM 串口调试助手, 打开对应端口, 如图 16.35 所示。

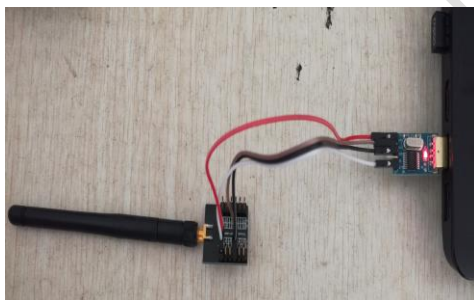


图 16.34 协调器实物图

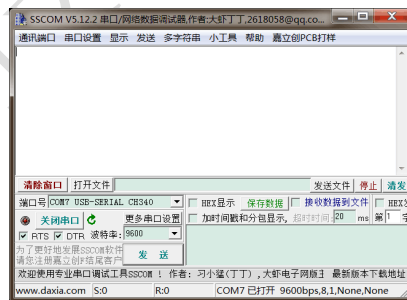


图 16.35 SSCOM 串口助手

(5) 给连接好的 Arduino MEGA 2560 单片机模块插上电源, 如图 16.36 所示。

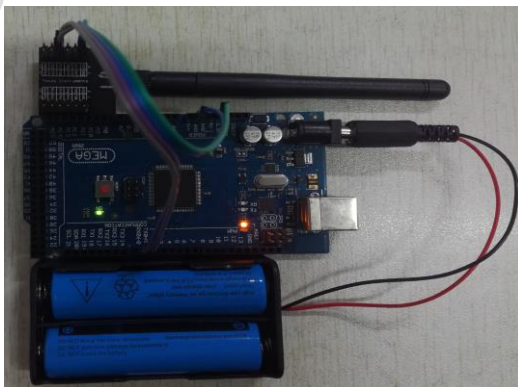


图 16.36 终端上电运行

(6) 终端上电运行后, 等待数秒后 SSCOM 串口助手中会显示协调器和终端连接成功, 如图 16.37; 之后每隔 2 秒, 终端会向协调器发送字符串“<http://www.thinkmesh.net/wireless/>”, 协调器收到后输出显示在 SSCOM 串口助手中, 如图 16.37。

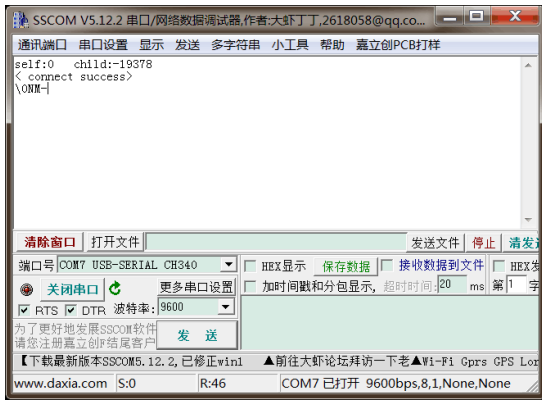


图 16.37 Zigbee 实验结果

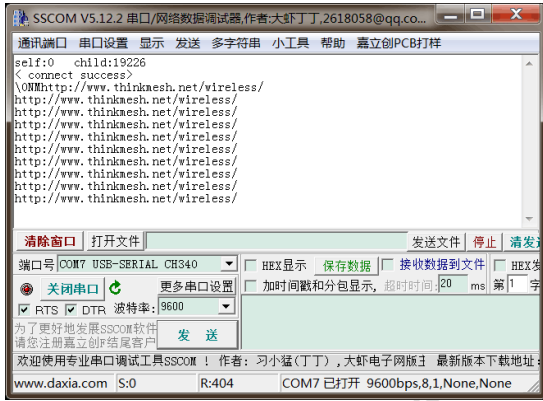


图 16.38 Zigbee 实验结果

5 结果分析与讨论

不同设备的 RSSI 受多种因素影响, 导致 RSSI 出现动态性。本实验测量 WiFi 和 ZigBee 的 RSSI 受环境变化的影响, 注意由于实验测量粒度较粗, 存在一定误差。

6 注意事项

实验过程中, 在对 CC2530 单片机代码的烧录过程中, 烧录线要插紧插槽, reset 键要多按几下, 不然烧录过程会出现其他问题。

如果问题无法解决, 可以联系我们, 联系邮箱和读者服务 QQ 群见本手册前言。

实验十七 ZigBee 户外环境监测网络

1 实验要求与目的

- 多人共同完成
- 了解 ZigBee 网络基本应用
- 熟悉 ZigBee 户外环境监测

2 实验背景与原理

2.1 实验背景

(1) WSN 和 ZigBee 原理知识：见教材第 7、8 章。

(2) ZigBee 户外环境监测网络由大量传感器节点组成，终端节点采集数据，并将采集到的数据处理后，通过路由器节点发送给协调器节点，协调器节点将收集的数据上传到 PC 机，完成存储，便于以后处理。ZigBee 户外环境监测网络是一类典型的 WSN 应用，还包含了一些关键技术，如节点部署、远程控制、数据采样、通信机制等，具有广泛的应用相关性，我们在具体应用的时候，要结合实际情况进行适当的修改。

2.2 实验原理

本实验中 ZigBee 户外环境监测系统由上位机监控端和下位机 ZigBee 网络两部分组成：下位机由搭载温湿度传感器的终端节点、路由器节点和协调器节点三部分组成，终端节点负责采集、存储和上传温湿度信息，路由器节点负责数据包转发和路由决策，协调器节点完成温湿度数据的汇聚并上传至 PC 机；上位机负责接收并显示数据，实现实时监控，与下位机中的协调器节点之间使用 USB 转串口模块进行通信。

实验所用网络拓扑如图 17.1 所示，读者可根据具体实验环境决定网络布局，如读者要扩大网络范围，则需增加终端节点和路由器节点。

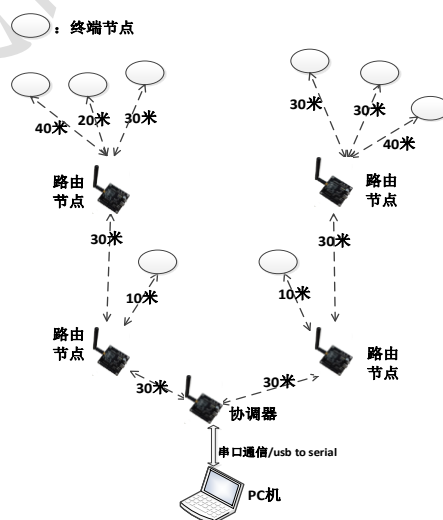


图 17.1 ZigBee 室外多跳网络拓扑

3 实验环境与资源

(1) 环境资源:

笔记本 PC 一台、CC2530 传感器节点套件若干及软件环境(详见实验十五第 3 节)、温湿度传感器若干、电池及 5V 稳压模块若干、母对母杜邦线若干、SSCOM 串口调试助手软件、CC2530 节点代码。

(2) 使用说明:

- CC2530 传感器节点套件介绍见实验十五, 其它各类硬件如图 17.2~17.4 所示;
- SSCOM 串口调试助手: 作者使用版本为 5.12 版, 压缩包大小约 400KB, 下载方法请见实验资源包 WNTLab/experiments/exp17, 官方下载地址为: <http://www.daxia.com/bibis/>。
- 本监测系统实验开放所有源码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp17。

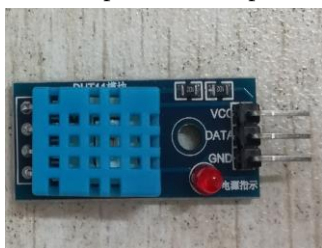


图 17.2 温湿度传感器



图 17.3 电池及 5V 稳压模块



图 17.4 母对母杜邦线

4 基础实验步骤与演示

基础实验中先组建一个单路由单终端网络进行测试, 主要为了展示路由器节点的数据包转发作用, 并为大规模户外组网做准备。

之后开始使用多个路由和终端节点组建户外环境监测网络, 真实地监测实验场地中的环境信息。

最后进行对比性实验, 改变终端节点中传感器所处的环境, 以测试系统准确性和可靠性。

(需注意, 进行该实验之前, 读者需先完成实验十五, 即配置好 ZigBee 网络的实测环境。)

4.1 CC2530 节点程序烧录

本节实验给 CC2530 节点烧录程序, 为后续实验做准备。

按照图 17.1 所示, 本实验需要 1 个协调器节点、4 个路由器节点、8 个终端节点。因此需对 13 个 CC2530 节点烧录不同节点类型的程序。(教学实验时可使用多个实验箱的节点共同进行实验, 可根据实际情况调整网络拓扑, 更改节点个数。)

具体烧录方法请参考实验十五 4.7 节, 此处只说明本实验烧录终端节点程序所需注意之处: 整个网络中需用到多个终端节点采集信息, 因此为了在上位机中识别每条数据对应的终端节点, 需在终端节点发出的数据包中添加终端节点的编号信息。

如图 17.5 所示, enddevice.c 文件中画红线处为终端节点的编号, 8 个终端节点编号分别为“E01”至“E08”, 即每次烧录终端节点程序都需要修改此处代码。

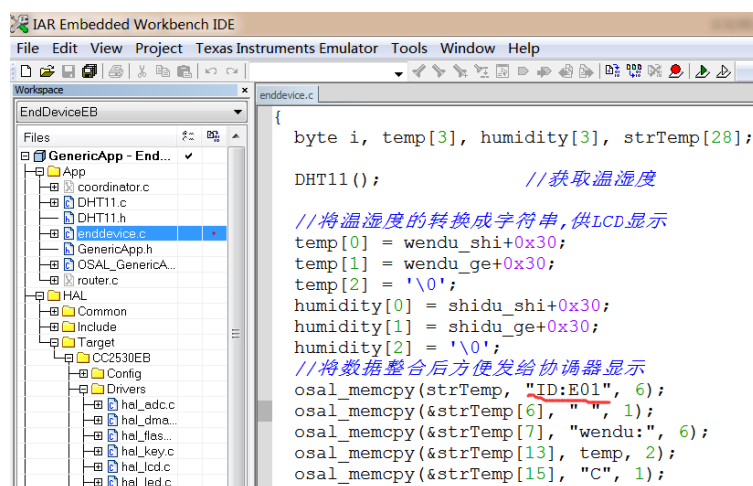


图 17.5 修改终端节点编号

4.2 单路由单终端网络组建

本节实验主要组建一个单路由单终端网络进行测试, 步骤如下:

- (1) 准备 3 个烧录了程序的 CC2530 节点, 其中一个为终端节点, 一个为路由器节点, 一个为协调器节点;
- (2) 将温湿度传感器与终端节点按照图 17.6 所示方式连接, 并将电池连接在节点的外接电源座, 如图 17.7 所示; 将协调器节点通过 Mini USB 线与笔记本 PC 的 USB 口相连, 如图 17.8 所示; 将电池连接在路由器节点的外接电源座;

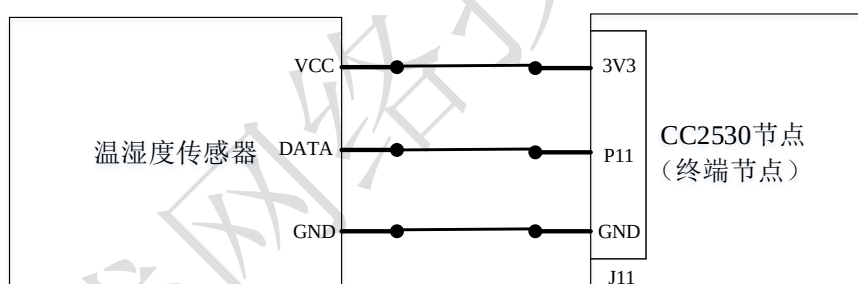


图 17.6 温湿度传感器与终端节点连线原理图

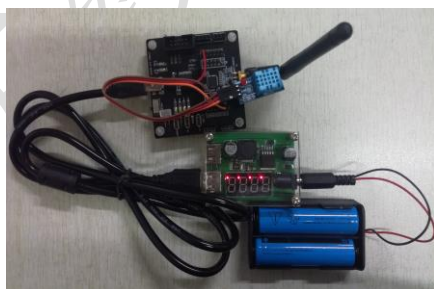


图 17.7 终端节点连线实物图 图 17.8 协调器节点与笔记本 PC 连接

- (3) 打开协调器节点的电源开关, 在笔记本 PC 中打开 SSCOM 串口调试助手软件, 并设置波特率为 9600, 打开对应端口号, 如图 17.9 所示;

- (4) 打开终端节点的电源开关, 等待数秒后协调器节点绿色灯开始闪烁, 表示终端节点已与协调器节点建立连接, 并开始传输数据, 此时 SSCOM 串口调试助手中会显示协调器节点收到的数据, 如图 17.10 所示;

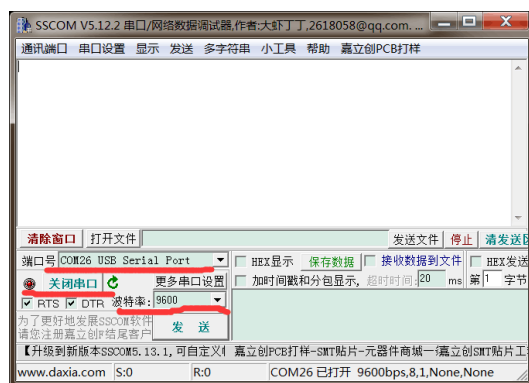


图 17.9 打开 SSSCOM 串口调试助手

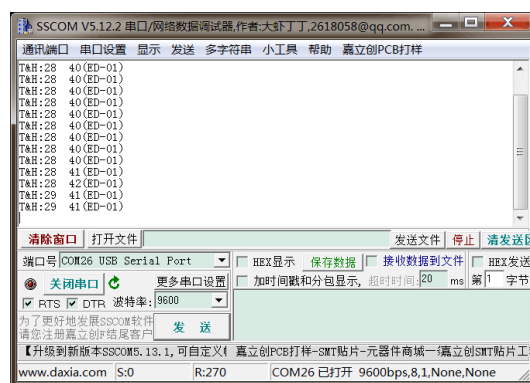


图 17.10 显示收到的数据

(4) 保持二者电源开关打开, 并保持协调器节点位置不变, 一人移动终端节点位置不断远离协调器节点, 移动过程中另一人随时观察 SSSCOM 串口调试助手接收数据的情况;

(5) 当在 SSSCOM 串口调试助手中观察到协调器节点接收不到数据时, 停止终端节点的移动。此时表明二者距离过远, 连接断开, 记录二者之间距离; (需注意, 实验环境的不同会导致该距离不同, 作者在户外晴天测试时该距离约为 40 米。)

(6) 保持二者位置不变, 并关闭二者的电源开关, 将第(2)步中准备的路由器节点放置在协调器和终端节点之间;

(7) 依次打开协调器、路由器、终端节点的电源开关, 等待数秒后协调器的绿灯开始闪烁, 并且 SSSCOM 串口调试助手中显示收到数据。此时路由器节点起到了数据包的转发作用, 即扩大了整个网络的范围。

4.3 多路由多终端网络组建

本节实验使用多个路由和终端节点组建大范围监测网络, 需多人共同完成。步骤如下:

(1) 根据 4.1 节内容准备好各个节点;

(2) 在关闭各节点电源开关的情况下, 按照图 17.1 所示网络拓扑部署各个节点; (需注意, 读者可根据实际情况改变节点布局、增加或减少节点个数; 并且两两节点之间的距离不能超过节点的通信距离, 通信距离可参考 4.2 节第(5)步中记录的距离。)

(3) 依次打开协调器节点、路由器节点以及终端节点的电源开关;

(4) 等待网络稳定后, 在 SSSCOM 串口调试助手中可观察到不同终端节点传输回的数据。

4.4 对比性实验环境下实验

本节实验在 4.3 节基础上进行对比性实验, 验证系统准确性和可靠性, 步骤如下:

(1) 保持 4.3 节中的网络为稳定状态;

(2) 在某个终端节点处, 一人近距离对着温湿度传感器持续呼气;

(3) 持续观察 SSSCOM 串口调试助手接收数据的情况, 发现步骤(2)中的终端节点采集的温湿度数据值都会明显上升。

5 结果分析与讨论

实验结果数据的显示与终端节点数目有关, 由于是简单组网实验, 故并未对终端节点数据加以区分, 实际操作时可选择特定实验环境加以区别。而且由于电池能量和信号干扰等因素, 数据到达可能会面临乱序或丢包等情况, 属于正常现象。且由于传感器节点的测量精确度只能控制在一定范围内, 实验数据可能和实际环境真实情况有出入。

6 注意事项

注意在组网过程中, 请依次开启协调器节点、路由节点、终端节点的电源, 且需要等到前者指示灯状态稳定后才可开启后者电源。

无线网络技术教程

实验十八 IEEE 802.15.4 和 ZBR 协议仿真

1 实验要求与目的

- 1人独立完成
- 理解 IEEE 802.15.4 和 ZBR 协议的工作机制
- 利用 NS2 分析 IEEE 802.15.4 和 ZBR 协议

2 实验背景与原理

2.1 实验背景

IEEE 802.15.4 和 ZBR 协议的理论知识详细内容请参考教材第 8 章。

2.2 实验原理

IEEE 802.15.4 协议仿真实验的网络拓扑结构含 7 个节点, 节点编号从 0 到 6, 如图 18.1 所示。ZBR 协议仿真实验网络拓扑结构含 21 个节点, 节点编号从 0 到 20, 如图 18.2 所示。

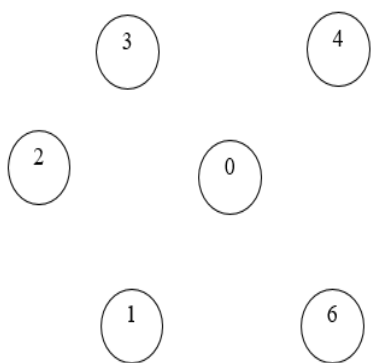


图 18.1 IEEE 802.15.4 协议仿真拓扑



图 18.2 ZBR 协议仿真拓扑

3 实验环境与资源

(1) 环境资源:

树莓派套件、WNT-NS2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;
- 本实验主要利用 NS2 仿真软件, 官方 NS2 版本中已包含 IEEE 802.15.4 协议仿真模块、未包含 ZBR 协议仿真模块。实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;
- 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp18/。

4 实验步骤与演示

4.1 IEEE 802.15.4 仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 18.3 所示。



图 18.3 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册所有实验脚本, 本实验的子文件夹为“exp18”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

`cd WNTLab/experiments/exp18/`

再输入以下命令运行本实验, 如图 18.4 所示。

`ns WPAN_MAC.tcl`

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp18
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp18/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp18$ ns WPAN_MAC.tcl
num_nodes is set 7
INITIALIZE THE LIST xListHead

Traffic: ftp
Acknowledgement for data: on

Starting Simulation...
--- startPANCoord [0] ---
[0.000000](node 0) performing active channel scan
[0.000000](node 0) scanning channel 11
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 35.9
SORTING LISTS ...DONE!
```

图 18.4 运行仿真实验

(4) 利用 NAM 动画观察实验结果, 如图 18.5 所示。注意, 观看 NAM 动画时, 运行的步长调整为 2ms 为宜。

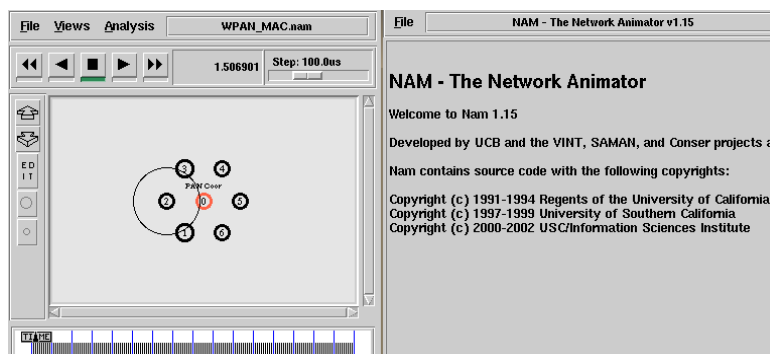


图 18.5 实验结果演示

程序运行说明: 仿真开始时(0~7s), 协调器(PANCoord)节点(0)开始广播控制消息, 以协调节点 1~6 的数据传输。从 7s 开始, 节点 0 与节点 1 建立 TCP/FTP 连接, 并开始数据传输; 7.2s 时, 节点 0 与节点 3 建立 TCP/FTP 连接, 并开始数据传输; 7.4s 时, 节点 0 与节点 5 建立 TCP/FTP 连接, 并开始数据传输。数据传输前, 节点 0 需要通过 RREQ/RREP 建立与节点 1、3、5 的路由。在数据传输过程中, 由于节点 0 与节点 1 持续的传输, 使得其他两条流的传输基本为 0。仿真时间 100s 时, 所有流数据传输结束, 仿真实验结束。

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析, 并绘制图形对比。

4.2 ZBR 协议仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

cd WNTLab/experiments/exp18/

再输入以下命令运行本实验, 如图 18.6 所示。

ns WPAN_ZigBee.tcl

```

wnt-ns2@WNTLab: ~/WNTLab/experiments/exp18
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp18/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp18$ ns WPAN_Zigbee.tcl
num_nodes is set 21
INITIALIZE THE LIST xListHead

Traffic: cbr + poisson
Acknowledgement for data: on

Starting Simulation...
--- startPANCoord [0] ---
[0.000000](node 0) performing active channel scan
[0.000000](node 0) scanning channel 11
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 32.2
SORTING LISTS ...DONE!

```

图 18.6 运行仿真实验

(2) 利用 NAM 动画观察实验结果, 如图 18.7 所示。在观看 NAM 动画时, 运行的步长调整为 2ms 为宜。

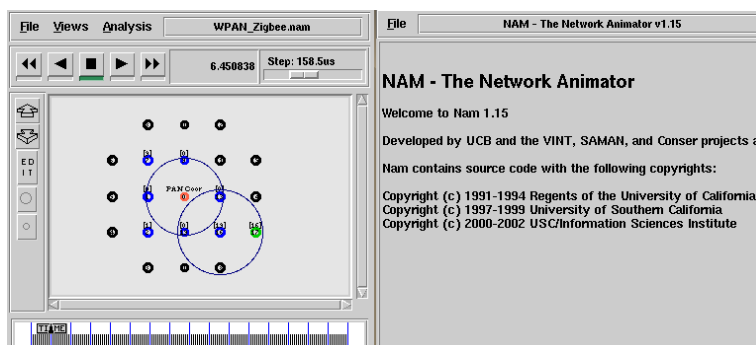


图 18.7 实验结果演示

程序运行说明：仿真开始时(0~10.3s)，网络中所有节点与节点 0 交换信息，以协调数据的传输。10.3s 时，节点 0、2、7 发送 Beacon 消息，同时节点 3 与节点 18 建立 UDP/CBR 数据流，并开始发送数据。10.6s 时，节点 9 与节点 17 建立 UDP/Exponential 数据流，并开始发送数据。在数据传输过程中，链路会断开，此时 ZBR 协议保证了数据传输的可靠性。仿真时间到达 100s 时，所有流传输同时结束，仿真实验结束。

如果读者想进一步分析仿真的深层次结果，可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析，并绘制图形对比。

5 实验扩展与分析

在本实验基础上，读者可自己动手设计不同网络场景下的仿真，分析 IEEE 802.15.4 和 ZBR 协议的性能，分析解读实验相关的程序代码。

6 结果分析与讨论

仿真分析的过程中还需要注意以下几个问题：

- (1) IEEE 802.15.4 和 Zigbee 路由协议具有什么特点？
- (2) IEEE 802.15.4 和 Zigbee 路由协议存在哪些问题？如何解决？

7 注意事项

仿真主要涉及相关脚本，无需修改仿真模块，具体代码参见电子资源中的实验源代码。

实验十九 RFID 和 NFC 数据读写和传输

1 实验要求和目的

- 1 人独立完成
- 了解 RFID 和 NFC 的基本概念和原理
- 掌握 RFID 与 NFC 的应用和实测实验

2 实验原理和背景

2.1 实验背景

射频识别(Radio Frequency Identification, RFID)和近场通信(Near Field Communication, NFC)的背景知识参见教材第9章。

2.2 实验原理

本实验流程: RFID 读卡器接收到来自 B 类卡的数据, 读取卡片序列号, 再通过 BLE 4.0 模块将数据发送到手机 APP 中, 与预设的数据进行对比验证。

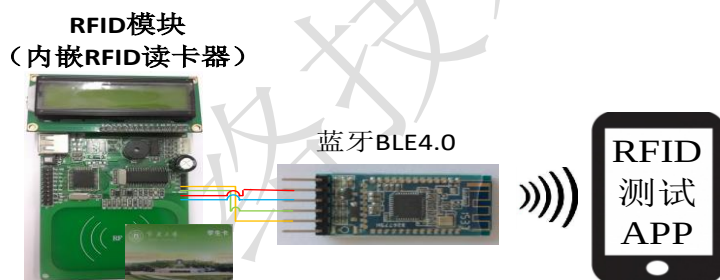


图 19.1 RFID 实验原理图

3 实验环境和资源

3.1 RFID 通信实验

(1) 环境资源:

RFID 读卡器模块 1 个、RFID 卡 1 张、USB 线 1 根、蓝牙 4.0 模块 1 个、母对母杜邦线若干、串口下载线 1 根、USB 转串口线、PC 一台、stc-isp 下载编程烧录软件、RFID 刷卡验证手机 APP、自备安卓手机 1 台。

(2) 使用说明:

- 各类硬件如图 19.2~19.8 所示, 详细的使用方法见实验步骤;
- stc-isp 下载编程烧录软件: 作者使用版本为 6.86 版, 压缩包大小约 7MB, 下载方法请见实验资源包 WNTLab/experiments/exp19, 官网下载连接为 <http://www.stcmcu.com/>。
- RFID 刷卡验证手机 APP 给出相应的 apk 安装包与所有源码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp19。



图 19.2 RFID 读卡器模块



图 19.3 RFID 卡



图 19.4 USB 线



图 19.5 蓝牙 4.0 模块



图 19.6 串口下载线



图 19.7 母对母杜邦线



图 19.8 USB 转串口线

3.2 NFC 读卡通信实验

(1) 环境资源:

NFC 读卡器 1 个、NFC 标签 1 个、USB 转串口模块 1 个、母对母杜邦线若干、PC1 台、SSCOM 串口调试助手。

(2) 使用说明:

- 各类硬件如图 19.9~19.13 所示, 详细的使用方法见实验步骤。

- SSCOM 串口调试助手的官方下载地址为: <http://www.daxia.com/bibis/>, 实验资源包 WNTLab/experiments/exp19 中给出了下载方法与使用说明。



图 19.9 NFC 读卡器



图 19.10 NFC 标签



图 19.11 USB 转串口模块



图 19.12 母对母杜邦线

4 基础实验步骤与演示

4.1 RFID 通信实验

4.1.1 RFID 代码烧录

RFID 实验的代码烧录步骤如下：

(1) 下载并安装完成 stc-isp 下载编程烧录软件；

(2) 连接硬件：将 USB 线一端连接至 PC，一端连接在 RFID 读卡器模块的 USB 接口，用于给模块供电；将 USB 转串口线的 USB 端连接至 PC、串口端连接至串口下载线的串口端，串口下载线的另一端连接在 RFID 读卡器模块的程序下载接口；



图 19.13 硬件连接图

(3) 代码烧录：打开步骤(1)中 stc-isp 软件，选择单片机类型、串口号并打开程序文件(选择 code.hex)，然后点击“下载/编程”进行下载(如图 19.16 所示)。完成后拔掉 RFID 读卡器模块上的所有连线。

(需注意，请按照 RFID 读卡器模块单片机上标注的型号选择烧录时的单片机型号，如图 19.14 红色部分标注有单片机型号；在烧录代码前先关闭模块电源接口旁边的电源开关，点击“下载/编程”按钮开始烧录后再打开开关。)



图 19.14 单片机型号

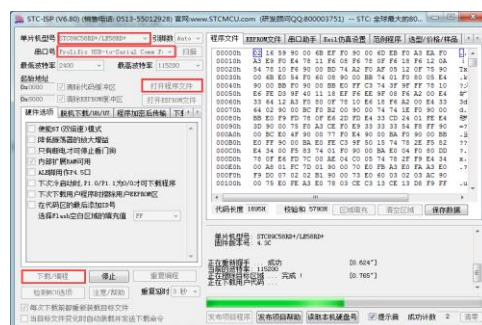


图 19.15 代码烧录

4.1.2 RFID 通信验证

烧录代码完成后在验证 RFID 通信时, 读者需要自备安卓智能手机, 手机中安装本实验提供的 RFID 刷卡验证 APP。具体步骤如下:

(1) 根据 3.1 节介绍, 在手机中下载 RFID 刷卡验证 APP 的安装包(可通过 QQ 从 PC 传给手机), 点击安装并等待安装完成, 如图 19.17 和 19.18 所示。(需注意, 该 APP 要求手机的系统为 Android 5.0 及其以上版本。)



图 19.16 开始安装



图 19.17 安装完成

(2) 安装完成后, 将本实验资源包中的 SmartAttendance 文件夹复制到手机本地存储空间中, 如图 19.9 所示。(需注意, 此过程中, 读者不要修改该文件夹名称。)



图 19.18 复制该文件夹到手机本地存储空间

(3) 打开该 APP, APP 会请求获取存储权限, 读者需要允许该权限, 如图 19.20 所示。在主页中点击“导入信息”按钮, 会从 APP 自带的文件中导入部分用户信息, 成功后下方会

提示导入成功的信息, 如图 19.21 所示。



图 19.19 允许使用存储权限



图 19.20 导入用户信息

(3) 将 RFID 读卡器模块与蓝牙 4.0 模块按照如图 19.22 所示方式连接, 并使用 USB 线连接 RFID 读卡器模块与 PC。

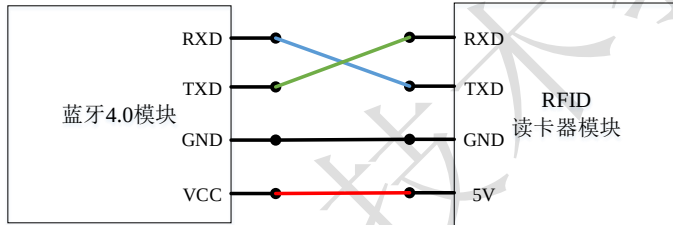


图 19.21 RFID 读卡器模块与蓝牙 4.0 模块连接原理图

(4) 打开模块电源开关, 将 RFID 卡放在模块的读卡区域, 此时显示区域会显示卡号, 如图 19.23 所示, 记录该卡号以待备用。

(5) 点击主页“卡号信息”按钮, 进入信息列表, 再点击某一用户信息, 如图 19.24 所示, 将卡号一项修改为步骤(4)中记录的卡号。



图 19.22 记录 RFID



图 19.23 导入用户信息

(6) 点击主页“开始刷卡”按钮, 进入课程签到界面时, APP 会请求获取蓝牙权限, 读者需要允许该权限, 如图 19.25 所示。允许后点击右上角按钮搜索蓝牙设备并选择蓝牙 4.0 模块对应的信号进行连接, 如图 19.26 所示。



图 19.24 请求蓝牙权限

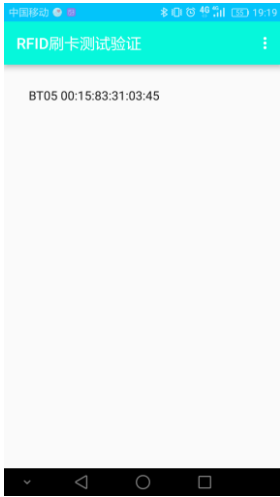


图 19.25 搜索并连接蓝牙信号

(7) 连接到蓝牙后, 进入测试验证界面, 如图 19.27 所示; 点击“开始签到”按钮, 如图 19.28 所示; 将 RFID 卡放在 RFID 读卡器模块的读卡区域, 此时 APP 中返回验证人的姓名, 验证成功, 如图 19.29 所示。



图 19.26 测试验证界面



图 19.27 开始签到



图 19.28 验证成功

4.2 NFC 读卡通信实验

(1) 将 NFC 读卡器模块与 USB 转串口模块按照如图 19.29 所示的方式连接, 并将 USB 转串口模块插入 PC 的 USB 接口中;

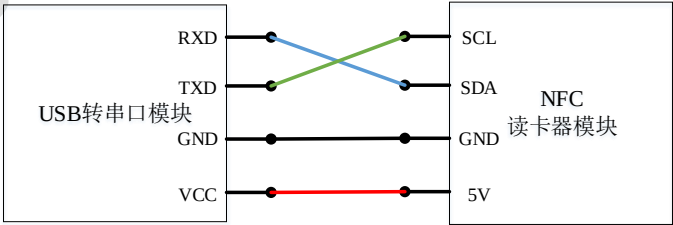


图 19.29 NFC 读卡器模块与 USB 转串口模块连接原理图

(2) 下载并安装完成 SSCOM 串口调试助手, 如图 19.30 所示;

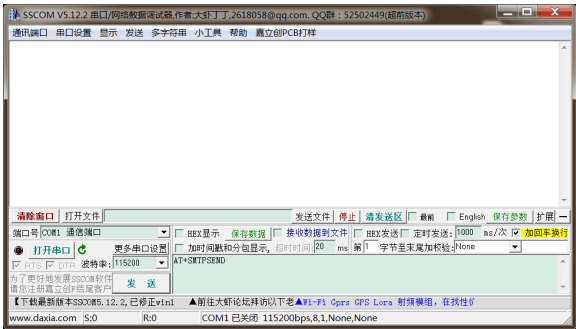


图 19.30 SSSCOM 串口调试助手

(3) 打开 ssscom 串口软件, 勾选“HEX 显示”、“加时间戳和分包显示”和“HEX 发送”三个复选框, 并选择波特率为 115200, 选择正确的端口号后点击“打开串口”按钮, 如图 19.31 所示。

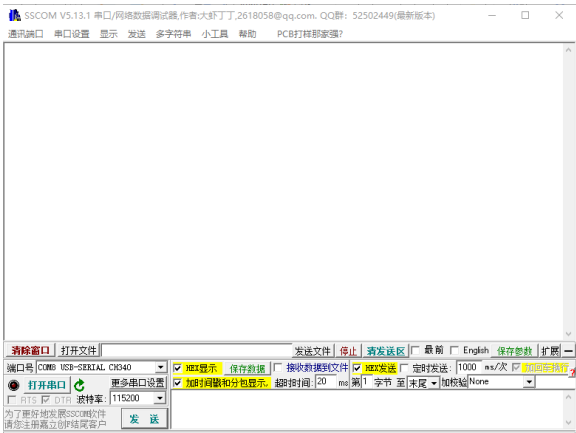


图 19.31 设置完成后的 SSSCOM 串口调试助手

(4) 向 NFC 模块发送指令串“55 55 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ff 03 fd d4 14 01 17 00”以激活 NFC 主控板, 串口软件收到反馈信息, 如图 19.32 所示。



图 19.32 激活 NFC 模块

(5) 向 NFC 主板发送指令串“00 00 FF 04 FC D4 4A 02 00 E0 00”来获取 NFC 卡的 UID, UID 是每张 NFC 卡的唯一 ID 号。串口软件接收到 NFC 主控板反馈的信息“00 00 FF 00 FF 00”, 系统进入读卡模式。这时把 NFC 卡放在 NFC 模块上, 串口软件接收到卡的 UID, 如图 19.37 所示。

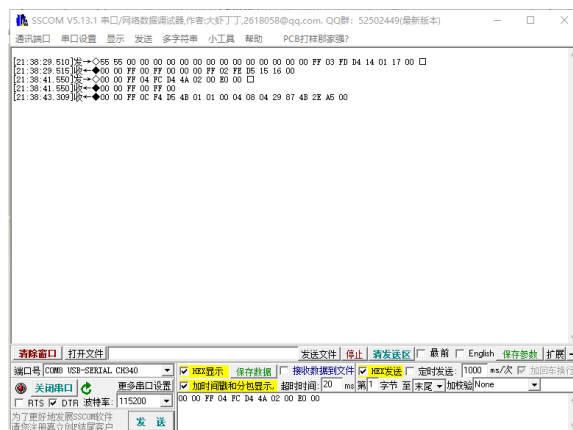


图 19.37 NFC 模块读卡

5 结果分析与讨论

本实验旨在让读者了解 RFID 与 NFC 技术在实际生活中的相关应用, 实验只是对 RFID 和 NFC 的烧录及测试做了详细说明, 并没有对实际开发做描述, 有兴趣的同学还可研究单片机工作的程序代码, 自行推敲学习。

6 注意事项

当 RFID 和 NFC 的 IC 卡或标签靠近读卡器进行读卡时, 串口调试软件却没有任何显示时, 检查是否附近有过强的电磁干扰, 插入 USB 接口后, 让读卡器尽量远离笔记本或电脑主机。用杜邦线接 RFID 与 BLE 4.0 模块, NFC 读写模块与 USB 转串口模块请勿接错。此外, 实验中杜邦线容易脱落, 注意不要生拉硬拽, 造成脱落。

实验二十 树莓派安装和配置 WiFi 路由器

1 实验要求与目的

- 1 人独立完成
- 掌握树莓派卡片电脑配置
- 熟悉树莓派操作系统基本配置
- 掌握树莓派配置无线 AP 的步骤

2 实验背景与原理

本实验在树莓派(背景内容见教材第9章)上安装操作系统, 并完成相应配置使树莓派成为1个带DNS服务的无线路由器。首先, 准备1张TF卡, 将树莓派操作系统 Raspbian 用 Win32DiskImager 烧录到 TF 卡上。然后将 TF 卡和 USB 无线网卡插入并启动树莓派, 进行相关配置。最后进行网络设置, 并安装 DNS 服务、无线共享软件、DHCP 服务。

3 实验环境和资源

(1) 环境资源:

树莓派套件(树莓派 Raspberry Pi3 一个、8GB TF 卡一张、树莓派电源适配器一个、HDMI 转 VGA 线一根、VGA 线一根)、USB 接口的键盘鼠标一套(需自备)、有源显示器一个(需自备)、PC 一台、Ubuntu MATE 系统镜像文件、SDFormatter 软件、Win32DiskImager 软件。

(2) 使用说明:

- 各类硬件如图 20.1~20.4 所示, 详细的使用方法见实验步骤;
- Ubuntu MATE: 16.04.2 为官方长期维护的免费版本, 专供树莓派使用, 下载后的压缩包大小约为 1.2GB, 详细下载方法请见实验资源包 WNTLab/experiments/exp20, 官网地址 <https://ubuntu-mate.org/>;
- SDFormatter 软件: 作者使用版本为 4.0 版, 安装包大小约为 7MB, 下载方法请见实验资源包 WNTLab/experiments/exp20, 官网下载地址为 <http://www.softpedia.com/get/System/File-Management/SDFormatter.shtml>;
- Win32DiskImager 软件: 作者使用版本为 1.0 版, 安装包大小约为 12MB, 下载方法请见实验资源包 WNTLab/experiments/exp20, 官网下载地址为 <https://sourceforge.net/projects/win32diskimager/>;

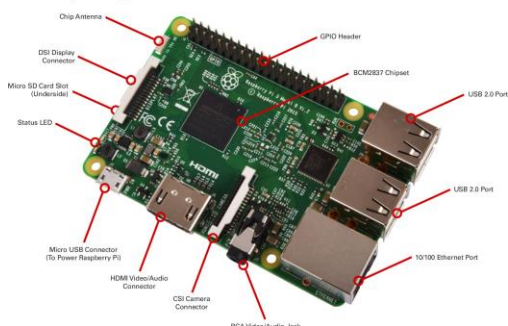


图 20.1 树莓派 Raspberry Pi3



图 20.2 树莓派电源适配器



图 20.3 HDMI 转 VGA 线



图 20.4 8GB TF 卡

4 实验步骤与演示

4.1 实验准备

树莓派系统安装过程如下：

(1) 下载官方新版 Raspbian 系统后，解压得到 img 格式的系统镜像；

(2) 将 TF 卡插入 PC，使用 SDFormatter 软件格式化 TF 卡，如图 20.5 所示。格式化完成后，开启 Win32DiskImager 软件，将步骤(1)中下载的系统镜像烧录到 TF 卡中，如图 20.6 所示。“Image File”选择解压后的 img 系统镜像文件，“Device”选择 TF 卡，然后点击“Write”按钮开始烧录系统，期间出现对话框选择是。出现“Write Successful”提示框时，说明系统烧录成功。



图 20.5 格式化 TF 卡

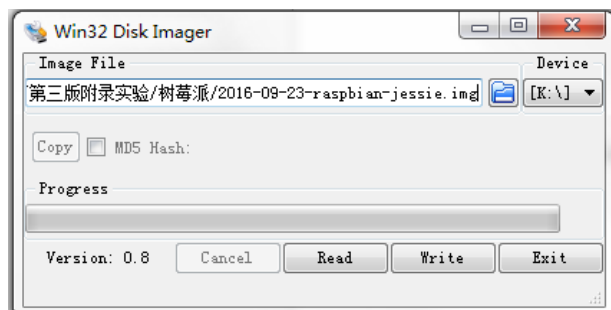


图 20.6 Win32DiskImager 界面

(3) 将烧录好的 TF 卡插入树莓派模块背面，如图 20.7 所示，并连接键盘、鼠标，用 VGA 线和 HDMI 转 VGA 线连接显示器和树莓派。接上电源线，开启电源，并且使用网线将树莓派连接至 Internet，如图 20.8 所示。



图 20.7 插入 TF 卡



图 20.8 连接树莓派

(4) 等待系统启动成功, 会出现初试配置界面, 依次设置: 系统语言(建议选择英文); 无线网络(选择不连接); 键盘布局(选择默认美式键盘布局); 位置信息(选择中国任意地区皆可); 用户信息(设置用户名、主机名、密码等)。设置完成后重启系统。

(5) 系统重启后, 使用快捷键“Ctrl+Alt+T”打开终端(Terminal), 使用 ping 命令检查树莓派是否通过网线连接到 Internet, 若收到如图 20.9 所示类似内容, 则代表树莓派已连接到网络。(需注意, 本实验中所有命令均在终端中运行。)

```
wnt@wnt-desktop:~$ ping www.baidu.com
PING www.a.shifen.com (115.239.211.112) 56(84) bytes of data:
64 bytes from 115.239.211.112: icmp_seq=1 ttl=52 time=12.2 ms
64 bytes from 115.239.211.112: icmp_seq=2 ttl=52 time=6.90 ms
64 bytes from 115.239.211.112: icmp_seq=3 ttl=52 time=7.37 ms
64 bytes from 115.239.211.112: icmp_seq=4 ttl=52 time=6.01 ms
64 bytes from 115.239.211.112: icmp_seq=5 ttl=52 time=5.75 ms
64 bytes from 115.239.211.112: icmp_seq=6 ttl=52 time=7.37 ms
64 bytes from 115.239.211.112: icmp_seq=7 ttl=52 time=5.98 ms
^C
--- www.a.shifen.com ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6009ms
rtt min/avg/max/mdev = 5.753/7.381/12.268/2.091 ms
```

图 20.9 树莓派的系统桌面

4.2 配置无线网卡

第 3 代树莓派自带无线网卡, 利用命令: `ifconfig`, 查看系统网卡信息, 如图 20.10, 其中 wlan0 即为无线网卡, 下面将对该网卡进行配置。

```
wnt@wnt-desktop:~$ sudo ifconfig
[sudo] password for wnt:
enxb827ebde4831 Link encap:Ethernet HWaddr b8:27:eb:de:48:31
  inet addr:192.168.1.157 Bcast:192.168.1.255 Mask:255.255.255.0
  inet6 addr: fe80::a10f:bcb8:3467:d9d0/64 Scope:Link
  UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
  RX packets:12003 errors:0 dropped:0 overruns:0 frame:0
  TX packets:23676 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:708193 (708.1 KB) TX bytes:32560547 (32.5 MB)

lo
  Link encap:Local Loopback
  inet addr:127.0.0.1 Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING MTU:65536 Metric:1
  RX packets:28651 errors:0 dropped:0 overruns:0 frame:0
  TX packets:28651 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1
  RX bytes:88436135 (88.4 MB) TX bytes:88436135 (88.4 MB)

wlan0
  Link encap:Ethernet HWaddr b8:27:eb:8b:1d:64
  UP BROADCAST MULTICAST MTU:1500 Metric:1
  RX packets:364 errors:0 dropped:364 overruns:0 frame:0
  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:129758 (129.7 KB) TX bytes:0 (0.0 B)
```

图 20.10 系统网卡信息

(1) 首先需要设置 wlan0 为静态 IP, 在终端中执行命令: `sudo nano /etc/network/interfaces`, 打开网络配置文件, 添加如下 wlan0 的配置内容:

```
allow-hotplug wlan0
```



```

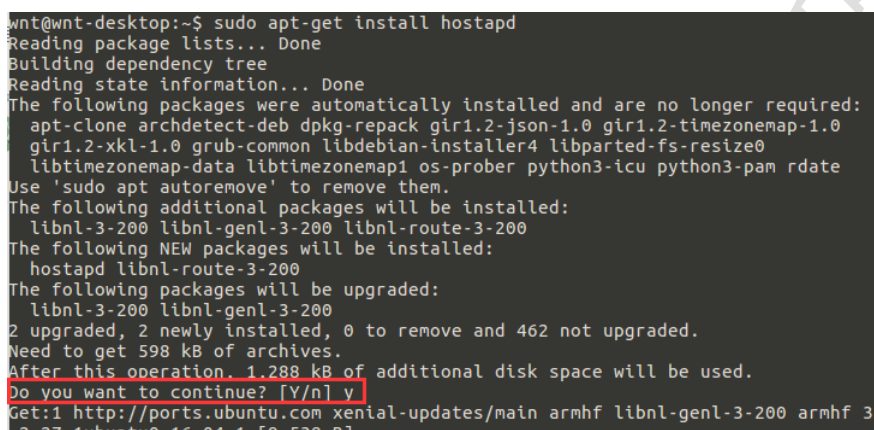
iface wlan0 inet static
address 192.168.100.1
netmask 255.255.255.0
network 192.168.100.0
broadcast 192.168.100.255

```

- (2) 执行命令：`sudo ifup wlan0`，更新重启无线网卡 wlan0 的配置。
- (3) 最后，再次执行命令 `ifconfig`，观察并确认 wlan0 的 IP 已设置好。

4.3 安装与配置 hostapd

(1) 安装 hostapd。在终端中执行命令：`sudo apt-get install hostapd`，安装期间会提示是否继续，输入“Y”并按回车继续安装。安装过程如图 20.11 所示。



```

wnt@wnt-desktop:~$ sudo apt-get install hostapd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  apt-c-clone archdetect-deb dpkg-repack gir1.2-json-1.0 gir1.2-timzone-map-1.0
  gir1.2-xml-1.0 grub-common libdebian-installer4 libparted-fs-resize0
  libtimzone-map-data libtimzone-map1 os-prober python3-icu python3-pam rdate
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libnl-3-200 libnl-genl-3-200 libnl-route-3-200
The following NEW packages will be installed:
  hostapd libnl-route-3-200
The following packages will be upgraded:
  libnl-3-200 libnl-genl-3-200
2 upgraded, 2 newly installed, 0 to remove and 462 not upgraded.
Need to get 598 kB of archives.
After this operation, 1.288 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ports.ubuntu.com xenial-updates/main armhf libnl-genl-3-200 armhf 3
2-27ubuntu0.16.04.1 [50,528 B]

```

图 20.11 hostapd 安装过程

(2) 配置 hostapd。执行命令：`sudo nano /etc/hostapd/hostapd.conf`，新建 hostapd 的配置文件，添加如下内容：

```

interface=wlan0
driver=nl80211
ssid=RaspberryPi
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=raspberrywifi
rsn_pairwise=CCMP

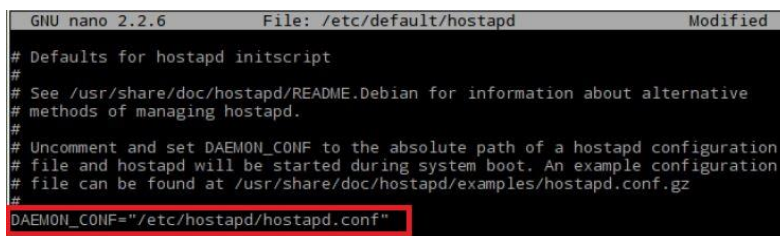
```

其中，`interface=wlan0`，指定网卡名称为 wlan0；`driver=nl80211`，指使用 nl80211 驱动；`ssid=RaspberryPi` 为热点名称，可根据实际情况修改；`hw_mode=g`，指网卡工作在 802.11G 模式；`channel=6`，指无线网卡选用 6 信道；`wpa_passphrase=raspberrywifi` 为热点密码（8 位以上），可根据实际情况修改。正确修改后，按下“Ctrl+X”关闭文件，输入“Y”并按回车键

保存。

(3) 测试 hostapd 配置。执行命令 `sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf`, 测试上述配置是否正确。若配置无误, 可在 PC 或手机中搜索到上述配置的无线路由器信号, 此处无需连接该信号, 测试无误后在终端中使用快捷键 Ctrl+C 退出测试。

(4) 修改 hostapd 配置文件路径。执行命令 `sudo nano /etc/default/hostapd`, 进入 hostapd 默认配置文件后找到 `#DAEMON_CONF= ""`, 去掉注释符号“#”并作如下修改: `DAEMON_CONF= "/etc/hostapd/hostapd.conf"`, 最后退出并保存, 如图 20.12 所示。



```

GNU nano 2.2.6      File: /etc/default/hostapd      Modified
# Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about alternative
# methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configuration
# file and hostapd will be started during system boot. An example configuration
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"

```

图 20.12 修改 hostapd 配置文件

4.3 安装与配置 dnsmasq

接下来, 将 DNS 服务架设在树莓派上。常用 DNS 查询将会缓存到树莓派上, 能起到 DNS 查询加速、提升上网体验的作用。

(1) 安装 dnsmasq。执行命令 `sudo apt-get install dnsmasq`, 进行安装并完成。

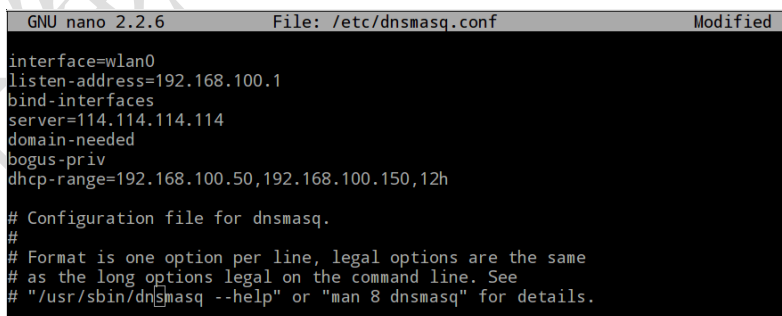
(2) 配置 dnsmasq。执行命令 `sudo nano /etc/dnsmasq.conf`, 打开 dnsmasq 的配置文件, 在其中添加以下内容:

```

interface=wlan0
listen-address=192.168.100.1
bind-interfaces
server=114.114.114.114
domain-needed
bogus-priv
dhcp-range=192.168.100.50,192.168.100.150,12h

```

添加后的文件内容如图 20.13 所示。



```

GNU nano 2.2.6      File: /etc/dnsmasq.conf      Modified
interface=wlan0
listen-address=192.168.100.1
bind-interfaces
server=114.114.114.114
domain-needed
bogus-priv
dhcp-range=192.168.100.50,192.168.100.150,12h

# Configuration file for dnsmasq.
#
# Format is one option per line, legal options are the same
# as the long options legal on the command line. See
# "/usr/sbin/dnsmasq --help" or "man 8 dnsmasq" for details.

```

图 20.13 修改 dnsmasq 配置文件

其中, server 为上级 DNS 服务器地址, 即本地缓存未找到的域名将会发送给上级 DNS 服务器, 然后本地再缓存结果; cache-size 为缓存大小。正确修改后, Ctrl+X 关闭文件, 输入“Y”并按回车键保存。

(3) 开启 NAT 功能。正式配置 NAT 功能前, 需开启系统的 IP 转发功能, 执行命令: `sudo nano /etc/sysctl.conf` 进入配置文件, 如图 20.14 所示, 去掉 `net.ipv4_forward=1` 前面的“#”号, 开启 IP 转发功能。


```

GNU nano 2.2.6      File: /etc/sysctl.conf      Modified
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
# Enabling this option disables Stateless Address Autoconfiguration
# based on Router Advertisements for this host
#net.ipv6.conf.all.forwarding=1

```

图 20.14 开启 ipv4 转发功能

(4) 开启 IP 转发功能后, 紧接着配置 iptables 防火墙, 依次执行下列 3 句命令:

- `sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE`
- `sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED, ESTABLISHED -j ACCEPT`
- `sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT`

(5) 至此, NAT 功能已启动。执行以下命令, 以保存 iptables 配置:

`sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"`

(6) 然后将上述 iptables 的配置设置为开机自动加载, 执行命令: `sudo nano /etc/rc.local`, 进入文件后在 `exit 0` 上方添加内容: `iptables-restore < /etc/iptables.ipv4.nat`, 修改后的文件如图 20.15 所示。

```

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

iptables-restore < /etc/iptables.ipv4.nat
exit 0

```

图 20.15 设置 iptables 配置开机自动加载

(7) 最后, 分别执行以下命令, 启动 hostapd 和 dnsmasq 服务:

`sudo service hostapd start`
`sudo service dnsmasq start`

(8) 至此, 基于树莓派的无线 AP 已配置完成, 执行命令: `sudo reboot`, 重启树莓派, 待重启完成后, 便会出现一个可用的无线 AP, 其中 AP 名和密码都是在文件 `/etc/hostapd/hostapd.conf` 中所设定的。

5 结果分析与讨论

在 PC 端连接树莓派路由器的无线网络信号, 查看无线配置信息, 如图 20.16。由其可知, ipv4 地址为 192.168.100.61, 在设置的 192.168.100.50-192.168.100.150 之间, 子网掩码和网关也与之之前配置一致。

```
无线局域网适配器 WLAN:  
    连接特定的 DNS 后缀 . . . . . :  
    本地链接 IPv6 地址. . . . . : fe80::7da9:4c7f:3bfl:d542%4  
    IPv4 地址 . . . . . : 192.168.100.61  
    子网掩码 . . . . . : 255.255.255.0  
    默认网关. . . . . : 192.168.100.1
```

图 20.16 无线配置信息

ping 任意可以访问的网址，结果如图 20.17 所示，网络已可用。

```
C:\Users\DK>ping www.nbu.edu.cn  
  
正在 Ping www.nbu.edu.cn [10.22.100.94] 具有 32 字节的数据:  
来自 10.22.100.94 的回复: 字节=32 时间=7ms TTL=124  
来自 10.22.100.94 的回复: 字节=32 时间=8ms TTL=124  
来自 10.22.100.94 的回复: 字节=32 时间=14ms TTL=124  
来自 10.22.100.94 的回复: 字节=32 时间=8ms TTL=124  
  
10.22.100.94 的 Ping 统计信息:  
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),  
    往返行程的估计时间(以毫秒为单位):  
        最短 = 7ms, 最长 = 14ms, 平均 = 9ms
```

图 20.17 ping 命令执行结果

6 注意事项

本实验步骤较多，读者在实验过程中应仔细按照步骤进行，注意每一条命令中的空格符。

实验二十一 Contiki 系统测试与组网基础

1 实验要求与目的

- 1 人独立完成
- 了解 Contiki 系统的基础知识
- 了解 ibc 协议通信的基本原理
- 掌握 Contiki 下代码编译和使用过程

2 实验背景与原理

2.1 实验背景

Contiki 是一个面向物联网的嵌入式操作系统(见教材第 9 章), 所需硬件要求门槛较低。ibc 全称是 identified broadcast, 属于 rime 协议栈的一种底层协议。ibc 协议的功能是向本地邻居节点广播数据包, 该数据包中添加了发送节点的地址作为一个包属性, 因此接收节点可以根据收到的数据包来识别发送方的地址。

2.2 实验原理

将 Contiki 系统中的 ibc 协议例程经过编译、烧录到 CC2530 节点后, ibc 协议开始工作。先开启的 CC2530 根节点向其通信覆盖范围内所有节点发送广播包, 处在通信范围内的节点接收到根节点的广播包后, 解析广播包, 读取根节点的数据, 包含根节点的地址等。

3 实验环境与资源

(1) 环境资源:

PC 一台、CC2530 传感器节点套件两套及软件环境(详见实验十五第 3 节)、wnt-contiki 仿真环境、SSCOM 串口调试助手软件。

(2) 使用说明:

- CC2530 传感器节点套件介绍见实验十五;
- 本实验主要用到仿真实验平台中的 Contiki 操作系统。实验一中已提供了实验平台搭建方法, 如果已完成实验一中 wnt-contiki 仿真环境的搭建, 本实验可直接运行;
- SSCOM 串口调试助手的官方下载地址为: <http://www.daxia.com/bibis/>, 详细下载方法及软件简介请见实验资源包 WNTLab/experiments/exp21;

4 实验步骤和演示

具体实验过程如下:

(1) 打开 wnt-contiki 虚拟机。

(2) 在“home/wnt-contiki/WNTLab/experiments”目录下已包含本手册的所有 contiki 相关实验的资源, 本实验二十一的子文件夹为“exp21”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入 rime 协议栈例程目录:

```
cd contiki/examples/rime/
```

可以执行以下命令, 查看当前目录下的文件信息, 如图 21.1 所示。

```
ls -l
```

```
wnt-contiki@wnt-contiki: ~/contiki/examples/rime
=====Welcome to use Contiki for WNTLab!=====
If you have any questions, please contact wireless_net AT 163.com
wnt-contiki@wnt-contiki:~$ cd contiki/examples/rime/
wnt-contiki@wnt-contiki:~/contiki/examples/rime$ ls -l
total 484
-rw-r--r-- 1 wnt-contiki user 2952 Mar 13 03:22 contiki-cc2530dk.lib
-rw-rw-r-- 1 wnt-contiki user 2963 Aug 25 2015 example-abc.c
-rw-rw-r-- 1 wnt-contiki user 3723 Aug 25 2015 example-announcement.c
-rw-rw-r-- 1 wnt-contiki user 3160 Aug 25 2015 example-broadcast.c
-rw-r--r-- 1 wnt-contiki user 130142 Mar 13 03:22 example-broadcast.cc2530dk
-rw-rw-r-- 1 wnt-contiki user 8831 Aug 25 2015 example-broadcast.csc
-rw-r--r-- 1 wnt-contiki user 130142 Mar 13 03:22 example-broadcast.hex
-rw-r--r-- 1 wnt-contiki user 377 Mar 13 03:22 example-broadcast.lk
-rw-r--r-- 1 wnt-contiki user 70323 Mar 13 03:22 example-broadcast.map
-rw-r--r-- 1 wnt-contiki user 1129 Mar 13 03:22 example-broadcast.mem
-rw-rw-r-- 1 wnt-contiki user 4174 Aug 25 2015 example-collect.c
-rw-rw-r-- 1 wnt-contiki user 13246 Aug 25 2015 example-collect.csc
-rw-rw-r-- 1 wnt-contiki user 3526 Aug 25 2015 example-mesh.c
-rw-rw-r-- 1 wnt-contiki user 9320 Aug 25 2015 example-multihop.c
-rw-rw-r-- 1 wnt-contiki user 10411 Aug 25 2015 example-neighbors.c
-rw-rw-r-- 1 wnt-contiki user 1265 Aug 25 2015 example-relay.c
```

图 21.1 rime 协议栈目录

(4) 执行命令: **make TARGET=cc2530dk example-broadcast**, 编译完成后的信息如图 21.2 所示。

```
wnt@wnt: ~/contiki/examples/rime
example-broadcast.c:84: warning 85: in function process_thread_example_broadcast
_process unreferenced function argument : 'data'
LD
example-broadcast.ihx
packihx example-broadcast.ihx > example-broadcast.hex
packihx: read 1547 lines, wrote 2996: OK.
cp example-broadcast.hex example-broadcast.cc2530dk

Report
=====
Code footprint:
Area                               Addr             Size             Decimal
-----
HOME, CSEG, CONST, XINIT, GS*      00000000         0000B84E =      47182. bytes (R
EL, CON, CODE)
Other memory:
Name                               Start            End              Size             Max
-----
PAGED EXT. RAM                    0x0000           0x0b14           2837             7936
EXTERNAL RAM                       0x0000           0xb84d          47182            65536
ROM/EPROM/FLASH                   0x0000           0xb84d          47182            65536
rm obj_cc2530dk/example-broadcast.app.rel example-broadcast.ihx
wnt@wnt:~/contiki/examples/rime$
```

图 21.2 编译 ibc

(5) 将编译好的 example-broadcast.hex 文件拷贝到 Windows 桌面上, 如图 21.3 所示。

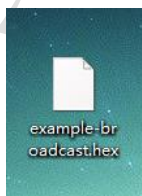


图 21.3 拷贝 .hex 文件

(6) 连接 CC Debugger 调试器, 将下载线和 USB A 型转 B 型线分别与 CC Debugger 调试器相连, 如图 21.4 所示:



图 21.4 连接 CC Debugger 调试器

(7) 连接 CC Debugger 调试器完成后, 将下载线一端连接到 CC2530 节点, 如图 21.5 所示, 将 USB 一端连接至 PC 的 USB 口; 使用 Mini USB 线连接 CC2530 节点的 Mini USB 口与 PC 的 USB 口, 按下 CC Debugger 的 reset 键, 显示灯为绿色, 连接完成后如图 21.6 所示:

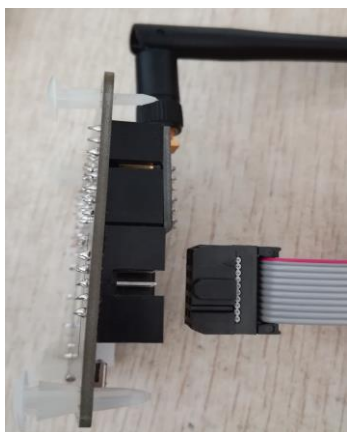


图 21.5 下载线与 CC2530 节点连接



图 21.6 节点与 PC 的连接

(7) 打开 SmartRF Flash Programmer, 在文件选择中选择 example-broadcast.hex, 点击 Platform actions, 开始对 CC2530 节点烧录代码, 实验中对两个 CC2530 节点均烧录此代码, 如图 21.7 所示。

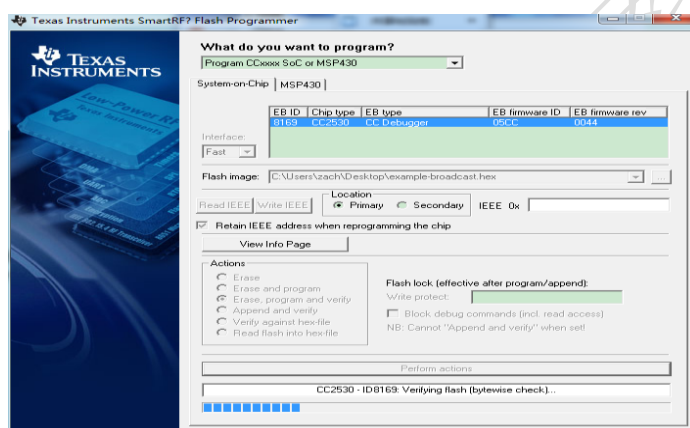


图 21.7 开始对 CC2530 节点烧录

(8) 使用 Mini USB 线连接 CC2530 节点与 PC, 两个 CC2530 节点均按照如图 21.8 所示方式连接。



图 21.8 CC2530 节点与 PC 连接

(9) 打开 SSCOM 串口调试助手, 选择正确端口号, 设置波特率为 115200, 并打开串口, 打开 2 个 CC2530 节点的电源开关, 稍等片刻, 数据会在串口软件右侧显示, 如图 21.9 所示。

此时两个 CC2530 节点相互发送广播包, 并且收到广播包后, 解析广播包, 读取根节点的数据, 包含根节点的地址等信息。

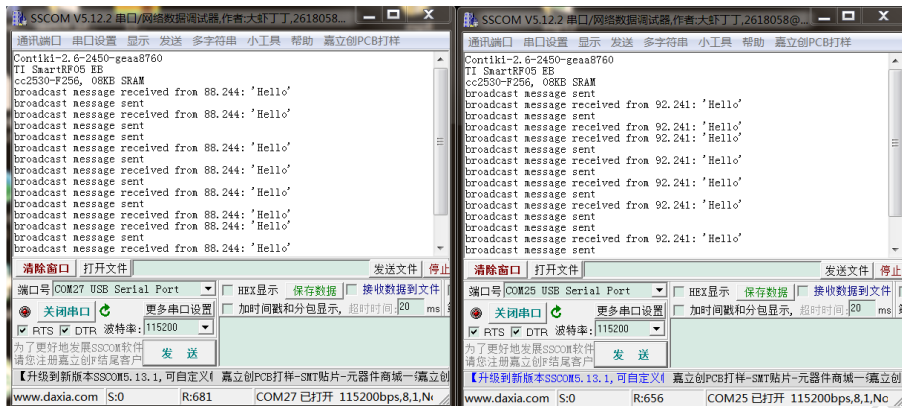


图 21.9 串口助手显示 ibc 广播包的内容

5 实验扩展和分析

本实验只采用 ibc 进行简单的演示, 感兴趣的读者可进行其他应用的测试, 例如匿名广播(abc)。此外, 读者也可设计多个节点的通信, 并进行分析。

6 结果分析和讨论

本实验介绍了 Contiki 系统的测试。此外, 利用 ibc 进行数据广播, 并借助大傻串口助手查看数据。

7 注意事项

实验的主要目的是介绍如何进行 contiki 系统的测试, 其中涉及一些 Linux 的基本操作, 需要读者有一定 Linux 基础。

在实验过程中, 需注意: 编译需要进入对应的目录下, 否则无法编译; 编译完成后, 可将生成的.hex 文件拷贝至 Windows 系统中借助 smartRF 进行烧录。在 CC2530 节点代码的烧录过程中, 烧录线要插紧插槽, reset 键要多按几下, 不然烧录过程会出现其他问题。

实验二十二 CoAP 协议仿真实验

1 实验要求与目的

- 1 人独立完成
- 理解 CoAP 协议的基本原理和实现
- 利用 Cooja 仿真器分析 CoAP 协议

2 实验背景与原理

2.1 实验背景

物联网的典型应用层协议是受限应用协议(Constrained Application Protocol, CoAP), 主要面向受限节点和低功耗有损网络环境的特制应用层 Web 传输, 类似因特网中的 HTTP。具体协议原理, 请参考教材第 9 章。

Cooja 是 Contiki 提供的仿真平台, 随 Contiki 系统安装, 基于 Java 语言开发, 用来模拟运行在 Contiki 操作系统上的网络。Cooja 具有图形化界面, 能观察节点布局、消息和数据的发送过程等功能。Cooja 中仿真节点与实际生活中使用节点相似性极高, 能较好模拟节点在现实生活中的实际运行状态。仿真器用 Java 语言执行, 但传感器节点的具体行为需用 C 语言编写。Cooja 的一个显著特点就是能够同时结合三个不同层次: 网络层、操作系统层和机器编码指令层。

Cooja 仿真器的界面如图 22.1 所示, 包含 5 部分。其中, Network 窗口主要显示网络节点的物理布局, 可根据需要添加节点, 并任意安排其位置。当需加入很多节点时, Cooja 也提供了一些布局方式供选择, 如线性、圆形、随机布局等; Simulation Control 窗口提供控制仿真的选项, 如开始、暂停、重新载入等, 也可控制仿真速度, 选择 Step 则会控制仿真单步进行, 方便观察。也可在 Speed Limit 菜单下选择不同仿真速度上限, 如 1%、10%、100% 等; Mote Output 窗口会显示所有节点输出及其他运行信息, 需显示的内容来自源代码中的 "printf" 命令。位于底部的 Filter 区域, 可过滤不需要的节点信息, 如仿真只想观察 Mote2 节点的输出, 就可在该区域中输入 "ID: 2", 那么得到的结果就都是 Mote2 的相关信息; Timeline 窗口可显示每个节点随仿真进行而发生的事件, 如无线流量、LED 活动情况等; Notes 窗口主要用来临时记录仿真过程中的笔记, 协助仿真分析。

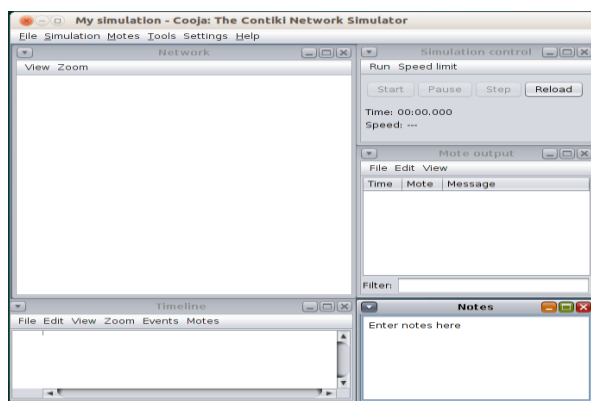


图 22.1 Cooja 仿真器界面

2.2 实验原理

本实验采用 Contiki 系统的 Cooja 仿真器。创建了 1 个 CoAP 服务器节点和一个边界路由节点, 并采用 tunslip6 连通边界路由和本地主机, 使主机能访问 CoAP 服务器节点。最后, 通过主机上的火狐浏览器对 CoAP 服务器发起访问请求。本实验中边界路由的功能是路由 RPL 网络和外部网络之间的数据, 即 CoAP 服务器节点和本地主机。

Contiki 中 tunslip6 的主要用途是为边界路由器赋予网络号, 即例子中的 `aaaa::1/64`, 它的原理是:

- (1) 在主机上为 tun 虚拟网卡配置网络参数 (虚拟网卡设备位于 `/dev/net/tun`, 可以直接用程序读写该设备, tun 建立在数据链路层, 所以读写内容为 IP 数据包);
- (2) 边界路由器 (border-router) 与主机间通信使用 slip 协议, slip 是一个串口 IP 协议;
- (3) tunslip6 程序中包含读串口的代码, 读入内容可写到 tun 设备上;
- (4) tunslip6 程序中包含读 tun 的代码, 可以将 tun 读入内容通过 slip 封装后写到串口。

3 实验环境与资源

(1) 环境资源:

PC 一台、wnt-contiki 仿真环境。

(2) 使用说明:

●本实验主要用到仿真实验平台中的 Contiki 操作系统。实验一中已提供了实验平台搭建方法, 如果已完成实验一中 wnt-contiki 仿真环境的搭建, 本实验可直接运行;

4 实验步骤与仿真演示

实验的具体步骤如下:

(1) 打开终端, 输入“`cd contiki/tools/cooja`”命令, 进入 Cooja 仿真器目录。输入“`ant run`”命令, 运行 Cooja 仿真器。如图 22.2 所示;

(2) 点击 File 按钮, 选择 Open simulation 中的 Browse 进入 Open 界面, 点击 home 按钮, 将目录选至 `/home/wnt-contiki/WNTLab/experiments/exp22/WNTExamples`, 选择 `sky-coap.csc` 点击 open 按钮。如图 22.3 所示。

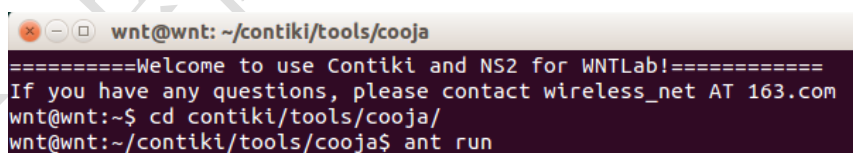


图 22.2 运行 Cooja 仿真器

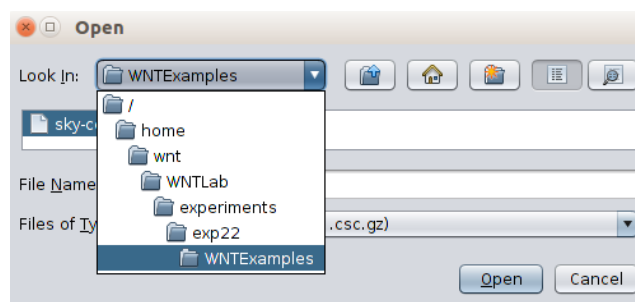


图 22.3 选择实验

(3) 实验加载完成将会出现 4 个窗口, 分别是 Simulation control 控制仿真开始停止或重新加载实验; Network 显示网络节点的物理布局; Mote output 显示节点输出信息; Radio

messages 捕获通信数据包（默认情况下该窗口最小化至 Cooja 界面的左下角）。

(4) 如图 22.4 所示,选择 Radio messages 界面的 Analyzer 菜单栏中的 6LoWPAN Analyzer with PCAP 选项,该功能将捕获的数据包以 PCAP 文件保存,后续步骤中将使用 Wireshark 软件打开此文件进行数据分析。

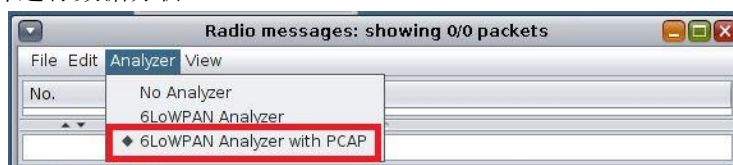


图 22.4 Radio messages 操作

(5) 在 Network 界面中能看到两个节点,其中节点 1 为边界路由,节点 2 为 CoAP 服务器节点。如图 22.5 所示,右击节点 1,选择 Mote tools for Sky1 选项中的 Serial Socket(SERVER) 选项。点击 Serial Socket(SERVER)面板的 Start 按钮,此时可看到下方状态栏显示“Listening on port 6001”。该功能用于创建一个连接 Cooja 仿真环境中 RPL 网络和本地主机。

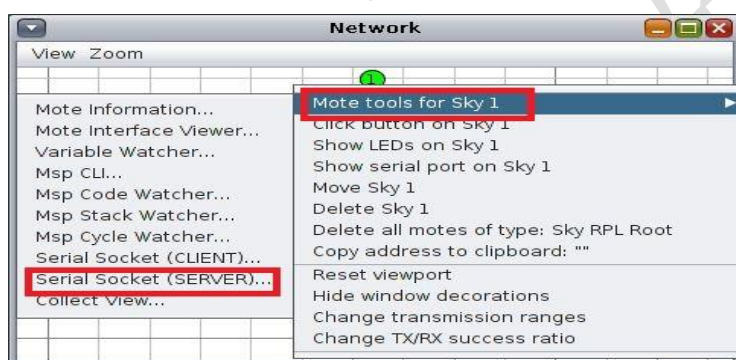


图 22.5 打开 Serial Socket (SERVER) 功能

(6) 如图 22.6 所示,打开新的终端,输入“cd contiki/tools”将目录切换至 contiki/tools 下,使用 make tunslip6 命令编译 tunslip6.c 代码,使用 sudo ./tunslip6 -a 127.0.0.1 aaaa::1/64 命令连接 RPL 网络和外部网络,即节点 2 本地主机。此过程中会提示输入密码,密码为“123456”。命令执行完成后 Serial Socket(SERVER)面板中状态会显示为“Client /127.0.0.1:xxxx connected”,如图 22.7 所示。

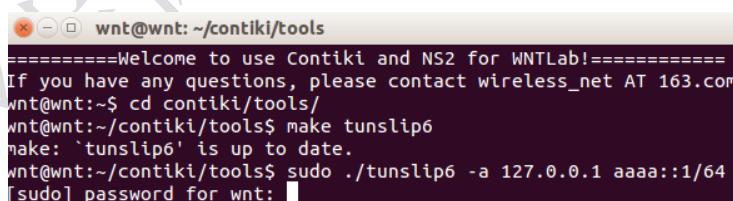


图 22.6 运行 tunslip6

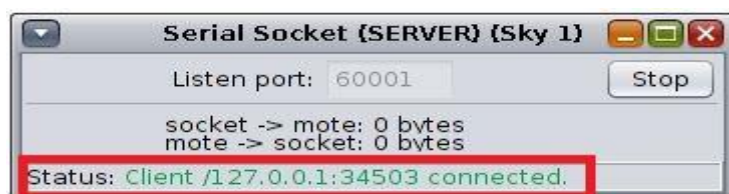


图 22.7 Serial Socket 连接成功

(7) 点击 Simulation control 面板上的 Strat 按钮开始仿真。此时可从 Mote Output 界面看到节点 1 和节点 2 分别分配一个 IPv6 地址为 fe80::212:7401:1:101 和 fe80::212:7402:2:202。由于 fe80::/10 为本地链路单播地址,该地址只用在单链路上,路由器不能将带有链路本地源地址或目标地址的任何包转发到其他链路上去。因此,运行 tunslip6 时,给 RPL 网络分

配了一个 `aaaa::/64` 的网络号, 故节点 1 和节点 2 对外的 IPv6 地址分别为 `aaaa::212:7401:1:101` 和 `aaaa::212:7402:2:202`。

(8) 打开火狐浏览器, 输入地址“`coap://[aaaa::212:7402:2:202]`”即可进入 CoAP 解析器, 这是已经安装好的 Copper 插件。如图 22.8 所示, 其中①区域是请求方式; ②区域显示请求的 URL; ③区域是请求内容选择栏; ④区域显示 CoAP 报文头部信息; ⑤区域显示报文内容; ⑥区域显示 CoAP 消息日志。(需注意, 如果③区域中内容不全, 请读者点击①区域的“Discover”按钮即可出现。)

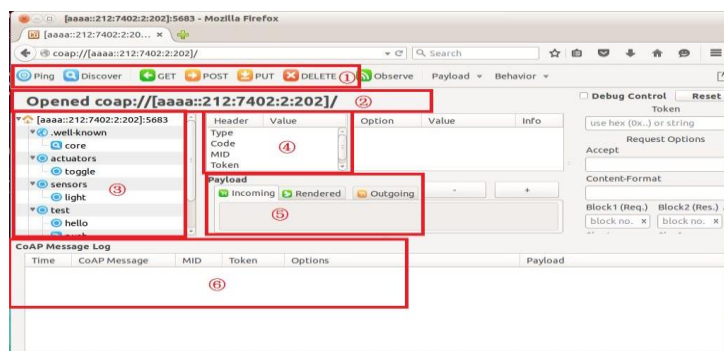


图 22.8 CoAP 解析器显示界面

(9) 点击③号区域 test 下的 hello, 此时在②区域中可看到将要请求的 URL 为“`coap://[aaaa::212:7402:2:202]:5683/test/hello`”。再点击①区域中的 GET 按钮发送请求, 即以 get 方式请求。待接到节点 2 的响应报文后, ④区域显示 CoAP 报文头部信息, ⑤区域显示服务器响应信息。如图 22.9 所示。

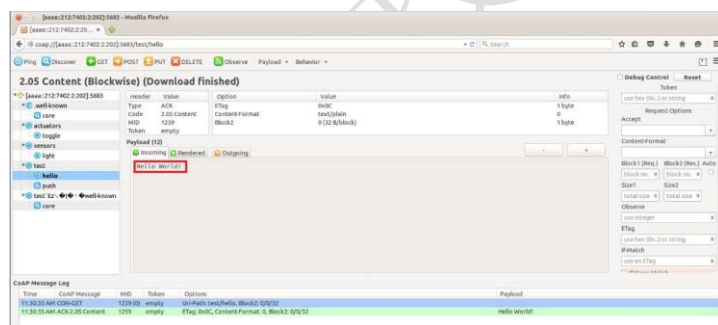


图 22.9 CoAP 请求与响应

(10) 点击 Simulation control 面板的 Pause 按钮停止实验。

(11) 打开桌面 Wireshark 软件 (密码: 123456)。选择 File 菜单栏中的 Open 按钮, 打开 `home/wnt/contiki/tools/cooja/build` 路径下最新“`radiolog-xxxx.pcap`”文件, 如图 22.10 所示。

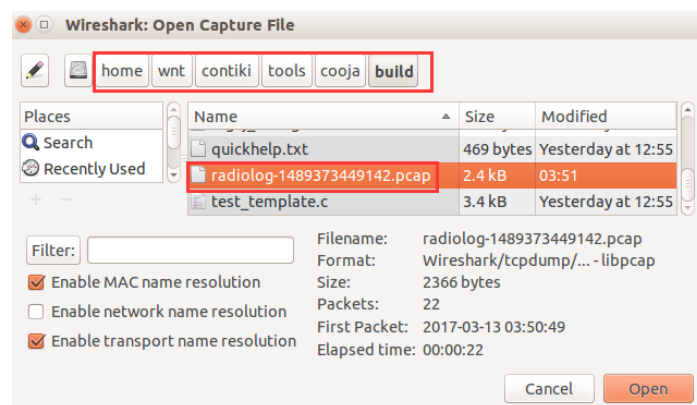


图 22.10 选择 Radio messages 工具捕获的数据包

(12) 图 22.11 为实验中捕获的数据包, 请读者自行分析数据包内容。

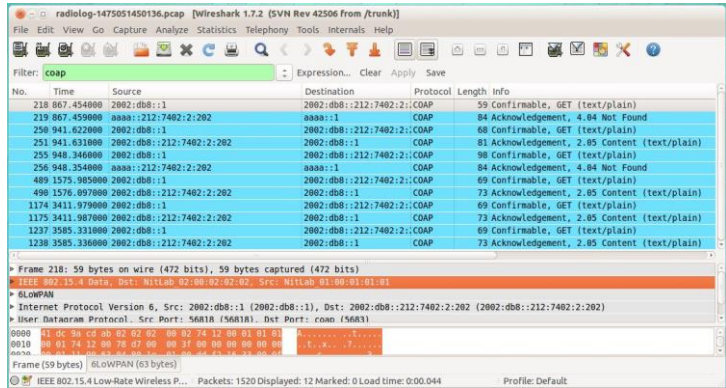


图 22.11 数据包显示

5 实验扩展与分析

在本实验基础上, 读者可以分析解读本实验相关的程序代码, 来了解 CoAP 协议。

6 结果分析与讨论

参考教材第 9 章有关 CoAP 内容分析报文, 来了解 CoAP 交互过程。

7 注意事项

仿真主要涉及到的相关代码无需修改, 具体代码参见本实验电子资源目录下的 border-router.c 和 example-server.c 文件。

实验二十三 RPL 协议仿真实验

1 实验要求与目的

- 1 人独立完成
- 理解 RPL 协议工作过程
- 利用 Cooja 仿真分析 RPL 协议

2 实验背景与原理

2.1 实验背景

RPL 面向物联网, 是为低功耗有损网络设计的基于 IPv6 的距离矢量路由协议, 其根据目标函数, 利用路由度量和约束条件算出最优路径, 构建面向目标的有向无环图(DODAG)。RPL 协议具体实现过程和原理, 请参考教材第 9 章。

2.2 实验原理

本实验分析采用 Cooja 仿真器, 具体使用 sky 节点, 共 30 个, 分别是节点 1-30。其中节点 1 为 DODAG 根节点, 其余均为普通节点。

3 实验环境与资源

(1) 环境资源:

PC 一台、wnt-contiki 仿真环境。

(2) 使用说明:

• 本实验主要用到仿真实验平台中的 Contiki 操作系统。实验一中已提供了实验平台搭建方法, 如果已完成实验一中 wnt-contiki 仿真环境的搭建, 本实验可直接运行;

4 实验步骤与演示

具体步骤如下:

(1) 打开终端, 输入“cd contiki/tools/cooja”命令, 进入 Cooja 仿真器目录。输入“ant run”命令, 运行 Cooja 仿真器。如图 23.1 所示;

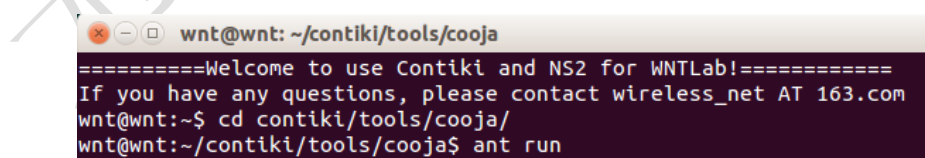


图 23.1 运行 Cooja 仿真器

(2) 点击 Cooja 界面中 File 按钮, 选择 Open simulation 中的 Browse 进入 Open 界面, 点击 home 按钮将目录选至/home/wnt-contiki/WNTLab/experiments/exp23/WNTExamples/rpl-udp, 选择 rpl-udp.csc 点击 open 按钮打开仿真。如图 23.2 所示;

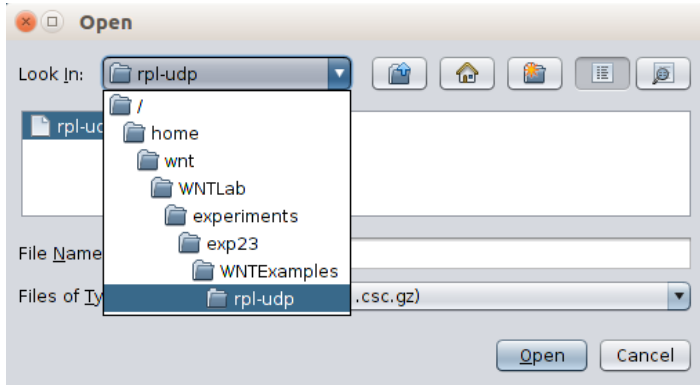


图 23.2 选择实验模块

(3) 实验加载完成会出现四个窗口, 分别是 Simulation control 用于控制仿真开始停止或重新加载实验; Network 显示网络节点的物理布局; Mote output 用于显示节点输出信息; Simulation script editor 仿真器脚本编辑器;

(4) 在 Network 窗口中勾选 View 菜单栏中的 Mote IDs、Addresses: IP or Rime、Radio traffic、Mote Type、Radio environment (UDGM) 选项。如图 23.3 所示。此时可看到根节点 1 显示为绿色, 其余节点为黄色;

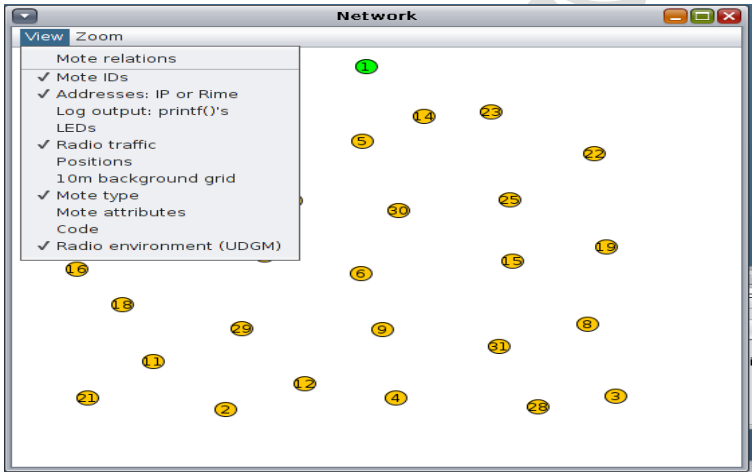


图 23.3 选择 Network 界面显示

(5) 如图 23.4 所示, 选择 Cooja 界面的 Tools 菜单栏中的 Radio messages 工具捕获数据包。选择 Radio messages 界面的 Analyzer 菜单栏中的 6LoWPAN Analyzer with PCAP 选项, 将捕获数据包以 PCAP 文件保存, 后续可使用 Wireshark 软件对其进行数据分析;

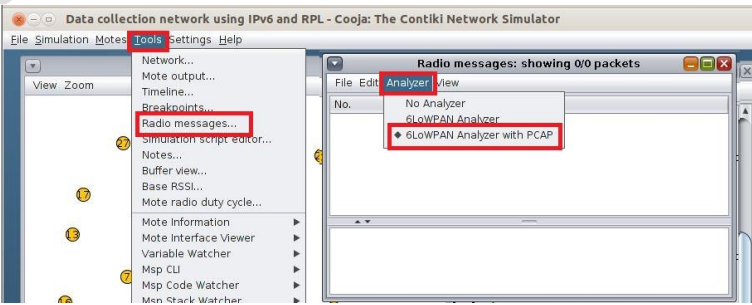


图 23.4 打开 Radio messages 工具

(6) 点击 Simulation control 界面的 Start 按钮开始实验仿真, 此时 Network 界面中的节点开始启动并广播自己的状态。在 Network 和 Mote output 界面中可以看出, 节点启动时各

自获得一个 IPv6 地址。待节点 1 启动后开始组建 DODAG, 即 RPL 网络, 如图 23.5 所示, 在 00:0.404 时节点 1 开始启动, 此时, 节点 1 得到 IPv6 压缩地址: aaaa::ff:fe00:1, 其他节点配置的服务器端地址为该地址, 在 00:01.439 时节点 1 创建了一个新的 DODAG;

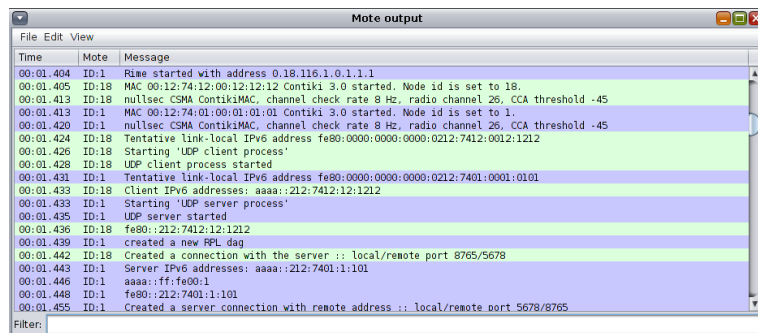


图 23.5 节点 1 启动并创建 DODAG

(7) 节点 1 创建 DODAG 后节点陆续加入 DODAG, 如图 23.6 所示, 节点 14 加入网络并设置默认路由为 fe80::212:7401:1:101 即节点 1, 从图 23.7 可看出节点 14 位于节点 1 附近;

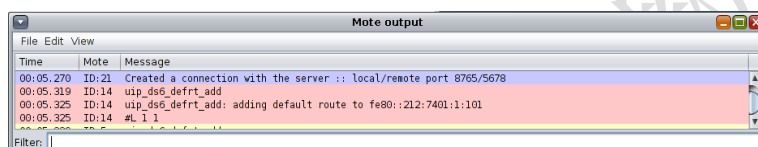


图 23.6 节点 14 加入 DAG

(8) 所有节点都加入 DODAG 后, 节点生成终端数据报文“Hello 1”, 封装成 UDP 报文段, 通过单跳或多跳发送至根。如图 23.7 所示, 节点 11 产生数据经过节点 29、10、30、14 后传至节点 1 (为使图显示清楚, 这里关闭了 Addresses: IP or Rime 显示功能)。图 23.8 展示了节点 11 的数据转发过程;

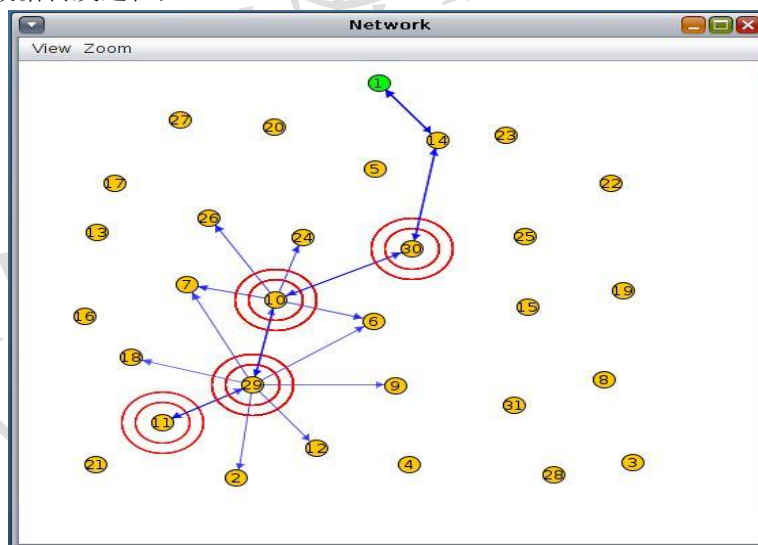


图 23.7 节点 11 的数据转发路径

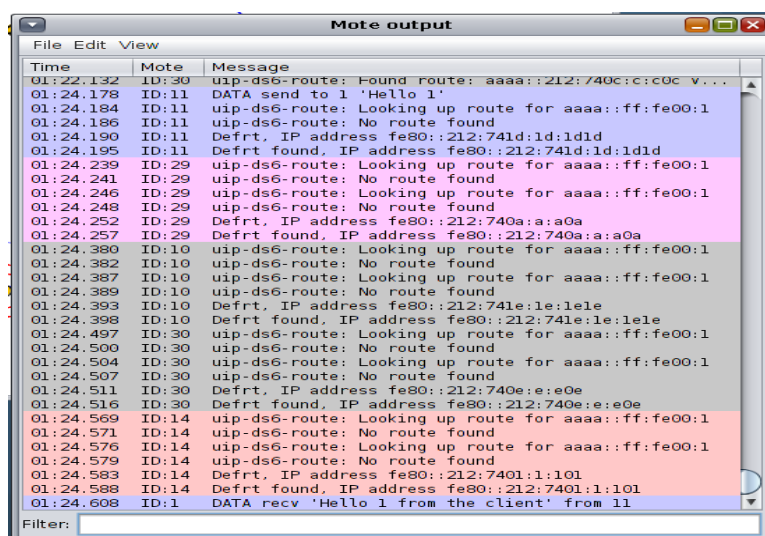


图 23.8 节点 11 的数据转发过程

(9) 点击 Simulation control 面板的 Pause 按钮停止实验。

(10) 运行桌面上的 Wireshark 软件(密码: 123456)。选择 File 菜单栏中的 Open 界面, 打开 home/wnt-contiki/contiki/tools/cooja/build 路径下最新的“radiolog-xxxx.pcap”文件, 如图 23.9 所示;

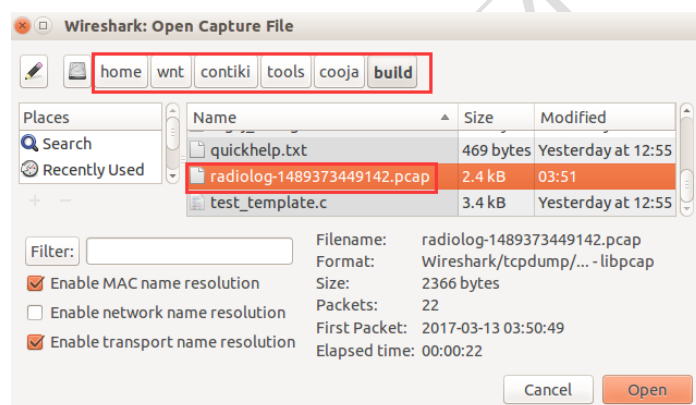


图 23.9 选择 Radio messages 工具捕获的数据包

(11) 图 23.10 为实验中捕获的数据包, 请读者自行分析数据包;

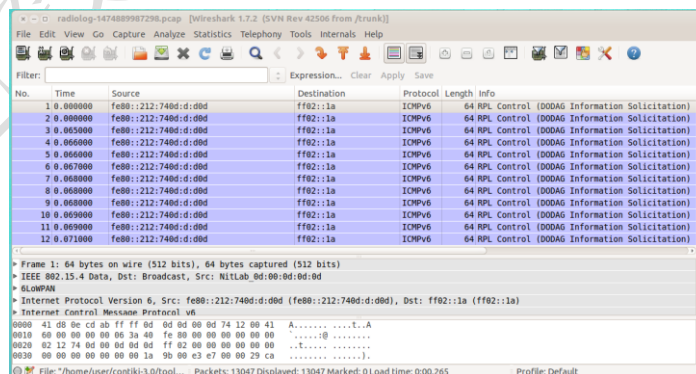


图 23.10 数据包显示

6 实验扩展与分析

在本实验的基础上, 读者可自己动手设计不同场景下的仿真实验, 分析网络性能, 分析解读本实验相关的程序代码。

自建场景步骤如下:

(1) 选择 Cooja 仿真器 File 菜单栏中的 New Simulation, 在 Create new simulation 界面中填写仿真环境的名称, 在 Radio medium 选项中选择 UDGm: Distance Loss 选项, 其余项默认即可, 如图 23.11 所示;



图 23.11 新建仿真

(2) 如图 23.12 所示, 选择要创建节点的类型, 在此我们选 Sky mote;

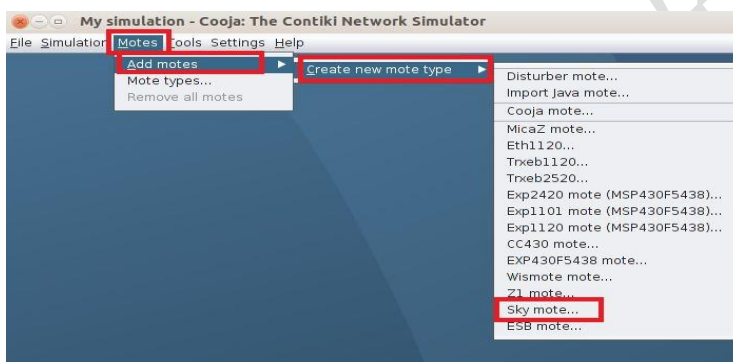


图 23.12 选择节点

(3) 在 Create Mote type 界面中选择创建节点所运行的代码, 并点击 Create 创建。如图 23.13 所示, 我们所用代码为 contiki-3.0/WNTEexamples/rpl-udp 路径下的 udp-client.sky 和 upd-server.sky, 其中 upd-server.sky 为根节点代码, udp-client.sky 为普通节点代码。注意根节点只需一个即可。

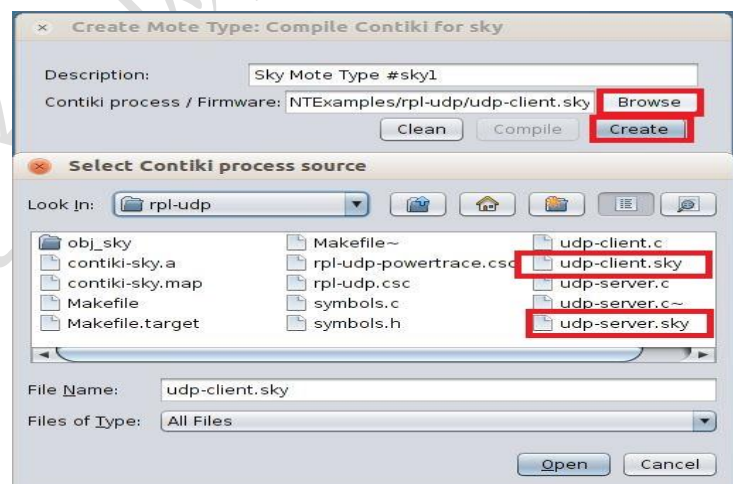


图 23.13 创建节点

5 结果分析与讨论

实验数据包分析课参考教材第 9 章, 实验过程中注意 Meto output 中节点输出内容, 分析节点加入 DODAG 过程。

7 注意事项

仿真主要涉及到的相关代码无需修改, 具体代码参见本实验电子资源目录下的 `upd-client.c` 和 `udp-server.c` 文件。

无线网络技术教程

实验二十四 基于 Contiki 和 RPL 的 无线传感网组网实测

1 实验要求与目的

- 1 人独立完成
- 理解 RPL 协议的工作过程
- 基于 CC2530 节点实现 RPL 组网

2 实验背景与原理

2.1 实验背景

本实验学习使用 RPL 协议(见教材第 9 章)在实际环境中组建物联网。具体使用 CC2530 节点, 通过 RPL 路由协议组成一个 WSN。具体的组网过程可参考实验十七。

2.2 实验原理

整个网络由搭载温度/湿度传感器模块的普通节点和 DODAG 根节点组成。普通节点负责采集、存储和上传温度/湿度信息, 并根据 DODAG 转发邻节点发送的温度/湿度信息。根节点完成温度/湿度数据汇聚并上传至 PC 机。网络拓扑示意图如图 24.1 所示, 节点数量可根据情况自行选择, 力求能体现实验效果即可。

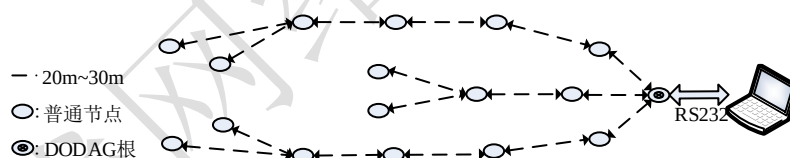


图 24.1 RPL 网络拓扑示意图

实验中传输层采用 UDP 协议进行数据封装, 所以烧录代码时根节点所使用的文件为“udp-server.hex”, 普通节点使用的文件为“udp-client.hex”。

3 实验环境与资源

(1) 环境资源:

PC 一台、CC2530 传感器节点套件若干及软件环境(详见实验十五第 3 节)、温湿度传感器若干、电池及 5V 稳压模块若干、母对母杜邦线若干、wnt-contiki 仿真环境、SSCOM 串口调试助手软件。

(2) 使用说明:

- CC2530 传感器节点套件介绍见实验十五, 其它各类硬件如图 24.2~24.4 所示;
- 本实验主要用到仿真实验平台中的 Contiki 操作系统。实验一中已提供了实验平台搭建方法, 如果已完成实验一中 wnt-contiki 仿真环境的搭建, 本实验可直接运行;

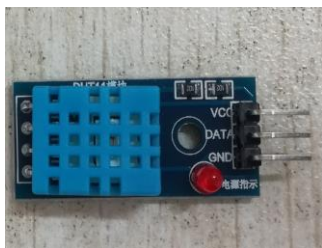


图 24.2 温湿度传感器

图 24.3 电池及 5V 稳压模块

图 24.4 母对母杜邦线

4 实验步骤和演示

下面详细描述实验步骤：

(1) 打开终端，输入“**cd WNTLab/experiments/exp24/WNTExamples/rpl-udp/**”进入实验目录下，使用 **make clean** 命令清除已编译好的文件，再使用 **make** 命令进行编译，如图 24.5 所示；

```
wnt@wnt: ~/WNTLab/experiments/exp24/WNTExamples/rpl-udp
=====Welcome to use Contiki and NS2 for WNTLab!=====
If you have any questions, please contact wireless_net AT 163.com
wnt@wnt:~/WNTLab/experiments/exp24/WNTExamples/rpl-udp$ make clean
using saved target 'cc2530dk'
rm -f *~ *core core *.srec \
*.lst *.map \
*.cprg *.bin *.data contiki*.a *.firmware core-labels.s *.ihex *.ini \
*.ce *.co *.cc2530dk *.lnk *.lk *.sym *.lib *.ihx *.rel *.mem *.rst *.as
*.hex *.omf *.cdb *.banks *.flags *.banked-hex symbols.c symbols.h
rm -rf obj_cc2530dk
wnt@wnt:~/WNTLab/experiments/exp24/WNTExamples/rpl-udp$ make
```

图 24.5 编译操作

(2) 输入“**nautilus**”命令打开文件管理器，进入 WNTLab/experiments/exp24/WNTExamples/rpl-udp 目录下。若在步骤(1)中编译成功，则在该目录下会出现“udp-server.hex”和“udp-client.hex”文件，如图 24.6 所示；

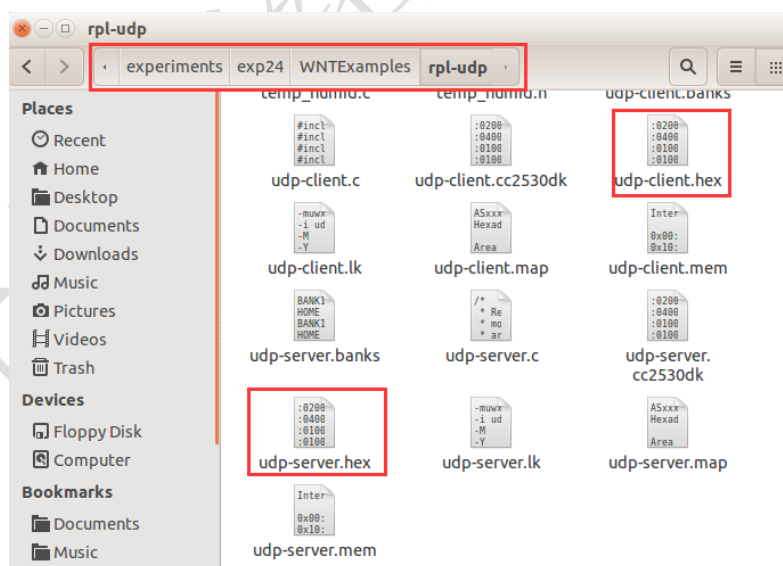


图 24.6 查看生成文件

(3) 将生成的两个 hex 文件拷贝至 Windows 环境下；

(4) 使用 CC Debugger 调试器连接 CC2530 节点和 PC，若 CC2530 节点电源开关未开启，则仿真器指示灯为红色。开启传感器节点电源开关，点击仿真器复位键，若 CC Debugger 仿真器驱动安装成功，则仿真器指示灯为绿色，即连接正常。待传感器节点和 PC 机连接成功后，打开 SmartRF flash Programmer 软件，则图 24.7 中的①区域会显示节点类型等信息，

表示传感器节点和 PC 机连接成功;

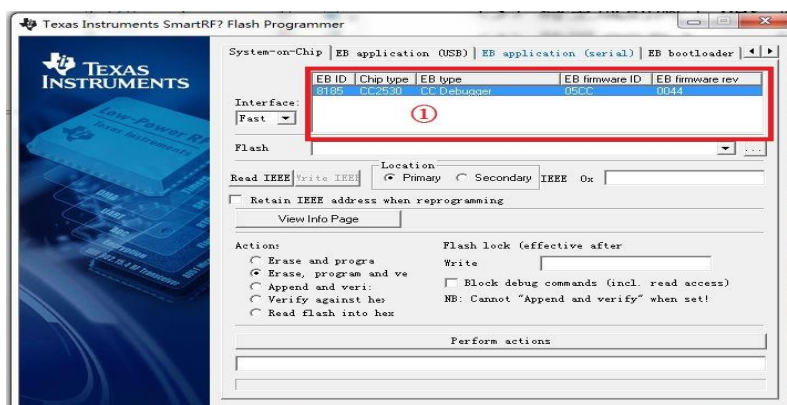


图 24.7 SmartRF flash Programmer 界面

(5) 在 Flash 选项中点击省略号按钮, 在目录中选择从虚拟机中粘贴过来的“udp-server.hex”文件, Actions 选项中选择“Erase, program and verify”, 点击 Perform actions 按钮进行代码烧录, 如图 24.8 所示;

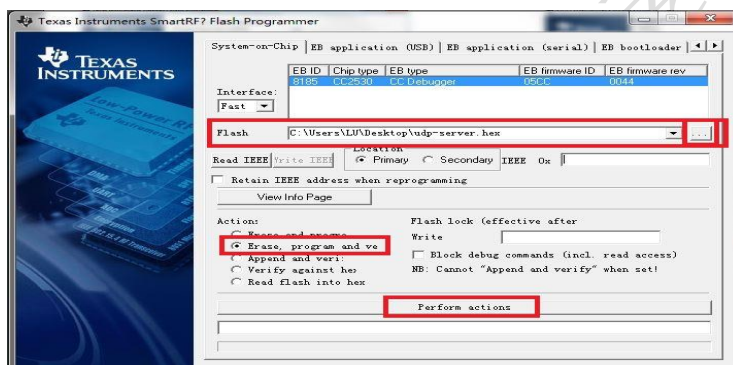


图 24.8 代码烧录

(6) 烧录完成后显示如图 24.9 所示。

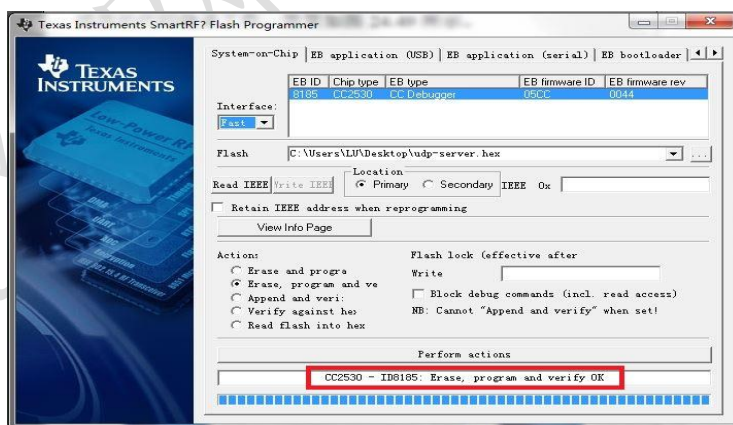


图 24.9 烧录完成

(7) 按照步骤(5)、(6)根据实际情况烧录若干个 udp-client 节点。

(8) 分别采用 Mini USB 线分别将根节点和一个普通节点连接到 PC 机 USB 端口, 打开串口调试软件。可同时运行两个串口调试软件, 来进行数据对比。

(9) 如图 24.10 所示, 在 PC 机设备管理器中查询节点与 PC 机连接的端口号。

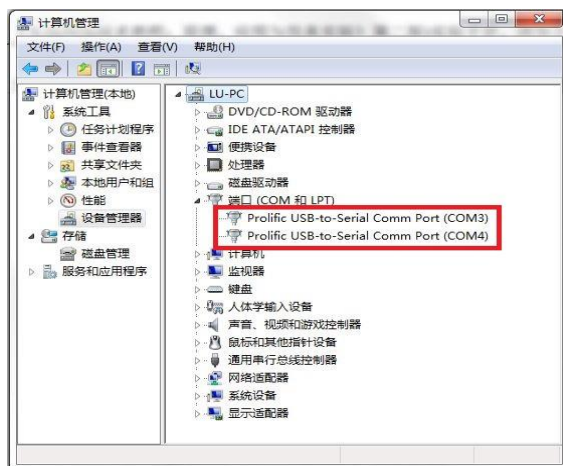


图 24.10 端口查询

(10) 将串口调试软件的端口设置为步骤(9)中查出的端口号, 波特率设置为 115200, 并取消十六进制显示, 如图 24.11 所示。

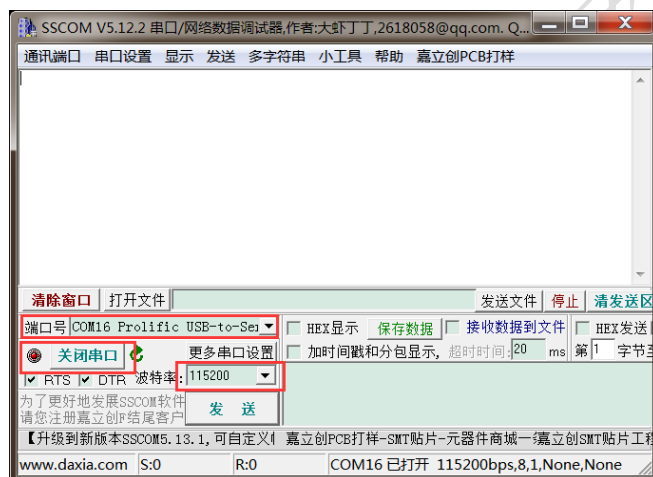


图 24.11 串口调试工具设置

(11) 点击串口调试软件中的打开串口按钮, 接收串口数据。

(12) 分别打开根节点电源开关和普通节点电源开关, 此时可在串口调试软件中看到节点相关信息。整个 RPL 网络组建完成, 普通节点开始向根节点发送终端数据报文, 如图 24.12 和 24.13 所示。终端数据报文格式为“T:x°C, H:y%”, 其中 T 代表温度, x 是由温度传感器采集到的摄氏温度值, C 代表摄氏度, H 代表湿度, y 是由湿度传感器采集到的湿度值。

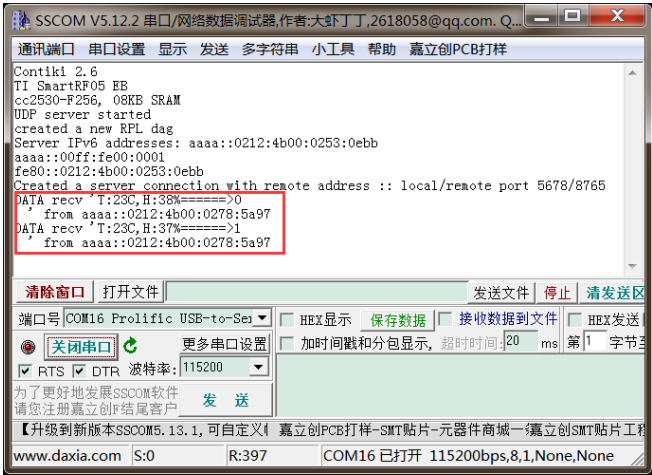


图 24.12 根节点信息及接收到信息

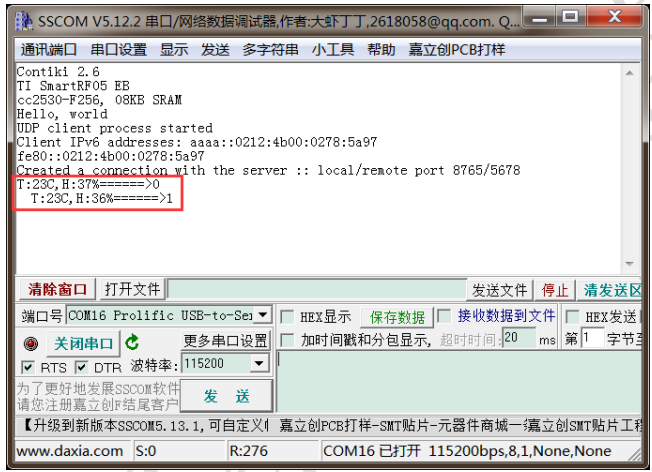


图 24.13 普通节点信息及发送信息

5 实验扩展与分析

考虑到实验复杂性, 本实验只实现了温湿度信息采集, 读者可自行添加电流传感器、磁场传感器、气体传感器等, 组建完整的环境多因素监测系统。

6 结果分析与讨论

实验过程中可能由于电池能量和信号干扰等因素, 数据到达会出现乱序或丢包等情况。且由于传感器节点的测量精确度只能控制在一定范围内, 实验数据可能有误差。

7 注意事项

仿真主要涉及到的相关代码无需修改, 具体代码参见本实验电子资源目录下的 `upd-client.c` 和 `udp-server.c` 文件。

实验二十五 IEEE 802.11p 和 VANET 仿真

1 实验要求与目的

- 1 人独立完成
- 理解 IEEE 802.11p 协议和无线车载网工作机制
- 利用 NS2 仿真分析 IEEE 802.11p 协议和无线车载网组网

2 实验背景与原理

2.1 实验背景

无线车载网络、VANET 和 IEEE 802.11p 的背景知识内容请参见教材第 10 章。本实验主要分析节点移动情形下, IEEE 802.11p 和 VANET 节点的数据传输情况。

2.2 实验原理

IEEE 802.11p 协议仿真实验主要分析节点移动情形下, MAC 层采用 IEEE 802.11p 时节点间的数据传输情况, 实验拓扑如图 25.1 所示。

VANET 仿真实验主要分析在街道场景(无障碍物)下, 随着车辆的运动, 车与车之间的通信性能。本实验构建了纵横形式的街道, 同时采用 9 个节点构建车载通信实验, 如图 25.2 所示。

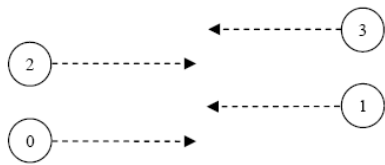


图 25.1 IEEE 802.11p 协议仿真拓扑

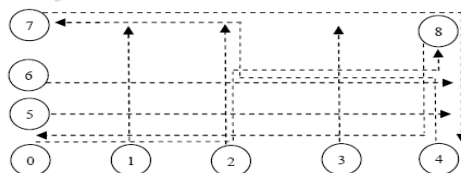


图 25.2 VANET 仿真拓扑

3 实验环境与资源

(1) 环境资源:

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;
- 本实验主要用到仿真实验平台中的 NS2 软件, 官方 NS2 版本中已包含 IEEE 802.11p 协议仿真模块与 VANET 仿真模块。实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;
- 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp25/。

4 实验步骤与演示

4.1 IEEE 802.11p 仿真分析

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 25.3 所示。



图 25.3 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本, 本实验的子文件夹为“exp25”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

`cd WNTLab/experiments/exp25/`

再输入以下命令运行本实验, 如图 25.4 所示。

`ns 802_11p.tcl`

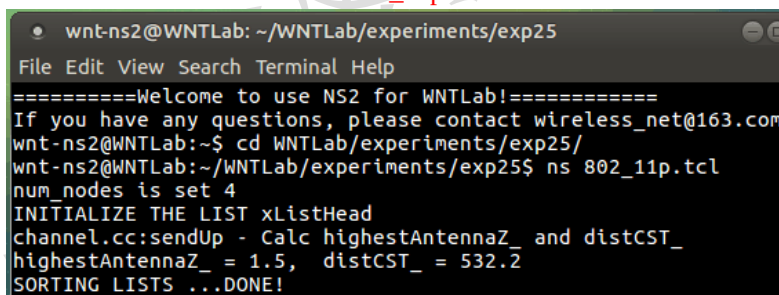


图 25.4 运行仿真实验

(4) 利用 NAM 动画观察实验结果, 如图 25.5 所示。注意, 观看 NAM 动画时, 运行步长调整为 2ms 为宜。

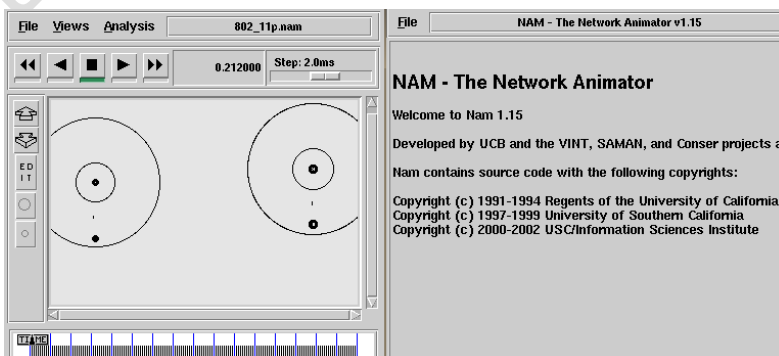


图 25.5 实验结果演示

程序运行说明: 仿真开始时, 节点 0、1、2、3 开始移动, 速度为 40 米/秒, 同时节点 0

和节点 1 分别与节点 2 和节点 3 建立 UDP/CBR 数据流。注意, 在数据传输前, 节点 0 和节点 1 需通过 RREQ/RREP 与节点 2 和节点 3 建立路由。仿真到达 20s 时, 两条流传输同时结束, 仿真实验结束。

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析, 并绘制图形对比。

更为详细的实验演示可运行仿真实验脚本进行观察, 对于没有实验环境的读者, 本实验手册的电子资源提供了仿真实验动画。

4.2 VANET 组网仿真分析

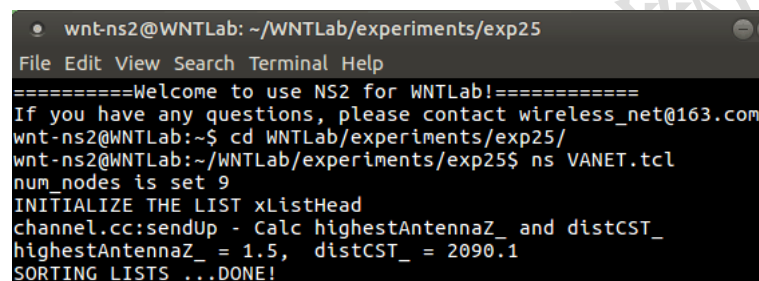
下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

`cd WNTLab/experiments/exp25/`

再输入以下命令运行本实验, 如图 25.6 所示。

`ns VANET.tcl`



```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp25
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp25/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp25$ ns VANET.tcl
num_nodes is set 9
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 2090.1
SORTING LISTS ...DONE!
```

图 25.6 运行仿真实验

(2) 利用 NAM 动画观察实验结果, 如图 25.7 所示。注意, 在观看 NAM 动画时, 运行的步长调整为 2ms 为宜。

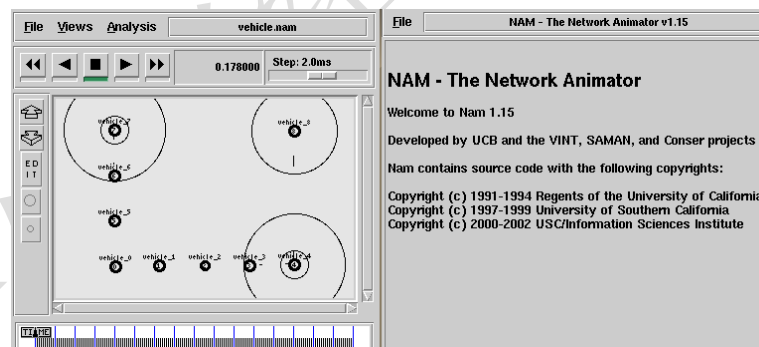


图 25.7 实验结果演示

程序运行说明: 仿真开始时, 节点移动轨迹如教材中实验部分的拓扑所示, 移动速度各不相同, 具体看实验代码。节点 0、3、1、6 分别与节点 4、8、2、7 建立 UDP/CBR 数据流, 并在 0s 后开始数据传输。注意, 节点 0、3、1、6 发送数据前需通过 RREQ/RREP 与节点 4、8、2、7 建立路由。仿真到达 100s 时, 4 条流传输同时结束, 仿真实验随之结束。

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析, 并绘制图形对比。

更为详细的实验演示可运行仿真实验脚本进行观察, 对于没有实验环境的读者, 本实验手册的电子资源提供了仿真实验动画。

5 实验扩展与分析

进行 IEEE 802.11p 协议实验时, 需考虑节点的传输范围, 在本实验中设置其为 250 米, 读者可调整不同的传输范围并重复实验。读者还可自己动手设计不同网络场景下的仿真, 分析 VANET 组网的性能, 分析解读实验相关的程序代码。

6 结果分析与讨论

仿真分析过程中请分析以下两个问题:

(1) IEEE 802.11p 协议与其他 MAC 层的协议有何差别?

(2) IEEE 802.11p 协议存在哪些不足?

仿真分析的过程中还需注意正确配置 MAC 层协议和参数, 以及网络层路由协议。

实验二十六 无线体域网健康监测系统

1 实验要求与目的

- 1人独立完成
- 了解无线体域网的基本原理和应用
- 了解无线体域网设备间的通信协议
- 深入学习各种传感设备通信的过程

2 实验背景与原理

2.1 实验背景

无线体域网(WBAN)越来越受到关注,目前仍处在早期阶段,在低功耗、互操作性、系统设备、安全性、验证、数据一致性等方面尚面临许多挑战。WBAN 相关协议和技术知识内容见教材第11章。

2.2 实验原理

本实验原理如图 26.1 所示。传感器与单片机相连,单片机可将获取到的感知数据通过蓝牙/IEEE802.15.6/802.15.4 等协议传输给手机 APP,从而进行通信,传递数据。



图 26.1 实验原理图

3 实验环境与资源

(1) 环境资源:

Arduino MEGA 2560 单片机模块一个、数字血压检测模块一个、气泵及电磁阀套件一套、电子血压计袖带一个、蓝牙 4.0 模块一个、电池及 5V 稳压模块一套、USB A 型转 B 型线、公对母杜邦线若干、PC 一台、Arduino IDE、心率血压测量程序、无线体域网监测系统手机 APP、安卓手机一部（需自备）。

(2) 使用说明:

- 各类硬件如图 26.2 至 26.9 所示;
- Arduino IDE 是 Arduino 系列单片机的开发环境,考虑到稳定性,推荐读者使用 1.0.x 系列版本,官网下载连接为 <https://www.arduino.cc/en/Main/OldSoftwareReleases#previous>,详细的下载方法及说明请见实验资源包 WNTLab/experiments/exp26。
- 心率血压测量程序开放源代码,供读者学习与二次开发,详见实验资源包 WNTLab/experiments/exp26;
- 关于无线体域网监测系统手机 APP,本实验提供 apk 安装包与所有源码,供读者学习与二次开发,详见实验资源包 WNTLab/experiments/exp26;

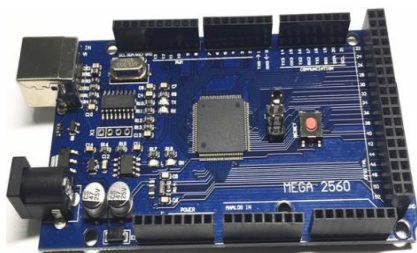


图 26.2 Arduino MEGA 2560 单片机模块

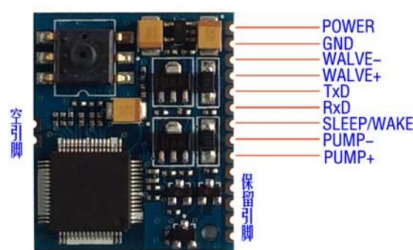


图 26.3 数字血压检测模块

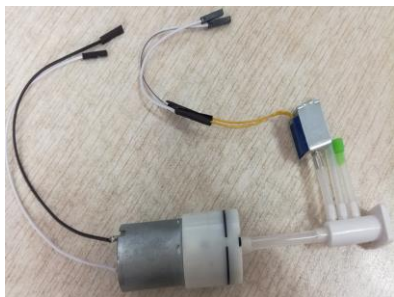


图 26.4 气泵及电磁阀套件



图 26.5 电子血压计袖带

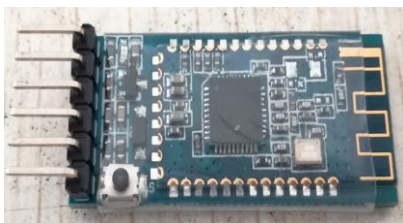


图 26.6 蓝牙 4.0 模块

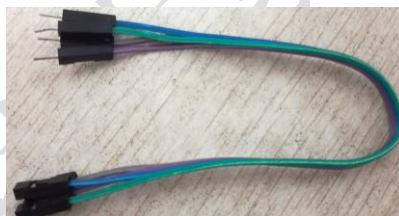


图 26.7 公对母杜邦线



图 26.8 电池及 5V 稳压模块



图 26.9 USB A 型转 B 型线

4 实验步骤和演示

本实验内容为根据图 26.1 测量人体心率、血压健康参数；实验要求 1 人独立完成。

(1) 根据 3.1 节准备好所有的实验资源；

(2) 在手机中下载无线体域网监测系统手机 APP 的安装包(可通过 QQ 从 PC 传给手机)，点击安装并等待安装完成，如图 26.10 和 26.11 所示。(需注意，该 APP 要求手机的系统为 Android 5.0 及其以上版本。)



图 26.10 开始安装



图 26.11 安装完成

(3) 使用 USB A 型转 B 型线连接 Arduino MEGA 2560 单片机模块与 PC，在 PC 端打开心率血压测量程序文件(code-exp26.ino);

(4) 在菜单栏选择“Tools-Board-Arduino Mega 2560 or Mega ADK”，并在“Serial Port”中选中相应端口号，如图 26.12 所示，点击“Upload”按钮后等待烧录成功；

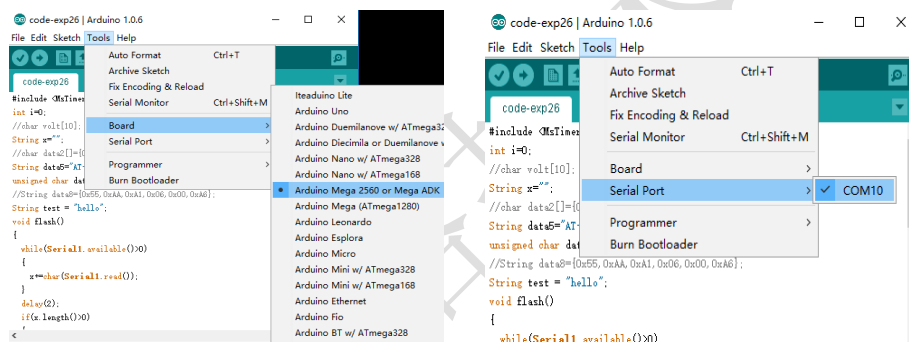


图 26.12 选择开发板型号与串口号

(5) 将数字血压检测模块、气泵及电磁阀套件、电子血压计袖带、Arduino MEGA 2560 单片机模块、蓝牙 4.0 模块按照如图 26.13 所示方式连接,并仔细检查,连接完成后如图 26.14 所示:

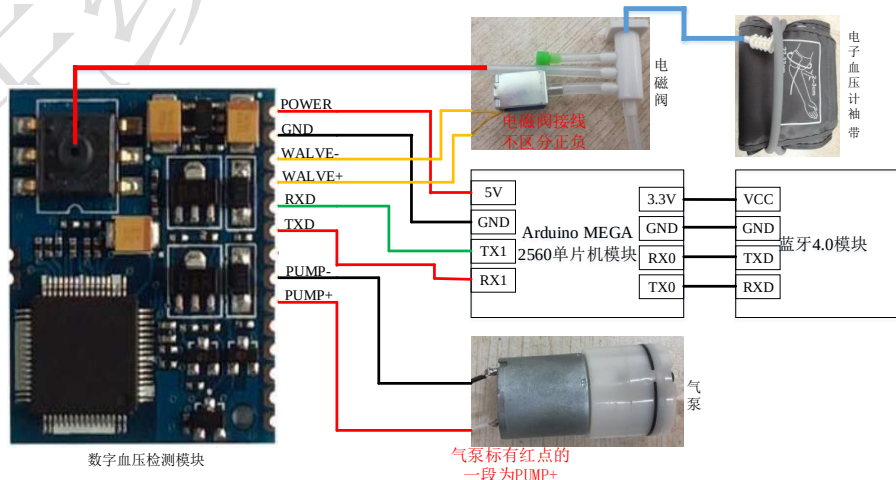


图 26.13 各模块连接图

(6) 使用 USB A 型转 B 型线连接 5V 稳压模块与 Arduino MEGA 2560 单片机模块，并

将电池连接在 5V 稳压模块的 DC 口, 如图 26.15 所示:

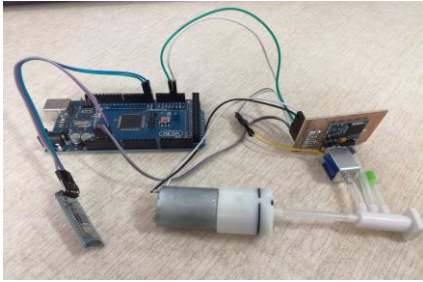


图 26.14 连接各模块

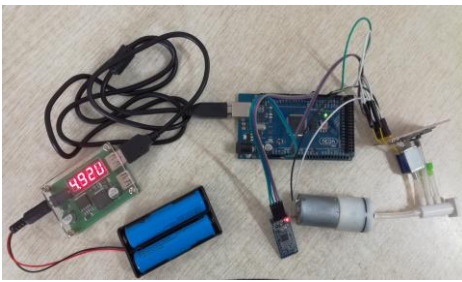


图 26.15 连接电源

(7) 所有硬件连接好并供电后, 将电子血压计袖带绑至手臂, 并且打开无线体域网监测系统手机 APP, 如图 26.16 所示:

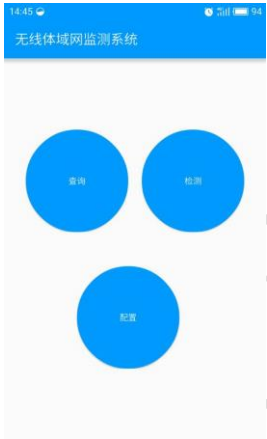


图 26.16 手机 APP 初始界面

(8) 点击手机 APP 主界面配置按钮进入配置系统, 设置好姓名、手机号以及亲友手机号等, 然后退出配置界面, 如图 26.17 所示;

(9) 点击主界面检测按钮, 弹出蓝牙权限请求, 点击允许并搜索蓝牙设备, 如图 26.18 所示;

(10) 如图 26.19 所示, 点击搜索到的对应蓝牙 4.0 模块的蓝牙信号进行配对;

(11) 进入控制检测界面, 点击开始检测, 系统显示正在检测, 图 26.20 和图 26.21;

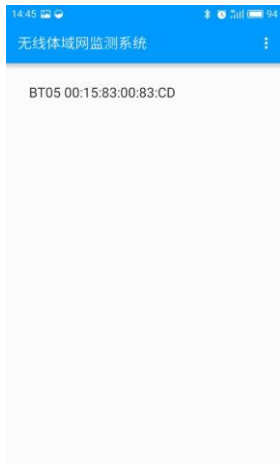


图 26.17 设置基本信息 图 26.18 系统请求蓝牙权限 图 26.19 搜索到蓝牙设备



图 26.20 控制检测界面



图 26.21 系统正在测量

(12) 此时硬件设备正在运行, 等待约 10~20 秒, 系统检测完毕;

(13) 当硬件检测完成后, 检测到的数据会通过蓝牙自动发送到手机 APP, APP 的控制检测界面将显示检测到的结果, 如图 26.22;



图 26.22 显示检测结果

(14) 点击发送按钮, 系统会将检测到的用户的生理参数通过短信发送给预先设定好的亲友手机号, 如图 26.23 所示;

(15) 点击 APP 主界面的查询按钮, 查看当前用户所有的检测记录, 如图 26.24 所示。



图 26.23 亲友手机号收到短信



图 26.24 用户的检测记录

5 实验扩展与分析

WBAN 还有许多可扩展的内容, 涉及各种健康信息, 对可靠性、准确性、安全性和传输时延等有严格要求。该实验由于诸多不确定性因素还存在着误差, 数据并不是很准确, 读者可考虑进一步扩展。

6 结果分析与讨论

本实验介绍了 WBAN 的概念、应用、通信协议, 并利用传感器与 APP 测量了人体各种生理特征数据。

7 注意事项

WBAN 存在诸多不确定因素, 如身体大幅动作造成的数据传输延迟或丢失、周围电磁信号对通信的影响等。

实验二十七 无线室内定位仿真实验

1 实验要求与目的

- 1人独立完成
- 理解无线室内定位的基本原理
- 利用 NS2 仿真分析无线室内定位

2 实验背景与原理

2.1 实验背景

无线室内定位技术的发展,催生了许多极具意义的应用,为生活、工作带来了便利,更多背景内容和原理知识见教材 11.2 节。本实验主要仿真无线室内定位的定位过程,具体的定位算法不在本实验的讨论范围,感兴趣的读者可自行阅读原始文献[1]。

已有 NS2 并未集成无线定位,为便于无线室内定位实验,Adnan Abu-Mahfouz 等人开发了基于 NS2 的无线定位仿真程序[1],能满足本实验的需求。

2.2 实验原理

实验中,利用已知节点对未知节点进行定位,采用的定位方法依赖于节点的 RSSI。实验拓扑结构如图 27.1 所示,共 8 个节点,其中节点 0 至节点 4 为未知节点,节点 5 至节点 7 为已知节点。

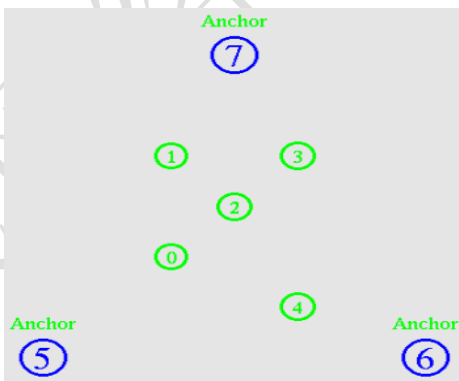


图 27.1 无线室内定位仿真拓扑结构

3 实验环境与资源

(1) 环境资源:

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;
- 本实验主要用到仿真实验平台中的 NS2 软件,官方 NS2 版本中未包含本实验所需模块。实验一中已提供了实验平台搭建模块和脚本,如果已完成实验一中 wnt-ns2 仿真环境的搭建,本实验可直接运行;
- 本实验提供源代码,供读者学习与二次开发,详见实验资源包

WNTLab/experiments/exp27/。

4 实验步骤与演示

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 27.2 所示。



图 27.2 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本。本实验的子文件夹为“exp27”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹:

```
cd WNTLab/experiments/exp27/
```

再输入以下命令运行本实验, 如图 27.3 所示。

```
ns Indoor_Localization.tcl 1.tcl
```

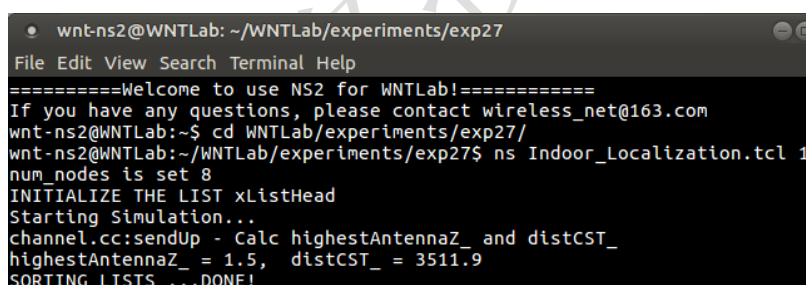


图 27.3 运行仿真实验

(4) 利用 NAM 动画观察实验结果, 如图 27.4 所示。注意, 在观看 NAM 动画时, 运行步长调整为 2ms 为宜。

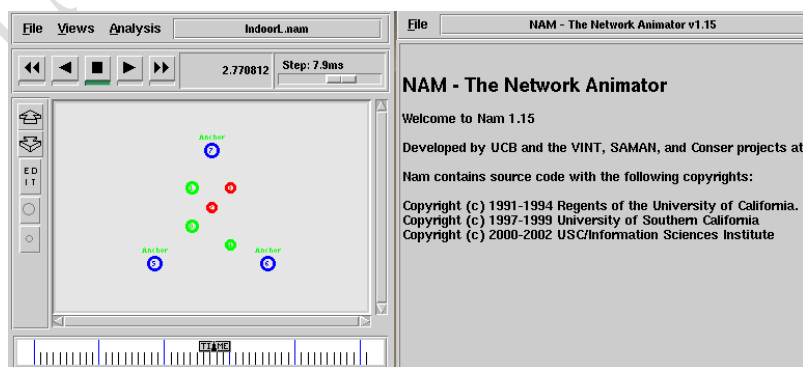


图 27.4 实验结果演示

程序运行说明: 仿真开始时, 节点被划分成两类, 即已知节点(蓝色)和未知节点(黄色)。仿真运行到 2.4s 左右时, 节点 2 被成功定位, 显示为红色; 3.2s 左右, 节点 0 被成功定位;

3.7s 左右, 节点 4 被成功定位; 4.6s 左右, 节点 1 被成功定位; 10.5s 左右, 节点 3 被成功定位。所有节点定位成功后, 仿真实验结束。

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析, 并绘制图形对比。

更为详细的实验演示可运行仿真实验脚本进行观察, 对于没有实验环境的读者, 本实验手册的电子资源提供了仿真实验动画。

5 实验扩展与分析

在本实验的基础上, 读者可自己动手设计不同网络场景下的仿真, 分析无线室内定位算法的性能, 分析解读实验相关的程序代码。

6 结果分析与讨论

仿真分析的过程中还需要注意已知节点和未知节点的位置, 避免无法定位的情况。

7 参考文献

[1] A. M. Abu-Mahfouz, G. P. Hancke. NS-2 Extension to Simulate Localization System in Wireless Sensor Networks. In Proceedings of the IEEE Africon, Sep. 2011.

实验二十八 WiFi 无线室内定位实践

1 实验要求与目的

- 1人独立完成
- 了解室内定位算法的基本原理
- 了解 WIFI 室内定位操作过程

2 实验背景与原理

2.1 实验背景

本实验对应的相关理论知识内容见教材 11.2 节“无线室内定位”。

常见的室内定位算法主要分为两类：基于测距的算法和距离无关的算法。前者一般通过节点间距离或角度来计算出未知节点位置，常见有：接收信号强度指示（RSSI）、到达角度（AOA）、到达时间（TOA）等。距离无关的算法有：质心法、APIT、凸规划等。这些算法都利用节点间的邻近关系实现定位的。

2.2 实验原理

本实验采用 RSSI，由于信号在传播过程中信号强度会降低，根据接收机收到的信号强度，可测算发射机与接收机的距离。无线信道的数学模型如下：

$$PL(d) = PL(d_0) - 10n \lg\left(\frac{d}{d_0}\right) - x_\sigma \quad (1)$$

式中 d 是发射机和接收机之间的距离； d_0 是参考距离； n 是信道衰减指数； x_σ 是均值为零、方差 σ 的高斯随机噪声变量； $PL(d)$ 是距离发射机 d 处的信号强度； $PL(d_0)$ 是距离发射机 d_0 处的信号强度，可通过经验或硬件规范定义中得到。故可通过上式求得距离 d 。

本实验采用多边定位算法，下面简单介绍一下三边定位算法。

三边定位法如图 28.1 所示，已知参考节点 A、B、C 的坐标为 (x_a, y_a) 、 (x_b, y_b) 、 (x_c, y_c) 。设待定位节点 O 的坐标是 (x, y) ，则有以下方程：

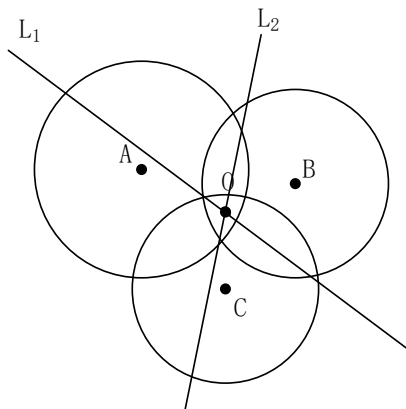


图 28.1 三边定位

$$\begin{cases} (X - x_a)^2 + (Y - y_a)^2 = d_a^2 \\ (X - x_b)^2 + (Y - y_b)^2 = d_b^2 \\ (X - x_c)^2 + (Y - y_c)^2 = d_c^2 \end{cases} \quad (2)$$

将前两个等式分别减去第三个等式, 即可得到图中的直线 L_1 、 L_2 :

$$\begin{cases} 2(x_a - x_c)X + 2(y_a - y_c)Y = x_a^2 - x_c^2 + y_a^2 - y_c^2 - d_a^2 + d_c^2 \\ 2(x_b - x_c)X + 2(y_b - y_c)Y = x_b^2 - x_c^2 + y_b^2 - y_c^2 - d_b^2 + d_c^2 \end{cases} \quad (3)$$

解方程组 (3) 可得到定位节点的估计坐标为:

$$\begin{cases} X = \frac{(y_a - y_c)c_2 - (y_b - y_c)c_1}{2(x_b - x_c)(y_a - y_c) - 2(x_a - x_c)(y_b - y_c)} \\ Y = \frac{(x_a - x_c)c_2 - (x_b - x_c)c_1}{2(x_a - x_c)(y_b - y_c) - 2(x_b - x_c)(y_a - y_c)} \end{cases} \quad (4)$$

其中,

$$\begin{cases} c_1 = x_a^2 - x_c^2 + y_a^2 - y_c^2 - d_a^2 + d_c^2 \\ c_2 = x_b^2 - x_c^2 + y_b^2 - y_c^2 - d_b^2 + d_c^2 \end{cases} \quad (5)$$

即将定位节点设置为两直线的交点。

3 实验环境与资源

(1) 环境资源:

PC 一台、无线网卡一个、智能手机三部 (需自备)、室内定位程序、Java 环境。

(2) 使用说明:

●无线网卡如图 28.2 所示, 若读者实验所用 PC 无内置无线网卡, 则使用该无线网卡。

第一次使用时需要安装驱动, 详见实验资源包 WNTLab/experiments/exp28。

●本室内定位程序开放全部源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp28/。

●Java 环境主要是 JDK, 它是 Java 语言的软件开发工具包, 包含了 Java 的运行环境 JRE 和 Java 工具。作者使用版本为 9.0.1, 安装包大小约 370MB, 下载方法请见实验资源包 WNTLab/experiments/exp4, 官网下载地址为: <http://www.oracle.com/technetwork/cn/java/javase/downloads/index.html>。



图 28.2 无线网卡

4 实验步骤与演示

4.1 实验环境布置

- (1) 测量实验所需的房间大小(长和宽), 下面选择了一间 7m×7.8m 的房间。
- (2) 给房间定义坐标, 下面选择房间的左上角作为 (0,0) 点。
- (3) 开启三部智能手机的热点, 任意设置热点的名称。这里分别设置为 AP1、AP2、AP3。
- (4) 部署参考节点: 将手机随机放到房间内, 测量它们所在的位置并标注在坐标上。如这里将手机分别放置在 (2.6,1.1)、(4.95,6.45)、(6.45,2.48) 的位置上, 如图 28.3 所示。
- (5) 部署定位节点: 本实验中, 定位节点为 PC, 将 PC 放置于任意位置即可。

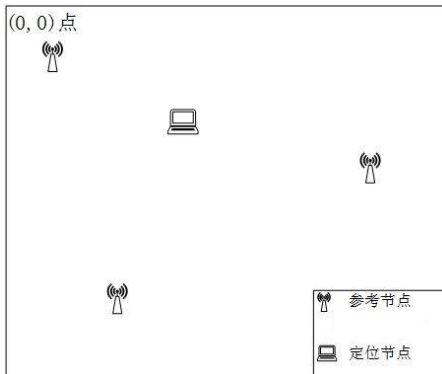


图 28.3 实验环境示意图

4.2 实验参数配置

从本手册电子资源中找到本实验执行文件压缩包: WirelessLocalization.rar, 将其解压缩至任一目录下, 如图 28.4 所示。

解压得到的文件中, “wifi.exe”为可执行程序, 用于扫描无线网卡以获得参考节点的 RSSI, 并将 RSSI 值以“SSID: RSSI”形式保存在“APRSSI.txt”中; “ReferAP.txt”中保存的是参考节点的位置, 并以 x:y 的形式保存, 该文件需要我们在定位前根据 4.1 节中测量的参考节点位置来手工输入并保存; “Localization.jar”是可执行的定位程序。

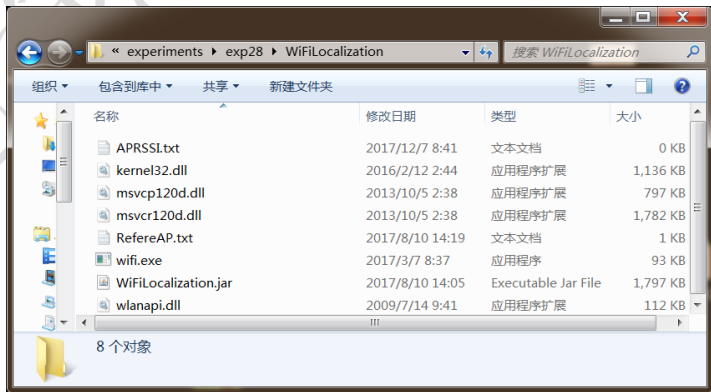


图 28.4 拷贝执行文件

关于“ReferAP.txt”配置文件, 详细介绍如下。

ReferAP.txt 中有 n 行数据, 每行代表一个参考节点的位置。如 4.1 节所述, 三个参考节点的位置分别为 (2.6,1.1)、(4.95,6.45)、(6.45,2.48), 因此在“ReferAP.txt”文件中记录如图 28.5 所示内容。



图 28.5 RefereAP.txt 配置文件

4.3 定位

(1) 在 PC 中, 打开命令提示符窗口, 利用 **cd** 命令进入本实验可执行文件所在目录, 如图 28.6 所示。

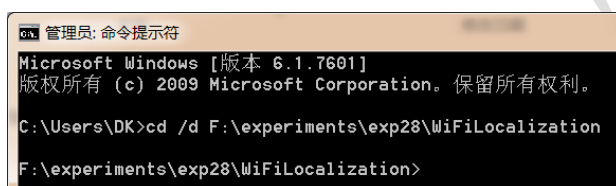


图 28.6 进入可执行文件所在目录

(2) 继续输入命令 **wifi.exe** 并执行, 进行无线网卡的扫描, 扫描结束后会列出所扫描到的 AP 信息, 如图 28.7 所示。

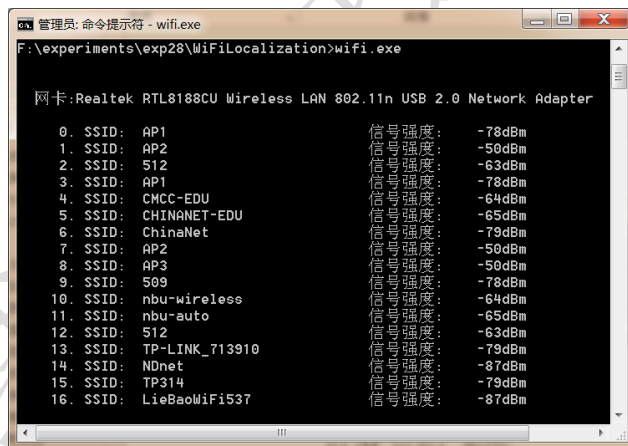


图 28.7 wifi.exe 扫描结果

(3) 观察上述扫描结果, 如果结果中有 3 个参考节点的信息, 则继续输入命令 **java -jar Localization.jar**, 并按照提示输入 3 个参考节点的热点名称, 输入完成后按下回车键, 得出定位结果, 如图 28.8 所示。(**需注意**, 若第(2)步扫描结果中的参考节点信息少于 3 个, 则重新执行第(2)步, 因为本实验程序要求定位节点收到大于 3 个参考节点的信息才能定位成功。)



图 28.8 输入 3 个参考节点的热点名称

5 结果分析与讨论

作者在实验中进行 10 次测试, 定位结果如表 28.1 所示。其中, 定位误差采用的是实际位置和定位位置之间的欧氏距离。可得到如下结论: 该定位算法的平均误差为 2.81m, 其中 50% 的定位节点的误差在 1~2 米之内; 第 5 次定位虽然由定位算法得到了定位位置, 但由于结果坐标超出了实验范围, 故定义为定位失败。

表 28.1 WiFi 室内定位结果

编号	实际位置 (m)	定位位置(m)	定位误差(m)	备注
1	(4.65,1.25)	(5.32,6.46)	5.25	
2	(2.30,2.65)	(5.16,6.12)	4.50	
3	(2.95,5.40)	(4.72,5.36)	1.77	
4	(5.46,5.31)	(5.90,7.11)	1.85	
5	(6.05,1.62)	(9.44,1.31)	3.40	定位失败
6	(4.60,4.50)	(5.17,6.10)	1.70	
7	(2.18,3.92)	(6.08,7.42)	5.24	
8	(4.95,7.20)	(5.95,7.16)	1.00	
9	(3.10,4.25)	(4.21,3.57)	1.30	
10	(0.60,1.50)	(1.38,3.46)	2.11	

6 扩展与分析

在本实验中仅讨论了 3 个节点的定位情况, 读者可在本实验基础上采用更多节点进行重复实验。

7 注意事项

实际实验中, 会发现定位误差无法消除, 而且影响因素较大。RSSI 定位算法目前被认为较为基础, 不如 TOA、AOA 等更为精确。具体内容参见教材第 11 章及相关技术文献。

实验二十九 无线网络攻击仿真实验

1 实验要求与目的

- 1人独立完成
- 掌握 Blackhole 的攻击原理
- 掌握 Greyhole 的攻击原理
- 利用 NS2 仿真实现 Blackhole 和 Greyhole 攻击

2 实验背景与原理

2.1 实验背景

无线网络尤其是 MANET, 在带来方便的同时产生了诸多安全问题, 开放的无线介质, 多跳、自组织、无中心的组网方式, 使得其更加脆弱。其中, 黑洞(Blackhole)和灰洞(Greyhole)攻击是典型代表, 对网络数据传输危害极大。本实验以黑洞和灰洞攻击为例, 通过对 NS2 中的 AODV 协议进行修改, 以实现简单的黑洞和灰洞实验。更为复杂的路由欺骗(为黑洞和灰洞做铺垫)在本实验中不深入论述。

黑洞特征主要有: ①即使没有到目标节点的路由信息, 黑洞节点也会声称自己具备该路由, 目的是干扰正常的数据传输; ②当收到其它节点传来的数据分组时, 黑洞节点会全部丢弃该分组, “黑洞”由此得名。有两种方式, 单节点黑洞攻击和多个节点配合的黑洞攻击。灰洞是比黑洞更具威胁, 其行为表现介于正常节点和黑洞节点之间, 按攻击方式大体分为 3 类: 节点选择型, 分组类型选择型和时间选择型。之所以更危险因为它比黑洞更难以检测。

对上述黑洞和灰洞的特征, 本实验并不完全实现, 仅提供部分满足实验需求的功能。其中, 黑洞主要实现了第 2 点, 灰洞则实现特定类型数据包的概率丢弃。黑洞和灰洞攻击的更多内容, 读者可阅读教材第 12 章。

2.2 实验原理

黑洞和灰洞攻击在实施前, 需进行路由欺骗, 以使源节点的数据包通过黑洞或灰洞节点, 进而能进行有效攻击。本实验采用了一个多跳无线网络拓扑, 如图 29.1 所示。为简便起见, 实验中并不考虑路由欺骗过程(源节点的数据包肯定会经过黑洞或灰洞节点), 其中黑洞或灰洞部署在节点 1 上。



图 29.1 无线网络攻击仿真拓扑

3 实验环境与资源

(1) 环境资源:

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;

●本实验主要用到仿真实验平台中的 NS2 软件, 官方 NS2 版本中未包含本实验所需模块。实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;

●本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp29/。

4 实验步骤与演示

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 29.2 所示。



图 29.2 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本, 本实验的子文件夹为“exp29”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

cd WNTLab/experiments/exp29/

再输入以下命令可分别运行黑洞攻击实验和灰洞攻击实验, 如图 29.3 所示。

ns BlackHole.tcl

ns GreyHole.tcl

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp29
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp29/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp29$ ns BlackHole.tcl
num_nodes is set 4
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 550.0
SORTING LISTS ...DONE!

wnt-ns2@WNTLab: ~/WNTLab/experiments/exp29
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp29/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp29$ ns GreyHole.tcl
num_nodes is set 4
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 550.0
SORTING LISTS ...DONE!
```

图 29.3 运行仿真实验

(4) 利用 NAM 动画观察实验结果, 如图 29.4 和图 29.5 所示, 其中前者是黑洞攻击, 后者是灰洞攻击。注意, 在观看 NAM 动画时, 运行步长调整为 2ms 为宜。

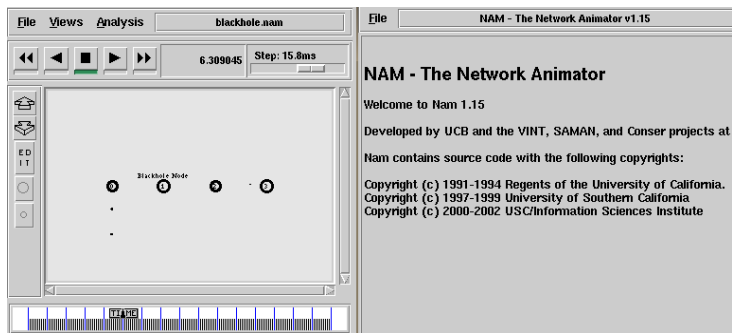


图 29.4 黑洞攻击实验结果演示

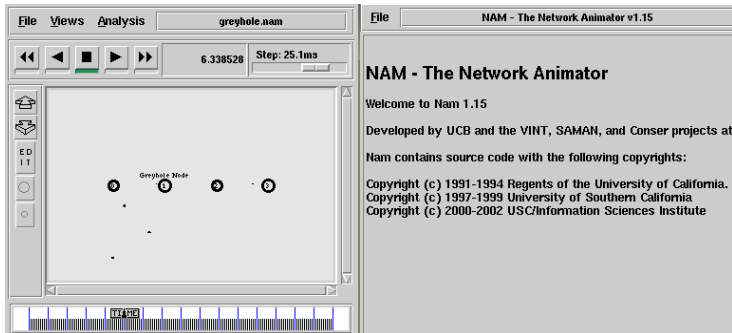


图 29.5 灰洞攻击实验结果演示

程序运行说明：仿真开始，所有节点未建立路由，无数据传输。在 1.0s 时，节点 0 和节点 3 建立 UDP/CBR 数据传输。需注意，数据传输前，节点 0 要通过 RREQ/RREP 与节点 3 建立路由，本实验的黑洞和灰洞不对路由进行攻击。仿真到 6s 时，节点 1 开始黑洞或灰洞攻击，使得数据包无法到达节点 3，或者仅部分数据包到达节点 3。在 20s 时，数据传输结束，仿真实验也随之结束。

如果读者想进一步分析仿真的深层次结果，可参考资料自行编码分析脚本对 trace 文件 (.tr) 进行分析，并绘制图形对比。

更为详细的实验演示可运行仿真实验脚本进行观察，对于没有实验环境的读者，本实验手册的电子资源提供了仿真实验动画。

5 实验扩展与分析

在本实验的基础上，读者可自己动手设计攻击模块，细化黑洞和灰洞攻击的性能，并在不同网络场景下进行仿真分析，同时考虑采用 Watchdog 机制进行攻击的检测。

6 结果分析与讨论

有关本实验的结果分析和代码说明详见教材 12.7 节，同时在仿真分析过程中还需注意攻击节点的位置，对于复杂的网络拓扑，可采用路由欺骗达到攻击的目的。

实验三十 Watchdog 无线网络监测仿真

1 实验要求与目的

- 1人独立完成
- 了解 Watchdog 的无线网络攻击检测原理
- 掌握 Watchdog 的攻击检测的仿真实现

2 实验背景与原理

2.1 实验背景

实验二十九已展示了黑洞和灰洞攻击, 为检测上述攻击, 本实验采用 Watchdog 进行实验分析。背景知识内容见教材第12章。

Watchdog 通过混淆模式监测邻节点转发数据包的情况, 并设定相应阈值, 以检测出攻击(自私)节点, 是有效防御黑洞和灰洞攻击的前提。本质上, Watchdog 监测每个包被转发的情况(与缓存记录比较), 如果节点未有效转发数据包, 则被认为恶意。换言之, 邻节点的每个包都被 Watchdog 监听, 并与自身缓存中的包进行比较, 以验证是否正确转发。

2.2 实验原理

实验采用 Watchdog(<http://safewireless.sourceforge.net/tools.html>) 监测邻节点的数据传输, 以分析检测出攻击节点。在具体分析中, 设置了源节点 0、目标节点 3、黑洞攻击节点 1 和 Watchdog 节点 4。具体实验场景和拓扑如图 30.1 所示。

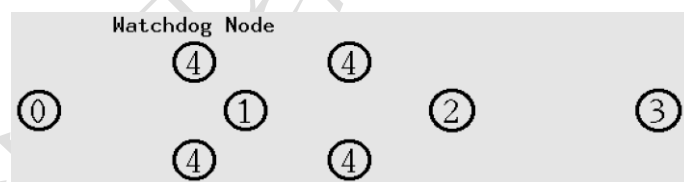


图 30.1 Watchdog 仿真拓扑

3 实验环境与资源

(1) 环境资源:

树莓派套件、wnt-ns2 仿真环境、实验代码。

(2) 使用说明:

- 树莓派套件见实验一第 3.1 节介绍;
- 本实验主要用到仿真实验平台中的 NS2 软件。实验一中已提供了实验平台搭建模块和脚本, 如果已完成实验一中 wnt-ns2 仿真环境的搭建, 本实验可直接运行;
- 本实验提供源代码, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp30/。

4 实验步骤与演示

下面详细描述实验步骤, 读者可根据该步骤运行和分析本实验。

(1) 若已完成实验一第 4.1 节, 将 TF 卡插入卡槽, 使用 HDMI 转 VGA 线和 VGA 线连接树莓派与显示器, 再将树莓派连接电源后等待系统启动, 如图 30.2 所示。



图 30.2 树莓派连接电源和显示器

(2) 在“home/wnt-ns2/WNTLab/experiments”目录下已包含本手册的所有实验脚本, 本实验的子文件夹为“exp30”。

(3) 打开 Terminal(快捷键“Ctrl+Alt+T”), 输入以下命令进入本实验脚本所在文件夹;

`cd WNTLab/experiments/exp30/`

再输入以下命令运行本实验, 如图 30.3 所示。

`ns Watchdog.tcl`

```
wnt-ns2@WNTLab: ~/WNTLab/experiments/exp30
File Edit View Search Terminal Help
=====Welcome to use NS2 for WNTLab!=====
If you have any questions, please contact wireless_net@163.com
wnt-ns2@WNTLab:~$ cd WNTLab/experiments/exp30/
wnt-ns2@WNTLab:~/WNTLab/experiments/exp30$ ns Watchdog.tcl
num_nodes is set 4
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 550.0
SORTING LISTS ...DONE!
```

图 30.3 运行仿真实验

(4) 运行一段时间后, Watchdog 节点能够检测出攻击节点, 如图 30.4 所示。

```
(Stored 1 neighbours)
-----
IP 0:0 (0) -> IP 3:0 (1), Type:2, Data: 10007, Time: 81.056000
IP 0:0 (0) -> IP 3:0 (1), Type:2, Data: 9883, Time: 81.058793
Node IP: 4 MAC: 4 says:
Node: IP 1 MAC 1 (81.058792 secs)
Total Rv: 6709 Fw: 672 St:2007 Beta:0.94 a:9.435245 b:0.40255
0 gamma:0.850000 (Rv: 6709 Fw: 0 (672) Ls:0 (5351) St: 686)
UDP Rv: 6709 Fw: 672 St:2007
Node of 1 route/s: [0, 3],

(Stored 1 neighbours)
-----
IP 0:0 (0) -> IP 3:0 (1), Type:2, Data: 10008, Time: 81.064000
IP 0:0 (0) -> IP 3:0 (1), Type:2, Data: 9884, Time: 81.068900
Node IP: 4 MAC: 4 says:
Node: IP 1 MAC 1 (81.068901 secs)
Total Rv: 6710 Fw: 672 St:2007 Beta:0.94 a:9.435245 b:0.40255
0 gamma:0.850000 (Rv: 6710 Fw: 0 (672) Ls:0 (5351) St: 687)
UDP Rv: 6710 Fw: 672 St:2007
Node of 1 route/s: [0, 3],
```

图 30.4 实验结果

如果读者想进一步分析仿真的深层次结果, 可参考资料自行编码分析脚本对 trace 文件(.tr)进行分析, 并绘制图形对比。

5 实验扩展与分析

在本实验的基础上, 读者可根据不同场景设置 Watchdog 的检测阈值, 甚至修改检测算法, 以更准确检测攻击节点。

6 结果分析与讨论

仿真分析过程中还需注意攻击节点的位置, 对于复杂的网络拓扑, Watchdog 节点的部署位置要合适。

无线网络技术教程

实验三十一 基于蓝牙、WiFi 和语音控制的智能灯控系统

1 实验要求与目的

- 1 人独立完成
- 掌握蓝牙和 WiFi 的原理与应用
- 掌握单片机控制蓝牙、WiFi 模块的过程与原理
- 了解 PocketSphinx 离线语音的基本原理和应用

2 实验背景与原理

2.1 实验背景

WiFi、蓝牙的背景知识请参考教材第3章、第9章；

CMU PocketSphinx 是语音识别的开源项目，是完全离线的语音识别，只要按照要求部署项目，识别率可以达到要求。

2.2 实验原理

本实验内容为：Android 手机 APP 通过 WiFi 模块或蓝牙模块与 Arduino 单片机通信，从而控制 LED 灯的状态变换，实验原理如图 31.1 所示。



图 31.1 实验原理

3 实验环境与资源

(1) 环境资源：

Arduino Mega 2560 单片机模块一个、WiFi 模块一个、蓝牙 4.0 模块一个、LED 灯模块一个、USB A 型转 B 型线、公对母杜邦线若干、PC 一台、Arduino IDE、Arduino 灯控程序、智能灯控手机 APP、Android 系统手机一部（需自备）。

(2) 使用说明

- 各类硬件如图 31.2 至图 31.7 所示，使用方法见实验步骤；
- Arduino IDE 是 Arduino 系列单片机的开发环境，考虑到稳定性，推荐读者使用 1.0.x 系列版本，官网下载连接为 <https://www.arduino.cc/en/Main/OldSoftwareReleases#previous>，详细的下载方法及说明请见实验资源包 WNTLab/experiments/exp31。
- 关于 Arduino 灯控程序、智能灯控手机 APP，本实验提供所有源码及相关 apk 安装包，供读者学习与二次开发，详见实验资源包 WNTLab/experiments/exp31。

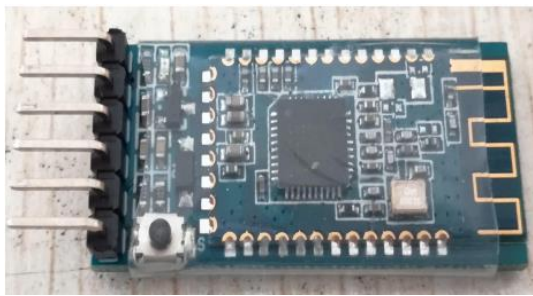


图 31.2 蓝牙 4.0 模块



图 31.3 LED 模块



图 31.4 Arduino MEGA 2560 单片机模块

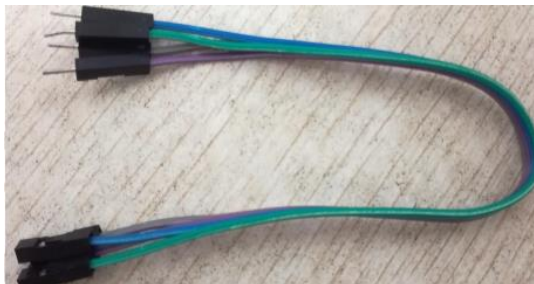


图 31.5 杜邦线



图 31.6 USB A 型转 B 型线

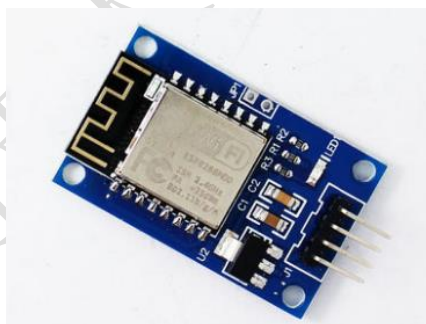


图 31.7 WiFi 模块

4 实验步骤与演示

4.1 蓝牙模式

4.1.1 软件安装及程序烧录

- (1) 根据 3.1 节介绍, 准备好所有的实验资源。
- (2) 在手机中下载智能灯控 APP 安装包(可通过 QQ 从 PC 传给手机), 点击安装并等待安装完成, 如图 31.8 和图 31.9 所示。(需注意, 该 APP 要求手机为 Android 4.4 及以上版本。)

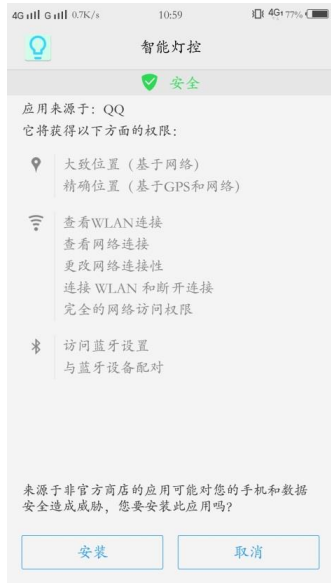


图 31.8 开始安装



图 31.9 安装完成

(3) 使用 USB A 型转 B 型线连接 Arduino Mega 2560 单片机模块与 PC。参考实验十六 4.1 节 Arduino 代码烧录过程, 打开 Arduino IDE, 编译本实验代码 (图 31.10), 并烧录到 Arduino Mega 2560 单片机中。 (需注意, 烧录程序时, Arduino 上不要连接任何模块, 否则会导致烧录失败。)

(4) 烧录程序完成后, 拔掉 Arduino Mega 2560 单片机模块与 PC 连线。



图 31.10 实验程序

4.1.2 硬件连接

(1) 根据图 31.11 所示方式, 使用公对母杜邦线连接蓝牙 4.0 模块、Arduino Mega 2560 单片机模块、LED 模块, 并仔细检查。

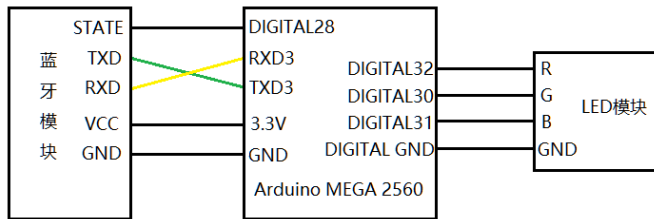


图 31.11 各模块连接图

(2) 模块连接完成后, 使用 USB A 型转 B 型线连接 Arduino Mega 2560 单片机模块与 PC, 如图 31.12 所示。

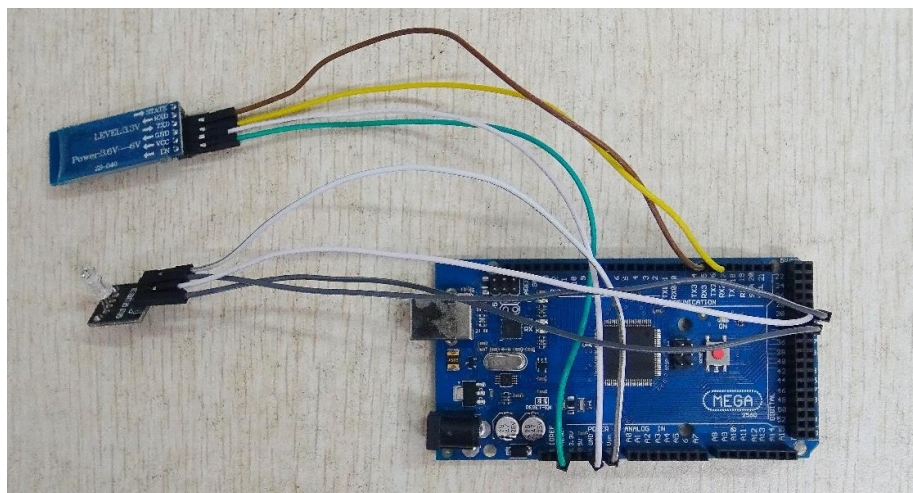


图 31.12 硬件连接

4.1.3 系统测试

- (1) 打开智能灯控手机 APP, 进入初始界面, 如图 31.13 所示。
- (2) 点击蓝牙模式, 弹出蓝牙权限请求, 点击允许并搜索蓝牙设备, 如图 31.14、31.15 所示。
- (3) 点击搜索到的对应蓝牙 4.0 模块的蓝牙信号进行配对后, 进入灯控界面, 如图 31.16 所示。

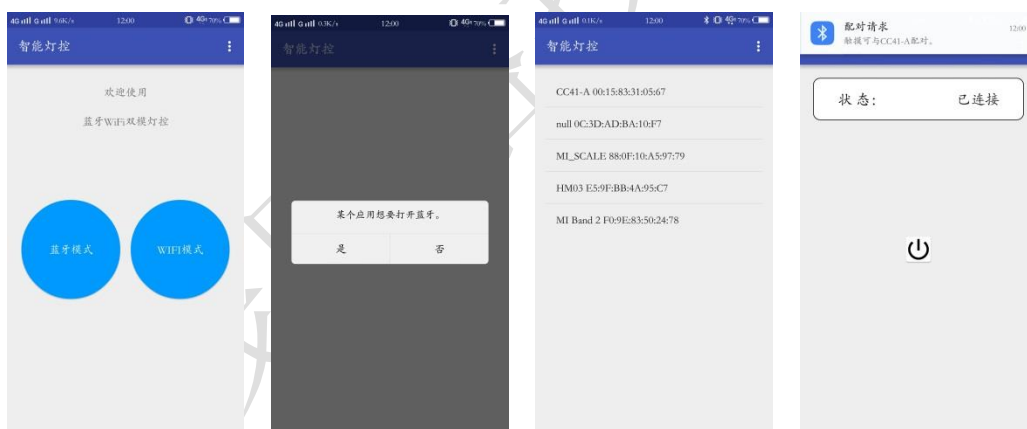


图 31.13 初始界面 图 31.14 蓝牙模式 图 31.15 搜索蓝牙 图 31.16 灯控界面

- (4) 点击手机 APP 灯控界面中间按钮, 可实现 LED 灯的开关与灯色的变换, 如图 31.17 所示。



图 31.17 灯色变化

4.2 WiFi 模式

4.2.1 实验准备

(1) 根据实验三第 4.1 节步骤, 组建无线局域网, 可以选择不接入 Internet, 实验在本地局域网下进行即可。(读者也可选择已有的无线局域网进行实验)

(2) 在手机中连接步骤(1)中建立的 WiFi, 打开智能灯控手机 APP, 进入 WiFi 模式, 查看页面上方显示的 IP 地址及端口号。

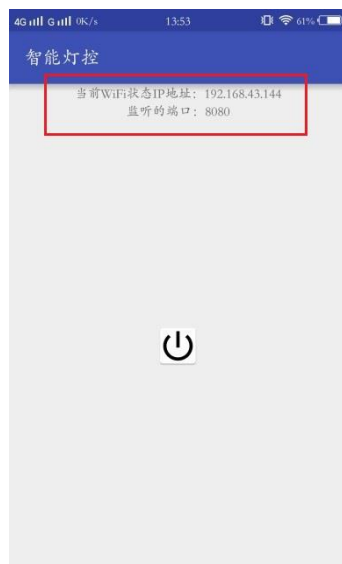


图 31.18 查看手机 IP 地址

(3) 修改实验程序。修改如图 31.19 红色框处定义的“SSID”、“pwd”、“Addr”三个字段的具体内容, 三个字段内容分别为步骤(1)和(2)中所连接 WiFi 的名称、密码以及手机获得的局域网 IP 地址。

(4) 使用 USB A 型转 B 型线连接 Arduino Mega 2560 单片机模块与 PC。参考实验十六 4.1 节 Arduino 代码烧录过程, 打开 Arduino IDE, 编译本实验代码, 并烧录到 Arduino Mega 2560 单片机中。(需注意, 烧录程序时, Arduino 上不要连接任何模块, 否则会导致烧录失败。)

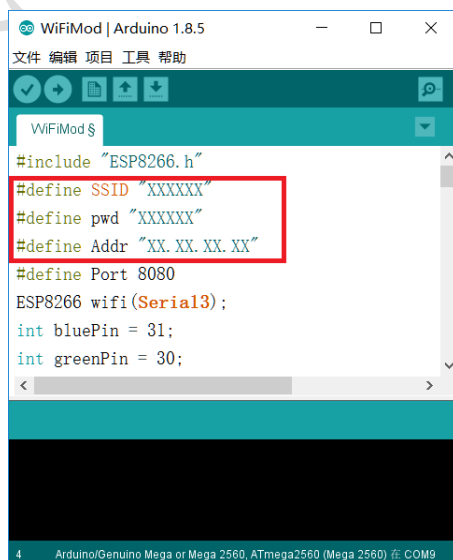


图 31.19 WiFiMod.ino 文件

4.2.2 硬件连接

(1) 根据图 31.20 所示方式, 使用公对母杜邦线连接蓝牙 4.0 模块、Arduino Mega 2560 单片机模块、LED 模块, 并仔细检查。

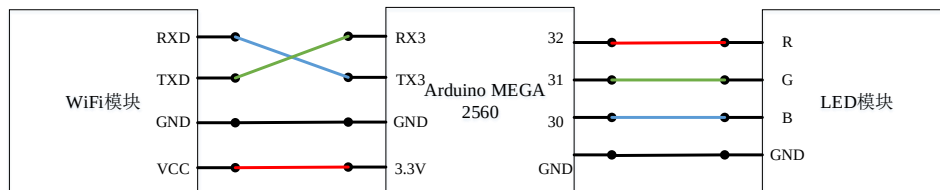


图 31.20 各模块连接图

(2) 模块连接完成后, 使用 USB A 型转 B 型线连接 Arduino Mega 2560 单片机模块与 PC, 如图 31.21 所示。

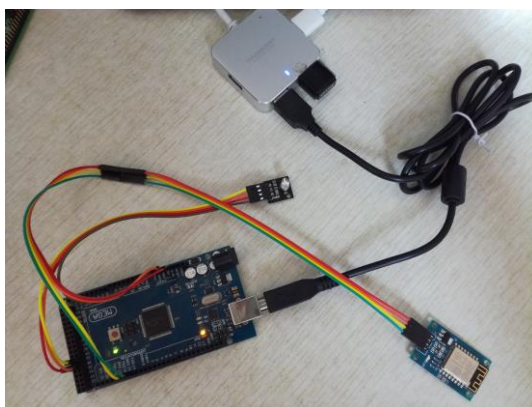


图 31.21 硬件连接

4.2.3 系统测试

(1) 点击 WiFi 模式, 进入灯控界面, 如图 31.22 和 31.23 所示。



图 31.22 初始界面

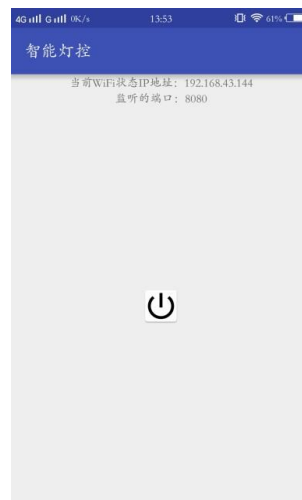


图 31.23 WiFi 模式灯控界面

(2) 点击手机 APP 灯控界面中间按钮, 可实现 LED 灯的开关与灯色的变换, 其效果同图 31.17。

4.3 语音灯控

4.3.1 实验准备

实验准备的具体内容请参照 4.1.1 节。

4.3.2 硬件连接

硬件连接的具体内容请参照 4.1.2 节。

4.3.3 系统测试

(1) 在 Android 手机 APP 中点击蓝牙模式, APP 请求蓝牙权限, 点击允许并搜索蓝牙设备, 如图 31.24 至图 31.26。

(2) 点击搜索到的对应蓝牙 4.0 模块的蓝牙信号进行配对后, 进入灯控界面, 如图 31.27。



图 31.24 初始界面



图 31.25 蓝牙请求



图 31.26 蓝牙搜索



图 31.27 控制界面

(3) 点击手机 APP 灯控界面中间按钮, 可以实现 LED 灯控制, 或者准确地读出单词 “open”、“change”、“close”, 可以实现语音控制 LED 灯的开关或者 LED 灯颜色的转变。打开 Arduino 软件的串口监视器, 如图 31.28 所示, 并观察串口监视器中的调试信息。



图 31.28 串口监视器

5 结果分析与讨论

本实验为智能灯控实验, 基于蓝牙和 WiFi 两种方式控制 LED 灯状态, 也可以使用语音命令方式控制 LED 灯状态。

物联网还有许多可扩展的内容, 本实验的 WiFi 模式是以手机端为服务器而进行的, 读者可以另外借助服务器, 实现远程控制, 从而进一步扩展本实验。

6 注意事项

在使用 WiFi 模式时, 会存在延时的现象。使用语音控制的过程中, 会存在识别失败的情况, 由于 PocketSphinx 离线语音的命令文件识别是采用相似音识别, 识别到声音时, 使用最相似的命令进行匹配, 所以发音不准时, 容易出错, 应当确保环境相对安静且语音命令较为准确。

实验三十二 基于 MQTT 协议的 温度实时监测

1 实验要求与目的

- 1 人独立完
- 了解 MQTT 协议的概念和基本原理
- 掌握 MQTT 协议的应用开发
- 掌握 Node.js 的开发原理

2 实验背景与原理

2.1 实验背景

消息队列遥测传输 (Message Queuing Telemetry Transport, MQTT, 相关理论内容见教材第 9 章) 是一个基于客户端-服务器的轻量级消息发布/订阅传输协议。可广泛应用于多种物联网应用场景中。

MQTT 协议运行在 TCP/IP 或其他网络协议, 提供有序、无损、双向连接, 其特点如下:

- (1) 使用发布/订阅消息模式, 提供一对多消息分发, 实现与应用程序的解耦。
- (2) 对负载内容屏蔽的消息传输机制。
- (3) 对传输消息有 3 种服务质量 (QoS):
 - 至多一次, 这一级别会发生消息丢失或重复, 消息发布依赖于底层 TCP/IP 网络。
 - 至少一次, 这一级别会确保消息到达, 但消息可能会重复。
 - 只有一次, 确保消息只有一次到达。在一些要求比较严格的计费系统中, 可以使用此级别。
- (4) 数据传输和协议交换最小化 (协议头部只有 2 字节), 减少网络流量。
- (5) 通知机制, 异常中断时通知传输双方。

2.2 实验原理

本实验的 MQTT 协议实现方式如图 32.1 所示: 发布者向代理发布温度信息; 代理收到信息后将数据存储到服务器, 并且服务器根据数据推送穿衣提示; 用户向代理订阅消息, 代理收到订阅请求后, 将温度数据和穿衣提示推送给用户。

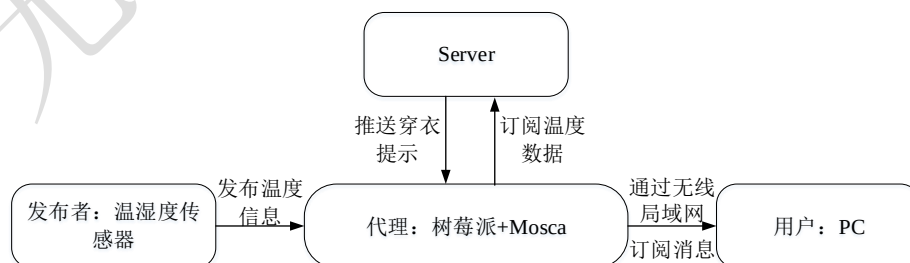


图 32.1 本实验 MQTT 协议实现原理图

3 实验环境与资源

(1) 环境资源:

树莓派套件 (树莓派 Raspberry Pi3 一个、8GB TF 卡一张、树莓派电源适配器一个、HDMI

转 VGA 线一根、VGA 线一根), USB 接口的键盘鼠标一套(自备)、有源显示器一个(自备)、PC 一台、温湿度传感器一个、杜邦线若干、Ubuntu MATE 系统镜像文件。

(2) 使用说明:

●Ubuntu MATE: 16.04.2 为官方长期维护的免费版本, 专供树莓派使用, 下载后的压缩包大小约为 1.2GB, 详细下载方法请见实验资源包 WNTLab/experiments/exp20, 官网地址 <https://ubuntu-mate.org/>。

●SDFormatter 软件: 作者使用版本为 4.0 版, 安装包大小约为 7MB, 下载方法请见实验资源包 WNTLab/experiments/exp32, 官网下载地址为 <http://www.softpedia.com/get/System/File-Management/SDFormatter.shtml>。

●Win32DiskImager 软件: 作者使用版本为 1.0 版, 安装包大小约为 12MB, 下载方法请见实验资源包 WNTLab/experiments/exp32, 官网下载地址为 <https://sourceforge.net/projects/win32diskimager/>。

4 实验步骤与演示

本实验内容为实现基于 MQTT 协议的温度监测系统。

4.1 硬件配置

(1) 本实验采用的主控板为树莓派 Raspberry Pi3, 其预留引脚分布如图 32.2 所示, 共 40 个引脚, 编号为 1~40。

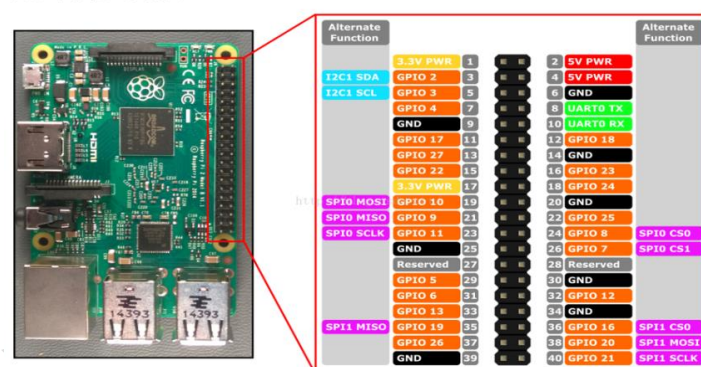


图 32.2 树莓派预留引脚分布图

(2) 使用母对母杜邦线, 按照图 32.2 所示方式, 连接树莓派和温湿度传感器。

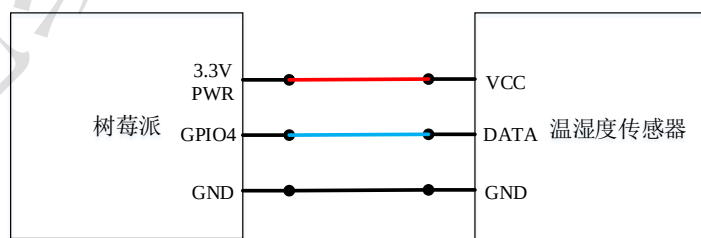


图 32.2 树莓派与温湿度传感器连接原理图

(3) 根据实验二十第 4.1 节介绍, 给 TF 卡烧录 Ubuntu MATE16.04.2 系统。

(4) 烧录完成后, 将烧录好的 TF 卡插入树莓派模块背面, 如图 32.3 所示, 并连接键盘、鼠标, 用 VGA 线和 HDMI 转 VGA 线连接显示器和树莓派。接上电源线, 开启电源, 等待系统启动。

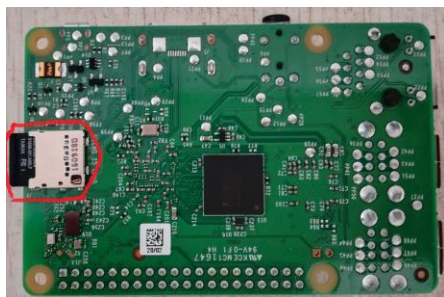


图 32.3 插入 TF 卡

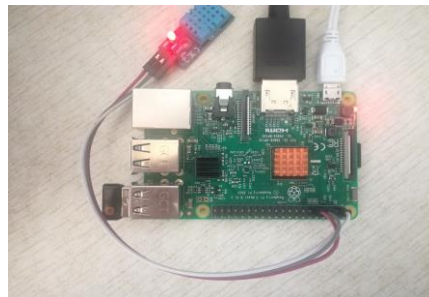


图 32.4 连接树莓派

(5) 系统启动成功后, 会出现初试配置界面, 依次设置: 系统语言(建议选择英文); 无线网络(选择不连接); 键盘布局(选择默认美式键盘布局); 位置信息(选择中国任意地区皆可); 用户信息(设置用户名、主机名、密码等)。设置完成后重启系统。

(6) 系统重启后, 连接 WiFi 网络, 如图 32.5 所示。使用快捷键“Ctrl+Alt+T”打开终端(Terminal), 使用 ping 命令检查树莓派是否连接到 Internet, 若收到如图 32.6 所示类似内容, 则代表树莓派已连接到网络。(需注意, 本实验中所有命令均在终端中运行。)

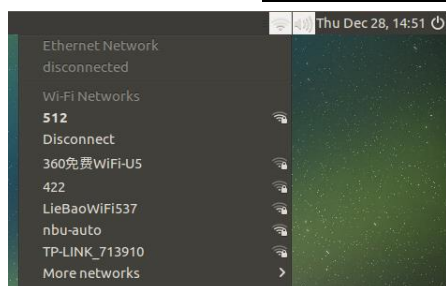


图 32.5 连接 WiFi

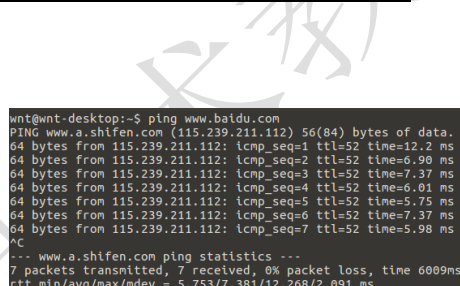


图 32.6 检查系统网络

4.2 开发环境配置

(1) 将本实验资源包 WNTLab/experiments/exp32 中的 MQTT-demo 文件夹拷贝到 Ubuntu Mate 系统中(可通过 U 盘方式进行文件拷贝), 如图 32.7 所示。

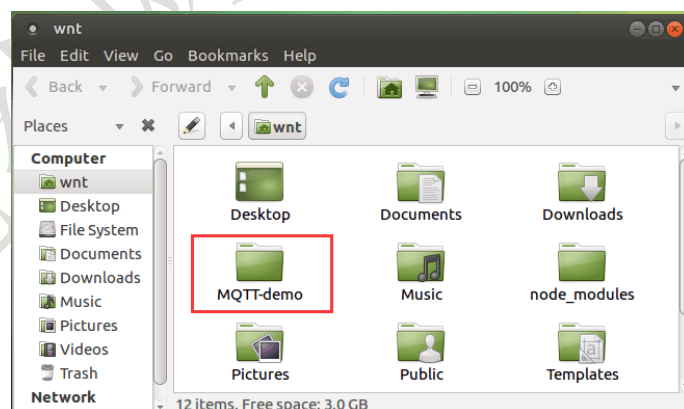


图 32.7 实验资源包

(2) 查询 Python 版本。打开 Terminal 终端, 执行命令 `python`, 查询 Python 的版本。如图 32.8 所示, 一般 Ubuntu 系统都预装有 Python, 若没有则需读者自行安装。


```
wnt@WNTLab:~
File Edit View Search Terminal Help
=====Welcome to use Contiki and NS2 for WNTLab!=====
If you have any questions, please contact wireless_net AT 163.com
wnt@WNTLab:~$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

图 32.8 查询 Python 环境

(3) 安装 npm 包管理工具和 n 模块。在 Terminal 终端中, 执行以下两条命令进行安装。

```
sudo apt-get install npm
```

```
sudo npm install -g n
```

(4) 安装最新稳定版 nodejs 并升级 npm。在 Terminal 终端中, 执行以下两条命令开始安装, 如图 32.9 和 32.10 所示。

```
sudo n stable
```

```
sudo npm install npm-g
```

```
wnt@WNTLab:~$ sudo n stable
install : node-v9.8.0
mkdir : /usr/local/n/versions/node/9.8.0
fetch : https://nodejs.org/dist/v9.8.0/node-v9.8.0-linux-armv7l.tar.gz
##### 100.0%
installed : v9.8.0
```

图 32.9 安装最新稳定版 nodejs

```
wnt@WNTLab:~$ sudo npm install npm -g
[.....] - loadRequestedDeps: sill install loadAllDepsIntoIdealTree
/usr/local/bin/npm -> /usr/local/lib/node_modules/npm/bin/npm-cli.js
/usr/local/bin/npmx -> /usr/local/lib/node_modules/npm/bin/npmx-cli.js
+ npm@5.8.0
added 89 packages, removed 42 packages and updated 67 packages in 218.964s
```

图 32.10 升级 npm

安装完成后, 使用命令 `node -v` 查询 nodejs 版本, 检查是否成功安装, 如图 32.11 所示。

```
wnt@WNTLab:~$ node -v
v9.8.0
```

图 32.11 查询 Node.js 版本

(5) 安装 MQTT 的代理服务器 Mosca。执行命令 `sudo npm install mosca --save`, 安装过程如图 32.12 所示。

```
wnt@WNTLab:~$ sudo npm install mosca --save
npm WARN saveError ENOENT: no such file or directory, open '/home/wnt/package.js
on'
npm WARN enoent ENOENT: no such file or directory, open '/home/wnt/package.json'
npm WARN wnt No description
npm WARN wnt No repository field.
npm WARN wnt No README data
npm WARN wnt No license field.

+ mosca@2.8.0
updated 1 package in 17.926s
```

图 32.12 安装 MQTT 代理服务器 Mosca

(4) 安装 MQTT 实验环境。执行命令 `sudo npm install mqtt --save`, 安装过程如图 32.13 所示。

```
wnt@WNTLab:~$ sudo npm install mqtt --save
npm WARN saveError ENOENT: no such file or directory, open '/home/wnt/package.js
on'
npm WARN enoent ENOENT: no such file or directory, open '/home/wnt/package.json'
npm WARN wnt No description
npm WARN wnt No repository field.
npm WARN wnt No README data
npm WARN wnt No license field.

+ mqtt@2.16.0
added 14 packages from 12 contributors and updated 2 packages in 41.284s
```

图 32.13 安装 MQTT 实验环境

(5) 安装 Koa。Koa 是 Node.js 下的 Web 框架，本实验将用它来编写 Web 应用。执行命令 `sudo npm install koa --save`，如图 32.14 所示。

```
wnt@WNTLab:~$ sudo npm install koa --save
npm WARN saveError ENOENT: no such file or directory, open '/home/wnt/package.js
on'
npm WARN enoent ENOENT: no such file or directory, open '/home/wnt/package.json'
npm WARN wnt No description
npm WARN wnt No repository field.
npm WARN wnt No README data
npm WARN wnt No license field.

+ koa@2.5.0
added 30 packages from 12 contributors in 110.865s
```

图 32.14 安装 Koa 框架

4.3 实验过程

(1) 树莓派运行 Mosca.js 脚本。打开 Terminal，使用 `cd` 命令将路径切换为 Mosca.js 脚本文件所在路径，并执行命令 `node Mosca.js`，如图 32.15 所示。

```
wnt@WNTLab:~$ cd MQTT-demo/src/broker/
wnt@WNTLab:~/MQTT-demo/src/broker$ node Mosca.js
Mosca server is up and running
```

图 32.15 运行 Mosca.js 脚本

执行 Node Mosca.js 后，本地的一个 Broker 运行在 localhost 的 1883 端口上。Published 表示向 Mosca 代理服务器发布的信息内容

(2) 树莓派运行 MQTT.js 脚本。使用 `cd` 命令将路径切换为 MQTT.js 脚本文件所在路径，并执行命令 `node MQTT.js`，向 MQTT 发布温度信息，具体结果如图 32.16 所示。

```
wnt@WNTLab:~$ cd MQTT-demo/src/temperature/
wnt@WNTLab:~/MQTT-demo/src/temperature$ node MQTT.js
{"humidity": 29, "temperature": 25}
>>> connected
```

图 32.16 运行 MQTT.js 脚本

temperature/目录下还存在一个 data_fetch.py 文件，该文件由 Python 语言编写，用于驱动树莓派硬件 I/O 与温湿度传感器。MQTT.js 调用 data_fetch.py 文件，完成发布温湿度信息的任务。

执行 Node MQTT.js 后，Client 端就已启动，可以看到在 MQTT.connect 方法中连接了上一节中启动的 Broker 地址，连接成功后，Client 会输出日志，“>>> connected”，Broker 的控制台也会输出 Client 的连接信息。

(3) 树莓派运行 server.js 脚本。使用 `cd` 命令将路径切换为 server.js 脚本文件所在路径，并执行命令 `node server.js`，程序向代理服务器订阅温度信息，并对温度进行分析，向服务器发布穿衣提示，如图 4.5.17 所示。

```
wnt@WNTLab:~$ cd MQTT-demo/src/server/
wnt@WNTLab:~/MQTT-demo/src/server$ node server.js
>>> connected
{"temperature":25,"tips":"今天天气正好, 可以穿上一件薄衣服"}
{"temperature":25,"tips":"今天天气正好, 可以穿上一件薄衣服"}
{"temperature":25,"tips":"今天天气正好, 可以穿上一件薄衣服"}
```

图 32.17 运行 server.js 脚本

Server 使用 MQTT.js 订阅 Client 发送到 temperature Topic 的数据进行处理, 把处理后的数据转译成 JSON 发送到另一业务主题/tips 中 (JSON 是保存数据的一种格式)。

(4) 树莓派运行 koa.js 脚本。使用 **cd** 命令将路径切换为 koa.js 脚本文件所在路径, 并执行命令 **node koa.js**。

```
wnt@WNTLab:~$ cd MQTT-demo/src/app/
wnt@WNTLab:~/MQTT-demo/src/app$ node koa.js
>>> connected
{"temperature":25,"tips":"今天天气正好, 可以穿上一件薄衣服"}
今天天气正好, 可以穿上一件薄衣服
{"temperature":25,"tips":"今天天气正好, 可以穿上一件薄衣服"}
今天天气正好, 可以穿上一件薄衣服
```

图 32.18 运行 koa.js 脚本

(5) 在 PC 端, 打开浏览器, 在浏览器的地址栏输入树莓派的 ip 地址以及 web 端所运行的端口号(3000), 例如: 192.168.1.107:3000。具体结果如图 32.18 和 32.19 所示:

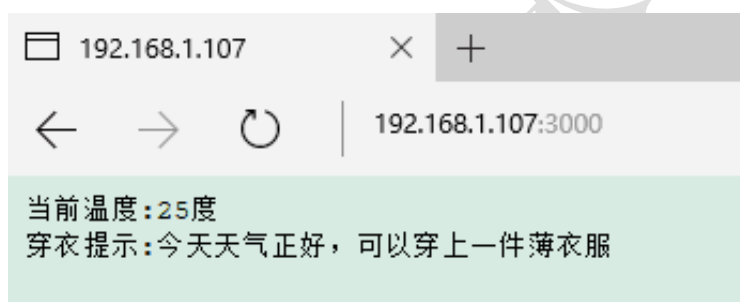


图 32.19 web 端收到信息

5 结果分析与讨论

本实验在树莓派上进行开发, 实现了用 MQTT 协议传输数据, 并体现出了 MQTT 传输数据的高效性。并介绍了如何使用 Python 对数据进行处理, 以及 Node.js 脚本语言的使用。本实验的结果以图形的方式显示, 读者可自行进行分析。

6 注意事项

本实验开发所有源代码, 供读者进行二次开发。读者在掌握了 linux 命令, Python 以及 Node.js 编程的基础上, 可以用 Android 端实现客户端接收 MQTT 消息的推送, 功能与本实验相同。

实验三十三 低功耗广域网(LoRa)数据传输

1 实验目的和要求

- 2 人共同完成
- 了解 LoRa 相关的通信原理
- 使用 LoRa 技术实现数据传输

2 实验背景与原理

2.1 实验背景

LoRa 是一种低功耗广域网通信技术，是美国 Semtech 公司推出的一种低功耗远距离无线传输方案。为用户提供一种简单的能实现远距离(几百上千米)、长电池寿命、大容量的系统，有效扩展了传统单跳仅几十米的 Zigbee 无线传感网。相关理论内容请参考教材第 9 章。

本实验涉及 Arduino 节点安装与开发，请参考实验十六内容。

2.2 实验原理

LoRa 技术的特点是低功耗远距离传输，因此实验场地建议选在户外。作者实验场地如图 33.1 所示，地图标记显示两节点间距离约为 450 米，读者亦可尝试更远距离。



图 33.1 实验场地

实验内容分为 2 部分:

- (1) 通过 LoRa 节点实现节点间数据传输, 如图 33.2 所示。
- (2) 通过 LoRa 节点获取环境温湿度数据发送至物联网服务器, 如图 33.3 所示。



图 33.2 LoRa 节点间数据传输原理图

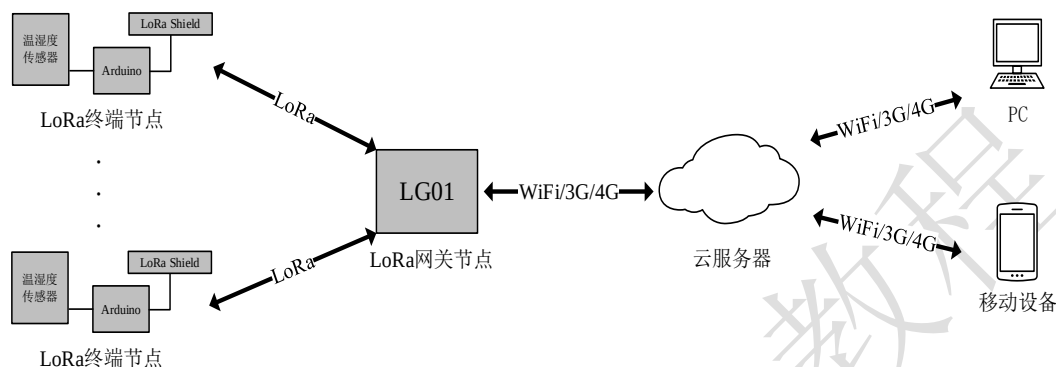


图 33.3 LoRa 物联网服务器数据传输原理图

3 实验环境与资源

(1) 环境资源:

PC 一台、Arduino Mega 2560 单片机模块 2 个、LoRa 模块 2 个、LoRa 单通道网关 1 个、温湿度传感器 1 个、USB A 型 B 型线 1 条、电池 2 对、Arduino IDE。

(2) 使用说明:

- 各类硬件如图 33.4~33.9 所示，详细的使用方法见实验步骤；
- LoRa 模块是基于 Arduino 接口和基于开源库的长距离收发器。该模块允许用户以低数据速率发送数据并达较长通信距离。它提供超长距离扩展通信和高抗干扰性，同时最小化电流消耗。本实验采用 LoRa 模块通信频率为 433 MHz。
- LoRa 单通道网关可将 LoRa 网络通过 WiFi、以太网口、3G/4G 连接到 Internet。该网关在开源嵌入式 Linux 系统上运行，有一个 USB 主机端口，2 个以太网口和 802.11 b/g/n WiFi 功能。该网关支持的 LoRa 通信频率为 433MHz。
- Arduino IDE 是 Arduino 系列单片机的开发环境，本实验需要较高版本的 Arduino 基础开发库，作者采用 1.8.5 版本。官网下载链接为 <https://www.arduino.cc/en/Main/Software>，详细下载方法及说明请见实验资源包 WNTLab/experiments/exp33。
- 本实验提供相关源代码，供读者学习与二次开发，详见实验资源包 WNTLab/experiments/exp33。



图 33.4 Arduino Mega 2560 单片机模块



图 33.5 USB A 型转 B 型线

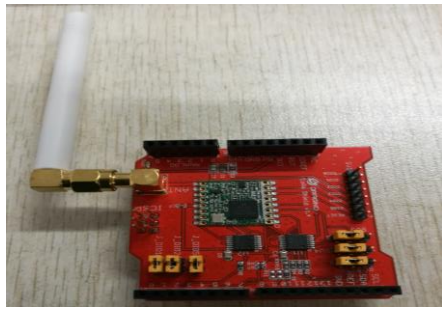


图 33.6 LoRa 模块



图 33.7 LoRa 单通道网关

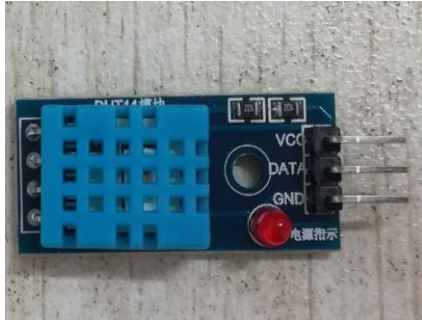


图 33.8 温湿度传感器



图 33.9 电池

4 基础实验步骤与演示

4.1 开发平台的配置

为实现 LoRa 节点间的数据传输, 首先需要为 Arduino IDE 添加 LoRa 节点开发库, 可通过在线方式进行安装, 步骤如下:

(1) 打开 Arduino IDE 界面, 通过点击项目->加载库->管理库, 打开库管理器界面, 如图 33.10、33.11 所示。

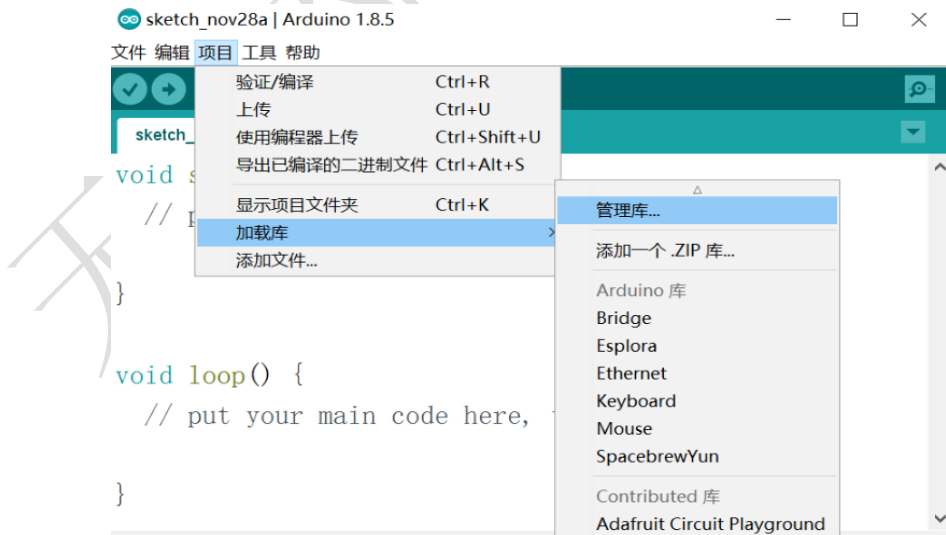


图 33.10 打开管理库

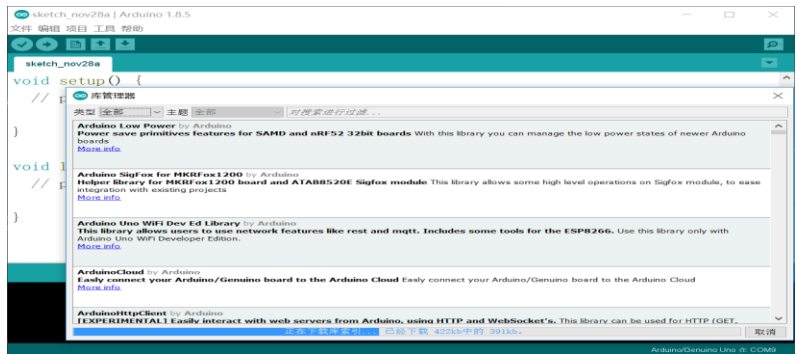


图 33.11 库管理器界面

(2) 在库管理器界面的搜索栏输入关键字“lora”，搜索开发库，如图 33.12 所示。

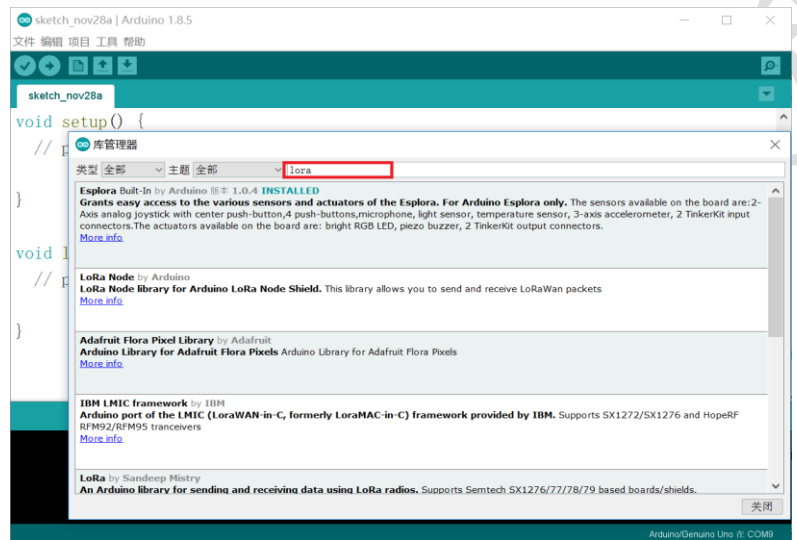


图 33.12 输入关键字“lora”

(3) 选择“LoRa”开发库，点击“安装”按钮，安装最新版本，如图 33.13 所示。

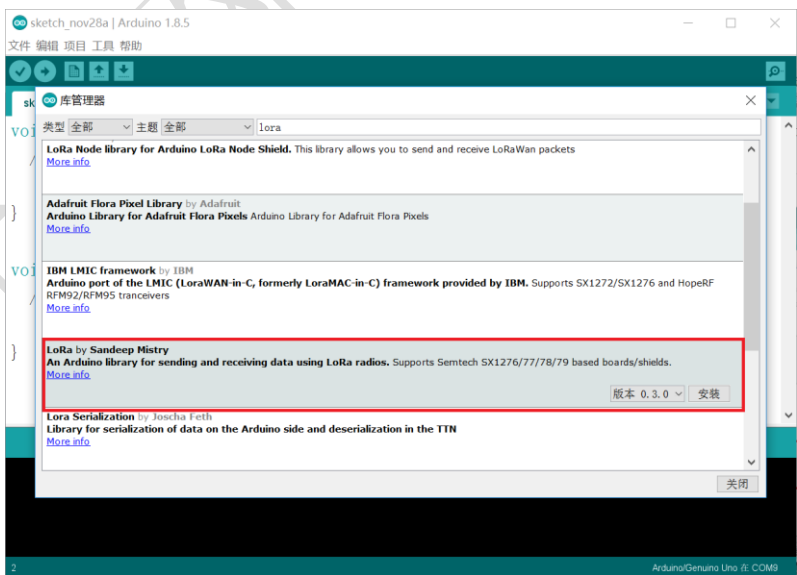


图 33.13 安装 LoRa 开发库

4.2 LoRa 节点组建

(1) 由于本实验采用的 LoRa 模块基于 Arduino 接口，所以仅需将 LoRa 模块按照其对应

接口插在 Arduino Mega 2560 单片机上即可, 并将天线固定至 LoRa 模块, 如图 33.14 所示。



图 33.14 LoRa 节点组建

(2) 这里需将 2 个 LoRa 节点分别烧录本实验提供的 LoRaSender 和 LoRaReceiver 文件代码, 将其分别配置为 LoRa 发送节点和 LoRa 接收节点。实验代码编译、烧录与结果查看过程与实验十六相同, 此处不再赘述。

4.3 LoRa 点对点通信

(1) 将 LoRa 发送节点和 LoRa 接收节点通过电池供电, 如图 33.15 所示。

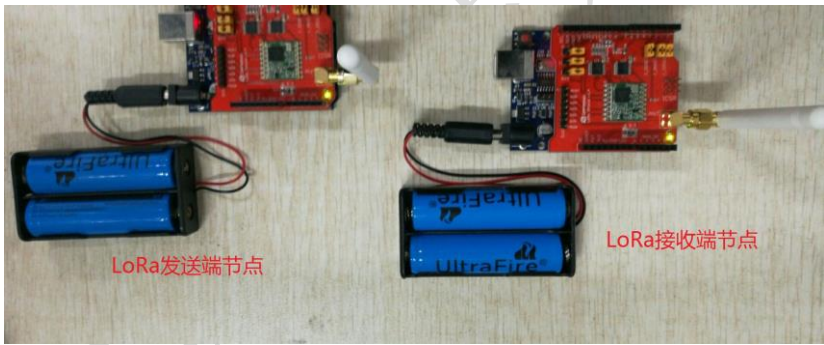


图 33.15 将 LoRa 节点通过电池供电

(2) 分别将两个 LoRa 节点通过 USB A 型 B 型线连接至 PC, 通过 Arduino IDE 串口监视器查看 (设置串口波特率为 9600)。可发现发送节点所发送的数据和接收节点所接收的数据一致, 由此确认这两个 LoRa 节点已正常通信, 串口监测数据如图 33.16、33.17 所示。

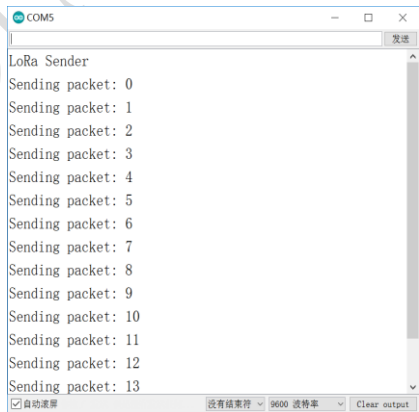


图 33.16 LoRa 发送端串口数据

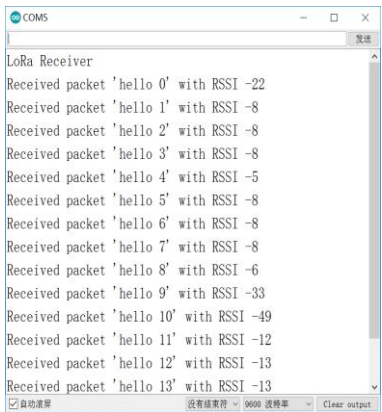


图 33.17 LoRa 接收端串口数据

5 扩展实验步骤与演示

5.1 开发平台的配置

为实现 LoRa 物联网服务器的数据传输, 需要为 Arduino IDE 添加 LoRa 网关开发库, 步骤如下:

(1) 打开 Arduino IDE 界面, 点击文件->首选项, 打开首选项界面, 如图 33.18、33.19 所示。



图 33.18 打开首选项

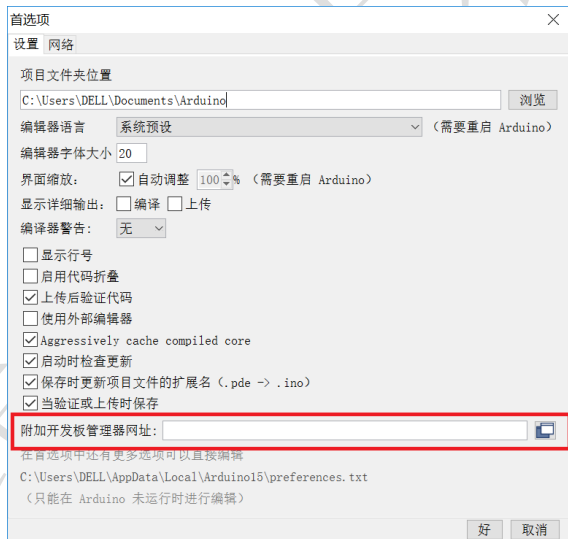


图 33.19 首选项界面

(2) 在首选项界面中的“附加开发板管理器网址”栏中输入添加 LoRa 网关开发库 URL: http://www.dragino.com/downloads/downloads/YunShield/package_dragino_yun_test_index.json 如图 33.20 所示。

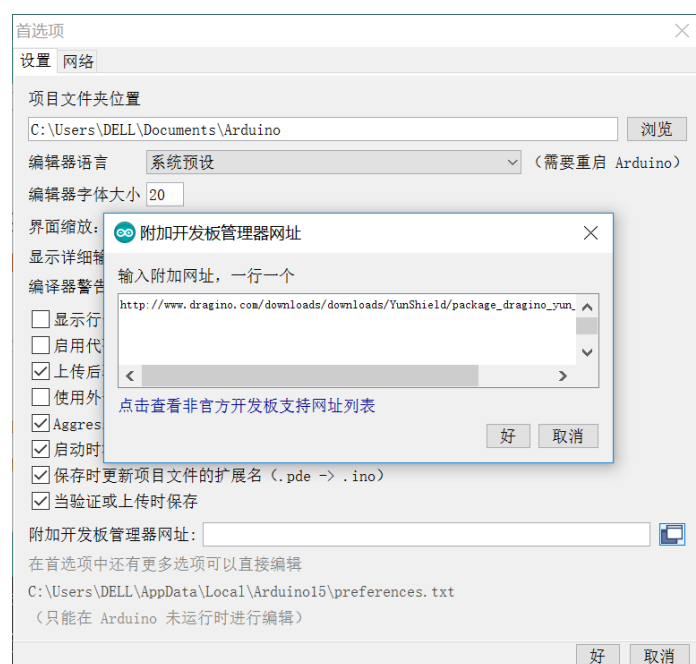


图 33.20 添加 LoRa 网关开发库 URL

(3) 点击工具->开发板->开发板管理器..., 打开“开发板管理器”界面, 如图 33.21、33.22 所示。



图 33.21 打开开发板管理器

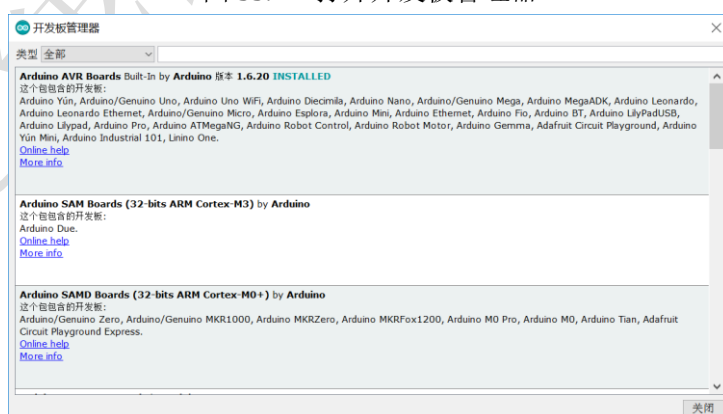


图 33.22 开发板管理器界面

(4) 在开发板管理器界面中的搜索栏输入关键字“dragino”，找到 Dragino Yun 开发库，点击“安装”按钮，安装最新版本，如图 33.23 所示。

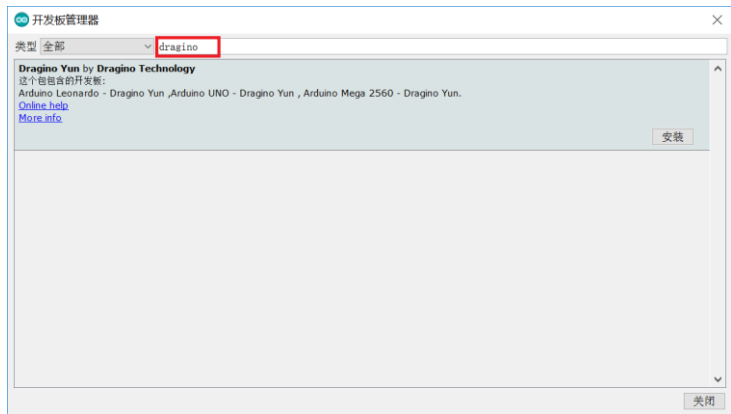


图 33.23 安装 Dragino Yun 开发库

(5) 打开 Arduino IDE 安装路径下的 libraries 文件夹，并将本实验资源包提供的 RadioHead 和 ThinkSpeak 开发库文件夹拷贝至当前路径，如图 33.24 所示。

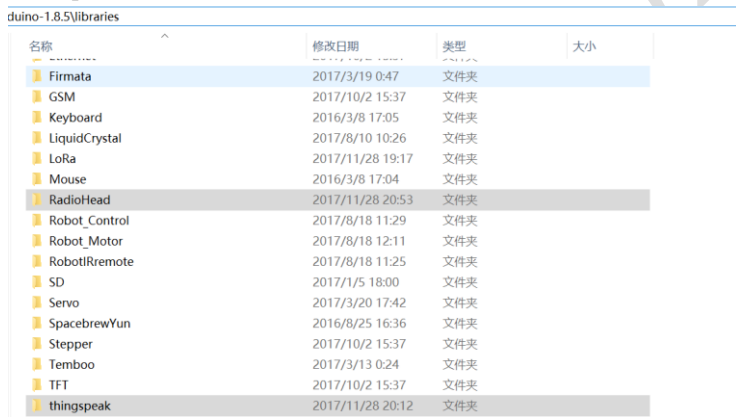


图 33.24 添加 RadioHead 和 ThinkSpeak 开发库

5.2 LoRa 终端节点与 LoRa 网关的组建

(1) LoRa 终端节点的组建方法同 4.2 节内容，此外需将温湿度传感器与 LoRa 终端节点连接。可使用公对母杜邦线将二者相连，连接原理图如图 33.25 所示。

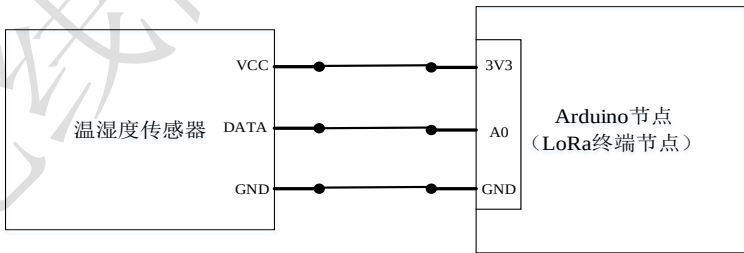


图 33.25 温湿度传感器连接原理图

(2) LoRa 网关可被视为一般无线路由器，将 LoRa 网关的 WAN 口通过网线接入 Internet，将电源口通过电源适配器供电，如图 33.26 所示。



图 33.26 LoRa 网关接入 Internet

(3) 此时 PC 可搜索到名为 dragino-xxxxxx 的网络名, 如图 33.27 所示。注意, 若 PC 没有无线网卡, 请将无线网卡插在 PC 的 USB 接口。



图 33.27 搜索 dragino 网络

(4) LoRa 网关的默认 IP 地址为 10.130.1.1, 可在 PC 浏览器中输入 10.130.1.1 进入网关的登录界面。登录账号默认为 root, 密码默认为 dragino, 如图 33.28 所示。

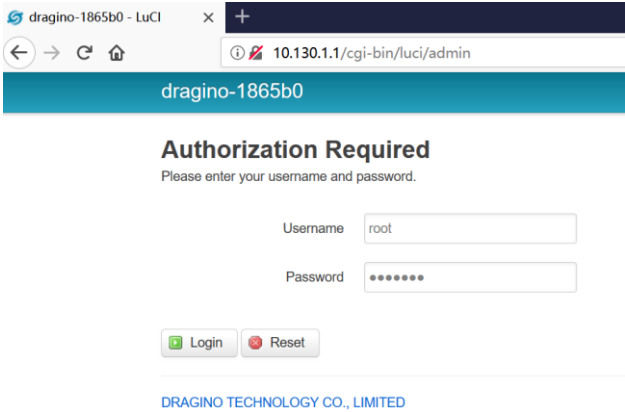


图 33.28 进入 LoRa 网关 Web 界面

5.3 部署物联网服务器

为简化 LoRa 物联网服务器部署过程, 本实验采用现有的开源物联网平台 ThingSpeak 作为测试服务器, 步骤如下:

(1) 进入 https://thingspeak.com/users/sign_up, 注册 ThingSpeak 物联网服务器账号, 如图 33.29 所示。

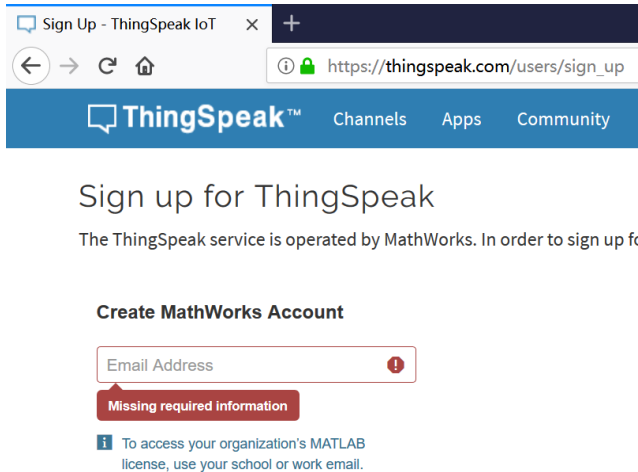


图 33.29 注册 ThingSpeak 物联网服务器账号

(2) 登录刚注册的账号, 点击 Channels 栏, 如图 33.30 所示。点击 New Channels 按钮, 创建数据通道, Field 为欲采集的数据单元, 本实验为温度和湿度, 最后保存通道数据。如图 33.31 所示。

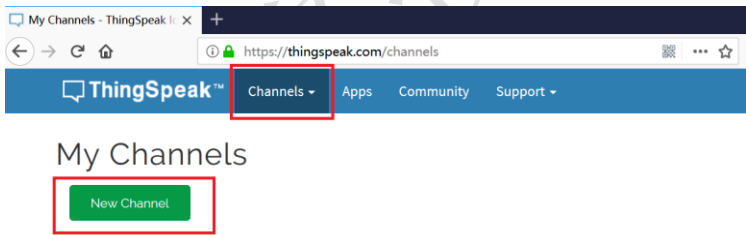


图 33.30 创建数据通道

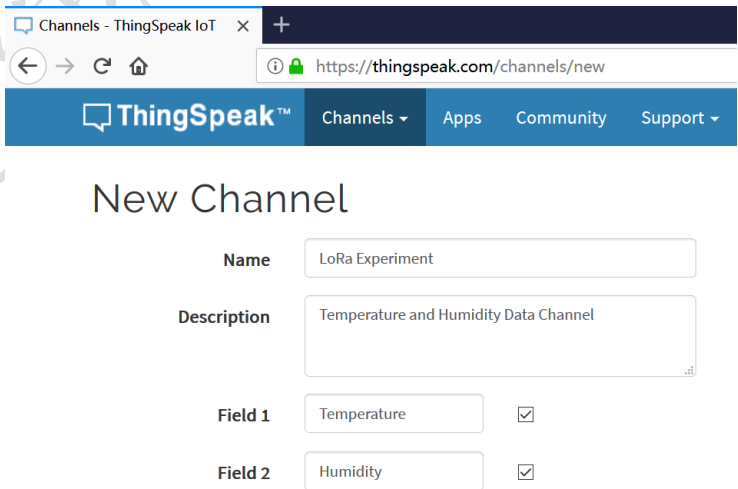


图 33.31 创建温湿度数据通道

(3) 完成数据通道创建后, 点击已创建的通道中的 API Keys 按钮, 查询该通道的 ID 和 API Key, 如图 33.32、33.33 所示。由于这里仅对服务器进行数据上传, 所以只需记录该通

道的 ID 和 Write API Key, 用于 LoRa 网关节点上传数据时的安全验证。

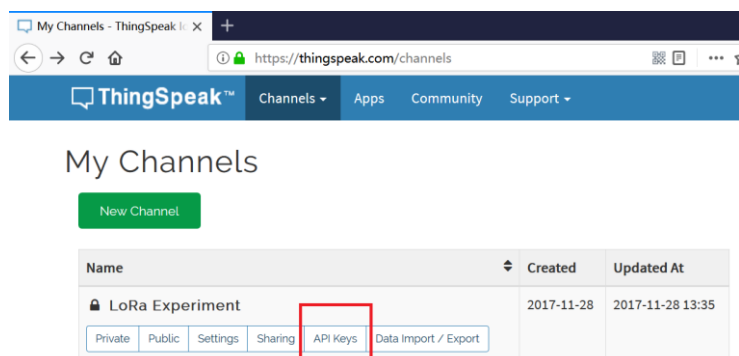


图 33.32 查询通道 API Keys

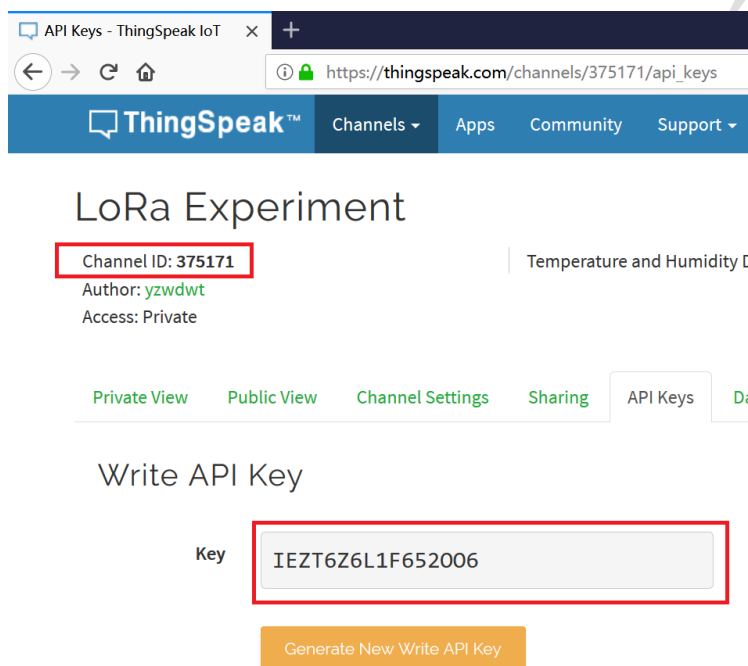


图 33.33 通道的 ID 和 API Key

5.4 代码烧录及测试

(1) 给 LoRa 终端节点烧录本实验提供的 dht11Client 源代码, 详细过程请参考实验十六。并将 LoRa 终端节点通过电池供电, 如图 33.34 所示。

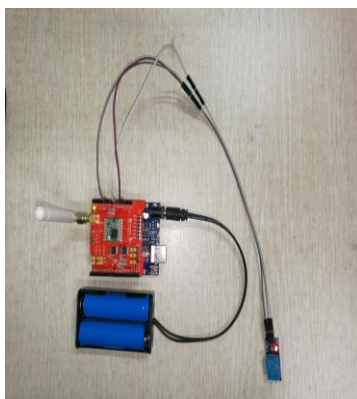


图 33.34 LoRa 终端节点通过电池供电

(2) 给 LoRa 终端节点烧录本实验提供的 dht11Server 源代码, 详细过程请参考实验十六。烧录之前, 需要将源代码中的 myChannelNumber 与 myWriteAPIKey 值修改为上述注册的 ThingSpeak 通道 ID 和 Write API Key, 如图 33.35 所示。

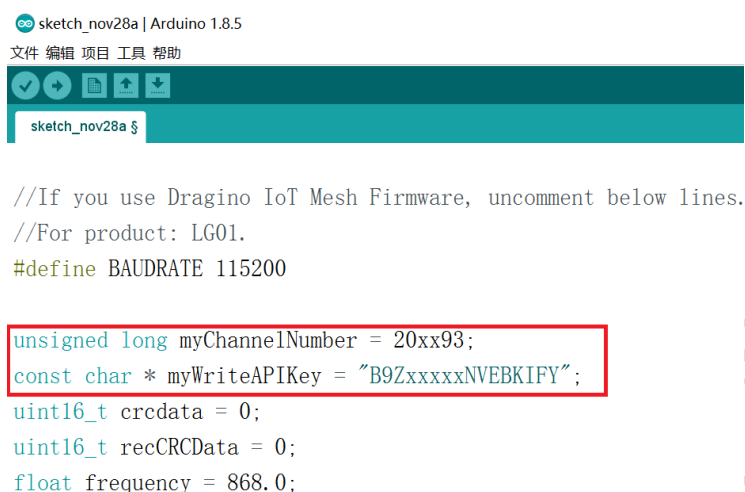


图 33.35 dht11Server 源代码

(3) 给 LoRa 网关烧录 dht11Server 文件代码, 首先点击工具->开发板, 将开发板类型切换为 Dragino Yun + UNO or LG01/OLG01, 如图 33.36 所示。

由于此时 PC 已接入 Dragino 网络, 可通过网络端口对 LoRa 网关进行代码烧录。点击工具->端口, 选择 10.130.1.1(Arduino Yun)网络端口, 如图 33.37 所示。注意, 若读者 PC 无法搜索到网络端口, 请将无线网卡插在 PC 的 USB 接口, 切换至该无线网卡, 重新接入 Dragino 网络。烧录过程中若需要输入开发板密码, 请输入密码 dragino。

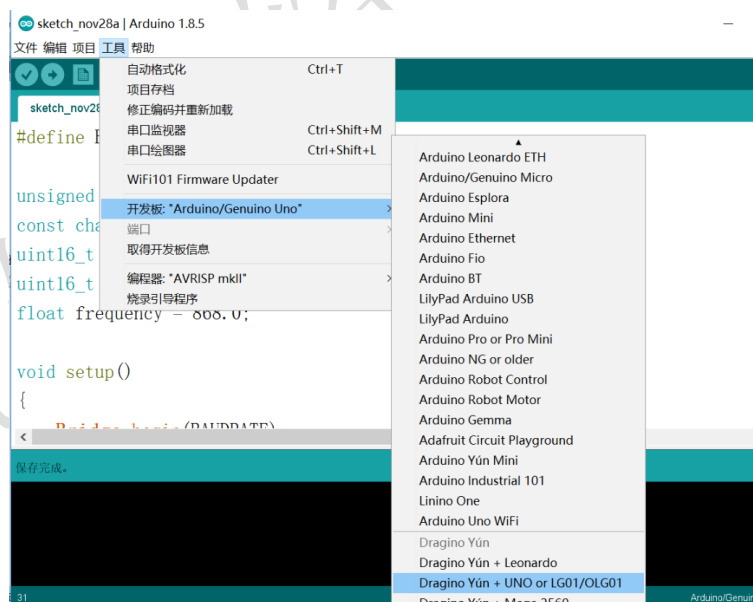


图 33.36 切换开发板类型



图 33.37 切换网络端口

(4) 将 LoRa 终端节点使用 USB 转串口线连接至 PC, 通过 Arduino IDE 串口监视器查看, 可确认 LoRa 终端节点与 LoRa 网关成功进行通信, 串口结果如图 33.38 所示。

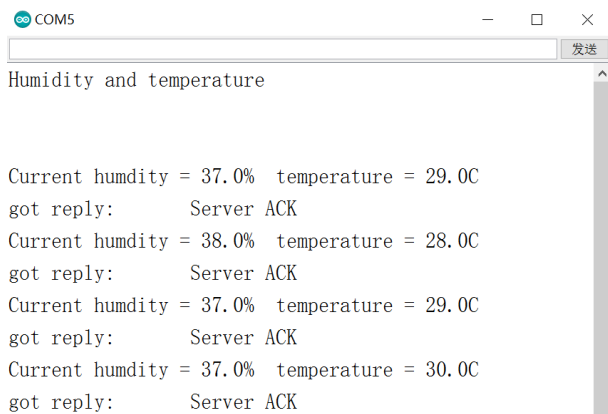


图 33.38 LoRa 终端节点串口结果

(5) 查看之前在 ThingSpeak 上所创建的通道状态, 可实时监测 LoRa 终端节点上传的温湿度信息, 如图 33.39 所示。注意, 本实验源码将 LoRa 终端节点传输间隔设定为 30s, 读者可自行修改。

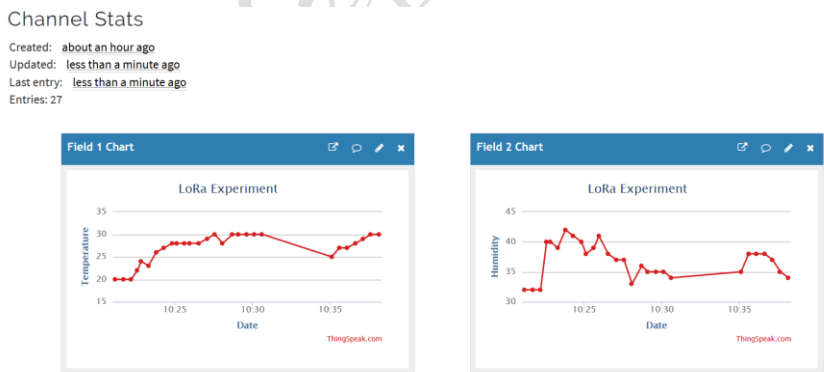


图 33.39 实时监测 LoRa 终端节点的温湿度信息

5 结果分析与讨论

本实验第一部分实现了 LoRa 节点间的数据传输。读者可尝试在户外进行实验测试, 体验 LoRa 长距离通信的特点。可通过设置 LoRa 协议中的扩频因子, 调节节点间的传输速率与通信最大距离。

第二部分实现了 LoRa 终端节点与物联网服务器间的数据传输。读者可增加其他传感器, 实现更完善的数据监测系统。

实验三十四 机器人环境配置与开发基础

1 实验要求与目的

- 1人独立完成
- 掌握基于 ROS 的机器人开发环境构建过程
- 理解 ROS 的开发基础

2 实验背景与原理

2.1 实验背景

Robot Operating System(ROS)是一个机器人软件框架。它提供了操作系统应有的服务,包括硬件抽象、底层设备控制、常用函数的实现、进程间消息传递、包管理等。

ROS 的主要目标是为研究和开发机器人提供代码复用的支持。ROS 是一个分布式的进程框架,这些进程被封装在易于分享和发布的程序包和功能包中。ROS 也支持一种类似于代码储存库的联合系统,可实现工程的协作及发布。这个设计可以使一个工程的开发和实现从文件系统到用户接口完全独立决策(不受 ROS 限制)。

更多关于 ROS 请参考博客: <http://www.dengke.xyz/archives/category/robot>。

2.2 实验原理

(1) 本实验所用机器人结构如图 34.1 所示, Raspberry Pi 3 作为主控板, 用激光雷达实现即时定位与地图构建(Simultaneous Localization And Mapping, SLAM)基础功能, Arduino、TB6612FNG 作为电机驱动模块。

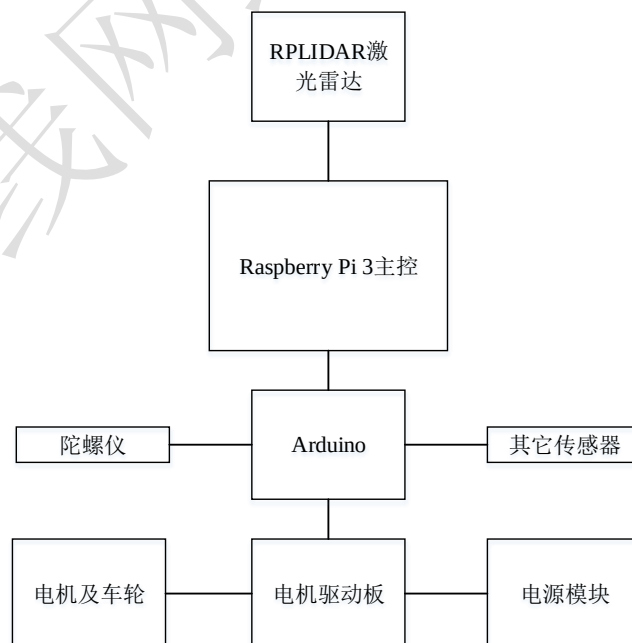


图 34.1 移动式机器人结构

(2) 本实验系统中, 机器人相关实验均通过 PC 端 Ubuntu 虚拟机控制移动式机器人进行

实验, 二者通过 WiFi 通信, 原理如图 34.2 所示。本实验为机器人基础实验, 将搭建机器人开发环境。

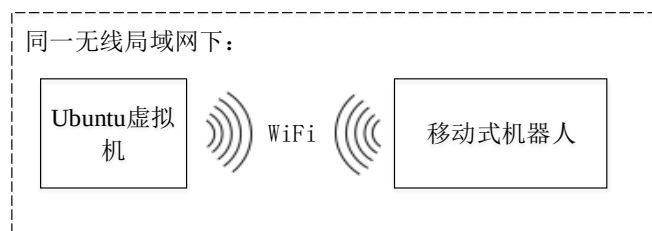


图 34.2 机器人控制原理

3 实验环境与资源

(1) 环境资源:

移动式机器人套件（树莓派 Raspberry Pi3、TF 卡、电机驱动板、激光雷达、电源、车体等）、树莓派电源适配器一个、HDMI 转 VGA 线一根、VGA 线一根、USB 接口的键盘鼠标一套（需自备）、有源显示器一个（需自备）、PC 一台、wnt-robot 系统镜像、robot-node 系统镜像、SDFormatter 软件、Win32DiskImager 软件、VMware Workstation 软件。

(2) 使用说明

- wnt-robot、robot-node 系统镜像文件为作者制作, 读者可从本实验教学系统官网获取:
<http://www.thinkmesh.net/wireless/>。

- SDFormatter 软件: 作者使用版本为 4.0 版, 安装包大小约为 7MB, 下载方法请见实验资源包 WNTLab/experiments/exp34, 官网下载地址为 <http://www.softpedia.com/get/System/File-Management/SDFormatter.shtml>。

- Win32DiskImager 软件: 作者使用版本为 1.0 版, 安装包大小约为 12MB, 下载方法请见实验资源包 WNTLab/experiments/exp34, 官网下载地址为 <https://sourceforge.net/projects/win32diskimager/>。

- VMware Workstation 软件: 作者使用版本为 VMware Workstation 12 Pro, 安装包大小约为 470MB, 下载方法请见实验资源包 WNTLab/experiments/exp34, 官网下载地址为:
<https://www.vmware.com/cn.html>。



图 34.3 HDMI 转 VGA 线



图 34.4 树莓派电源适配器



图 34.5 VGA 线



图 34.6 移动式机器人

4 实验步骤与演示

4.1 实验准备

4.1.1 搭建无线局域网

根据 2.2 节实验原理介绍, 机器人和 Ubuntu 虚拟机通过 WiFi 通信, 因此首先需要组建无线局域网, 步骤如下 (需注意, 为了便于管理, 建议读者组建自己的无线局域网, 而非使用公用 WiFi 网络。):

(1) 根据实验三第 4.1 节步骤, 组建无线局域网, 其中工作模式选择为“AP”。选择不接入 Internet, 实验在本地局域网下进行即可。

(2) 进入无线路由器管理界面, 点击左侧“DHCP 服务器-静态地址分配”按钮, 进入如图 34.7 所示界面, 点击“添加新条目”便可以为某个终端设备分配静态局域网 IP 地址。请读者尝试操作, 后续实验步骤将用到此功能。



图 34.7 静态地址分配界面

4.1.2 导入系统镜像

本实验需要 wnt-robot、robot-node 两个系统镜像, 其中 wnt-robot 为 PC 端的 Ubuntu 虚拟机文件, robot-node 为机器人中树莓派使用的系统 (需注意, 为了方便, 读者应先按本节所述方法导入镜像后, 再根据 4.1.3 所述组建机器人硬件平台。)。导入系统镜像的步骤如下:

(1) 根据第 3 节准备好系统镜像与相关软件。

(2) 在 PC 端连接组建好的 WiFi 网络。(需注意, 如果 PC 连接了有线网络, 请在系统的网络连接中禁用有线连接, 以确保 PC 使用的是该无线局域网。)

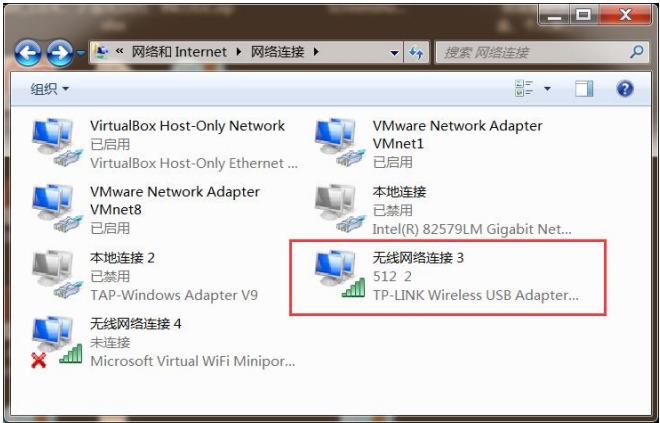


图 34.8 连接组建好的 WiFi 网络

- (3) 参照实验一第 4.2 节所述方法, 在 PC 端使用 VMware 软件导入 wnt-robot 虚拟机镜像。
- (4) 参照实验一第 4.1 节所述方法, 在树莓派中导入 robot-node 系统镜像。并连接组建好的 WiFi 网络, 此后树莓派重新开机均会自动连接该 WiFi 网络。
- (5) 根据 4.1.1 第(2)步所述方法, 进入无线路由器管理界面, 为树莓派分配一个有效的静态 IP 地址, 如图 34.9 所示。



图 34.9 为树莓派分配静态局域网 IP 地址

4.1.3 组建机器人硬件平台

本实验所用机器人为移动式机器人, 请读者参考实验教学视频组建该硬件平台, 下述为组建过程中的注意事项:

- (1) 确认已将 robot-node 系统镜像导入树莓派后, 再开始组建。
- (2) 准备好移动式机器人套件, 主要包含树莓派 Raspberry Pi3、TF 卡、电机驱动板、激光雷达、电源、车体等配件, 读者需再自行准备螺丝刀等工具。
- (3) 本实验所用机器人分为上中下两层: 下层连接电机与车轮, 中层搭载树莓派和电机驱动板; 上层搭载激光雷达。读者应从下到上进行组装。

4.2 ROS 多机通信基础

根据 2.2 节实验原理所述, 本实验通过 PC 控制移动式机器人, 因此需要先配置 ROS 多机通信的环境, 具体步骤如下:

- (1) 打开 PC 端 wnt-robot 虚拟机和机器人电源开关, 并确保二者都接入组建好的无线局域网。
- (2) 在虚拟机中打开 Terminal(快捷键“Ctrl+Alt+T”), 执行以下命令编辑 hosts 文件, 将如图 34.10 红色方框所示处 robot.local 的值修改为 4.1.3 节第(5)步中分配的静态 IP, 修改后保存退出。

`sudo gedit /etc/hosts`

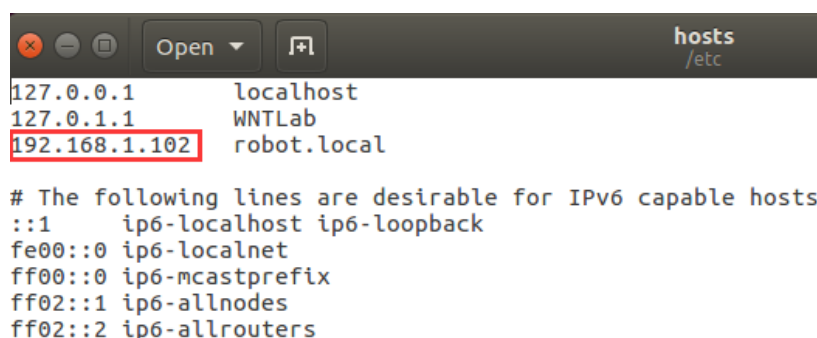


图 34.10 修改 hosts 文件

- (3) 执行以下命令编辑.bashrc 文件，确保最后几行内容如图 11 所示。

`sudo gedit .bashrc`

这几行内容分别是 ROS 环境变量设置、本机 IP/名称、节点管理器设置。如果内容不同，请读者根据图 34.11 修改。

```
source /opt/ros/kinetic/setup.bash
export ROS_IP='hostname -I'
export ROS_HOSTNAME='hostname -I'
export ROS_MASTER_URI=http://robot.local:11311
```

图 34.11 确认.bashrc 文件配置

- (4) 执行以下命令，如果收到类似如图 34.12 所示内容，则配置完成。

`ping robot.local`

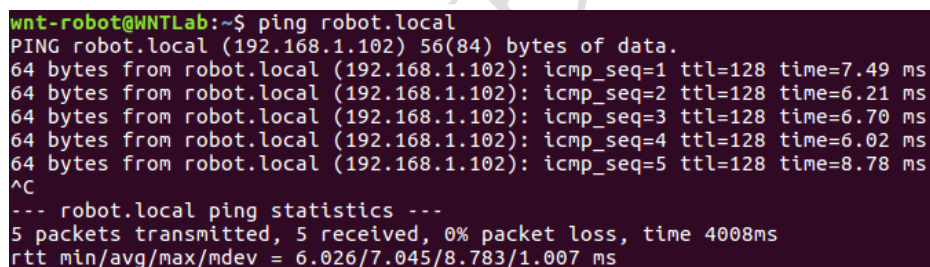


图 34.12 测试配置

4.3 控制机器人基础运动

ROS 中，功能包(package)是一种特定结构的文件/文件夹的组合。它是 ROS 中各种软件的组织方式，包含节点、配置文件、ROS 依赖库、第三方软件及其它任何逻辑构成。

本节实验将使用一个基本功能包实现 PC 端控制机器人进行基本运动，步骤如下：

- (1) 打开 PC 端 wnt-robot 虚拟机和机器人电源开关。

- (2) 在虚拟机中打开一个 Terminal 窗口(快捷键“Ctrl+Alt+T”), 执行以下命令，并输入密码 123456，即实现远程登录到机器人树莓派端。

`ssh rikirobot@robot.local`

```
wnt-robot@WNTLab:~$ ssh rikirobot@robot.local
rikirobot@robot.local's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.38-v7+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

684 packages can be updated.
132 updates are security updates.

Last login: Wed Jan 24 17:40:08 2018 from 192.168.1.100
rikirobot@robot:~$
```

图 34.13 远程登陆到树莓派

(3) 登陆后, 运行以下命令, 可得到类似如图 34.14 所示内容, 这些内容表示的是各电机转速的编码输出。保持该窗口该命令运行。

`roslaunch rikirobot bringup.launch`

```
[INFO] [1516786995.519695]: Encoder RearLeft: 0
[INFO] [1516786995.524463]: Encoder FrontRight: 0
[INFO] [1516786995.529613]: Encoder RearRight: 0
[INFO] [1516786995.714708]: Encoder FrontLeft: 0
[INFO] [1516786995.719597]: Encoder RearLeft: 0
[INFO] [1516786995.724361]: Encoder FrontRight: 0
```

图 34.14 运行 bringup.launch

(4) 再打开一个 Terminal 窗口, 运行以下命令, 可得到如图 34.15 所示内容, 根据图 34.15 红色方框处指示, 读者可通过键盘上这几个键控制机器人朝各个方向运动。

`roslaunch teleop_twist_keyboard teleop_twist_keyboard.py`

```
Reading from the keyboard and Publishing to Twist!
-----
Moving around:
u   i   o
j   k   l
m   ,   .

For Holonomic mode (strafing), hold down the shift key:
-----
U   I   O
J   K   L
M   <   >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5      turn 1
```

图 34.15 运行 teleop_twist_keyboard.py

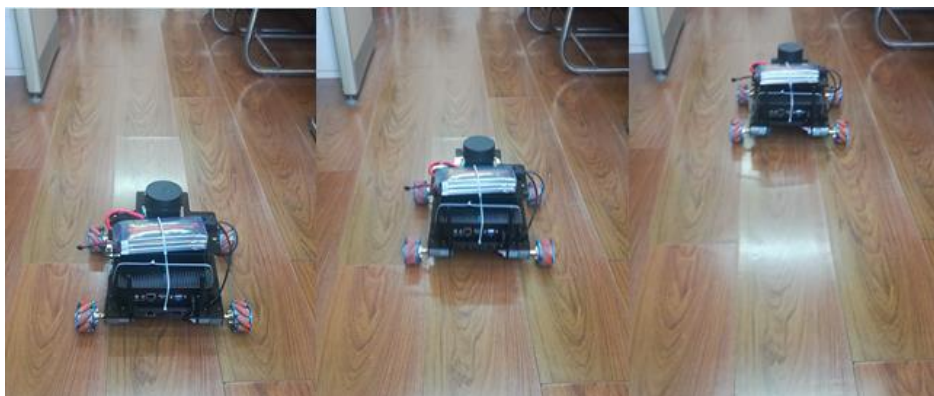


图 34.16 键盘控制机器人运动

4.4 语音控制机器人

本实验系统集成了语音识别功能包，该功能包采用了开源语音识别引擎 CMUSphinx 提供的声学模型，以及在其官网编译生成的离线词库。若需进一步了解 CMUSphinx，读者可参考 <https://cmusphinx.github.io/>。本节实验将测试语音控制机器人运动的功能，具体实验步骤如下：

(1) 将带麦克风的耳机插入 PC，点击虚拟机桌面右上角设置按钮，选择系统设置如图 34.17。在弹出的窗口中，点击“sound”，再选择“input”项，在界面中选择该耳机作为语音输入设备。

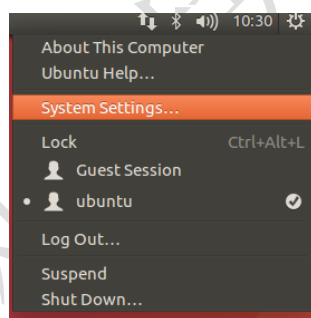


图 34.17 打开系统设置

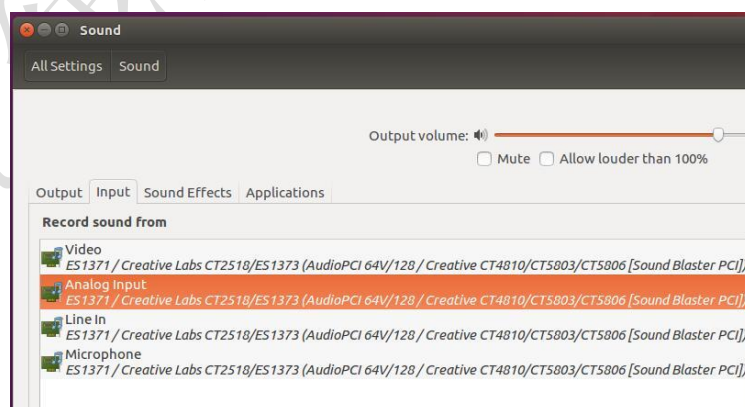


图 34.18 选择语音输入设备

- (2) 同 4.3 节步骤(2)至(3)，远程登录到树莓派，并启动 `bringup.launch`。
 (3) 再打开一个 Terminal 窗口，运行以下命令，可得到如图 34.19 所示内容

`roslaunch pocketsphinx voice_cmd.launch`


```

hones
INFO: dict2pid.c(131): Allocated 42672 bytes (41 KiB) for word-final triphones
INFO: dict2pid.c(195): Allocated 42672 bytes (41 KiB) for single-phone word trip
hones
INFO: ngram_model_arp.c(477): ngrams 1=7, 2=6, 3=5
INFO: ngram_model_arp.c(135): Reading unigrams
INFO: ngram_model_arp.c(516): 7 = #unigrams created
INFO: ngram_model_arp.c(195): Reading bigrams
INFO: ngram_model_arp.c(533): 6 = #bigrams created
INFO: ngram_model_arp.c(534): 2 = #prob2 entries
INFO: ngram_model_arp.c(542): 3 = #bo_wt2 entries
INFO: ngram_model_arp.c(292): Reading trigrams
INFO: ngram_model_arp.c(555): 5 = #trigrams created
INFO: ngram_model_arp.c(556): 2 = #prob3 entries
INFO: ngram_search_fwdtree.c(99): 5 unique initial diphones
INFO: ngram_search_fwdtree.c(147): 0 root, 0 non-root channels, 6 single-phone w
ords
INFO: ngram_search_fwdtree.c(186): Creating search tree
INFO: ngram_search_fwdtree.c(191): before: 0 root, 0 non-root channels, 6 single
-phone words
INFO: ngram_search_fwdtree.c(326): after: max nonroot chan increased to 134
INFO: ngram_search_fwdtree.c(338): after: 5 root, 6 non-root channels, 5 single-
phone words

```

图 34.19 运行 voice_cmd.launch

(4) 启动完成后, 读者可通过语音控制机器人运动。控制机器人前进、后退、左转、右转、停止的指令分别为 go、back、left、right、stop。(需注意, 读者需在安静的环境下测试语音识别控制功能, 由于语音模型本身和环境干扰等因素, 语音识别存在一定的误差。)

5 结果分析与讨论

本实验学习搭建基于 ROS 的机器人开发环境, 后续机器人相关实验均使用该环境。

6 注意事项

电池电量不足会导致部分功能无法正常实现, 读者可用万用表测量电池电压, 一般电压低于 11V 时需要及时充电。

实验三十五 机器人即时定位和地图构建

1 实验要求与目的

- 1 人独立完成
- 进一步理解 ROS 的开发过程
- 掌握利用机器人实现 SLAM 的过程

2 实验背景与原理

2.1 实验背景

(1) Robot Operating System(ROS)是一个机器人软件框架, 提供了操作系统应有的服务, 包括硬件抽象、底层设备控制、常用函数的实现、进程间消息传递、包管理等。

ROS 的主要目标是为机器人研究和开发提供代码复用的支持。ROS 是一个分布式的进程框架, 这些进程被封装在易于被分享和发布的程序包和功能包中。ROS 也支持一种类似于代码储存库的联合系统, 可实现工程的协作及发布。可以使一个工程的开发和实现从文件系统到用户接口完全独立决策(不受 ROS 限制)。

更多关于 ROS 请参考博客: <http://www.dengke.xyz/archives/category/robot>。

(2) 即时定位与地图构建(Simultaneous Localization And Mapping, SLAM)研究受到广泛关注, 问题可以描述为: 机器人在未知环境中从一个未知位置开始移动, 移动过程中根据位置估计和地图进行自身定位, 同时构建增量式地图, 从而实现机器人的自主定位和导航。

(3) 惯性测量单元(Inertial Measurement Unit, IMU)是测量机器人三轴姿态角和加速度的装置。本实验机器人所用 IMU 包括三轴加速度计、三轴陀螺仪、三轴电子罗盘, 分别用于测量加速度、方向以及磁感应强度, 并以此解算出机器人的姿态。IMU 在机器人 SLAM 中有着重要作用。

2.2 实验原理

本实验基于实验 34 搭建好的开发环境。实验中首先需要校准机器人的 IMU、角速度及线速度参数, 再使用 gmapping 包构建室内地图, 最后不断修正地图并保存地图以待备用。实验流程如图 35.1 所示

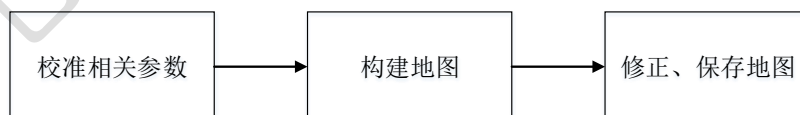


图 35.1 实验流程

(2) 本实验系统中, 机器人相关实验均通过 PC 端 Ubuntu 虚拟机控制移动式机器人进行实验, 二者通过 WiFi 通信, 原理如图 35.2 所示。

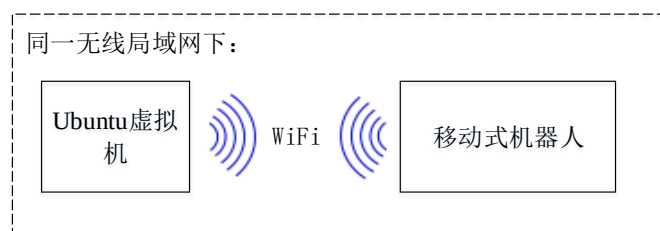


图 35.2 机器人控制原理

3 实验环境与资源

(1) 环境资源:

移动式机器人套件（树莓派 Raspberry Pi3、TF 卡、电机驱动板、激光雷达、电源、车体等）、机器人开发环境。

(2) 使用说明

机器人开发环境为实验 34 中搭建好的环境，读者进行本实验前需先完成实验 34。

4 实验步骤与演示

4.1 实验准备

实验准备主要工作为校准机器人相关参数：IMU、角速度、线速度。以上参数均存在一定误差，只有将这些误差校准到可接受范围内，地图构建才能有较好的结果。

同前所述，打开 wnt-robot 虚拟机和机器人电源开关，并确保二者接入同一无线局域网。具体校准步骤如 4.1.1 至 4.1.3 节所述，在此之前需了解 bringup.launch 文件。该文件为机器人的启动文件，在实验 34 第 4.3 节中就用到了该文件。

在虚拟机中打开 Terminal(快捷键“Ctrl+Alt+T”)，执行下述命令可进入 bringup.launch 文件的编辑界面。如图 35.3 红色横线出所示，①②③三处内容分别是校准 IMU、角速度、线速度时需要修改的参数。（需注意，读者需记下编辑该文件的方法，以待备用。）

`sudo gedit /catkin_ws/src/rikirobot_project/rikirobot/launch/bringup.launch`

```
<launch>
<!--serial communication between arduino and pc via usb /-->
<node name="arduino_serial_node" pkg="roscserial_python" type="serial_node.py" output=
  <param name="port" value="/dev/rikistm32base" />
  <param name="baud" value="57600" />
</node>

<node pkg="ros_arduino_imu" type="raw_imu_bridge_node" name="raw_imu_bridge" output="s
  <rosparm>
    imu/gyroscope bias: {x: -0.013545, y: 0.056887, z: 0.016997} ①
    imu/accelerometer bias: {x: -0.422578, y: 0.053516, z: -0.310391 }
  </rosparm>
  <param name="imu/perform_calibration" value="false" />
</node>

<node pkg="imu_filter_madgwick" type="imu_filter_node" name="imu_filter_madgwick" out
  <param name="use magnetic field msg" value="false" />
  <param name="fixed_frame" value="odom" />
</node>

<node name="riki_base_node" pkg="rikirobot" type="riki_base_node">
  <param name="angular_scale" type="double" value="1.0" /> ②
  <param name="linear_scale" type="double" value="1.0" /> ③
</node>
</launch>
```

图 35.3 bringup.launch 文件内容

4.1.1 校正机器人 IMU

(1) 校准 IMU 前，将 bringup.launch 文件位置①处 z 值修改为：0.0，并保存。

(2) 在虚拟机中打开两个 Terminal 窗口(快捷键“Ctrl+Alt+T”)，在两个窗口中均执行以下命令，并输入密码 123456，即实现远程登录到机器人树莓派端。完成后保持两个窗口运行。

`ssh rikirobot@robot.local`

```
wnt-robot@WNTLab:~$ ssh rikirobot@robot.local
rikirobot@robot.local's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.38-v7+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

684 packages can be updated.
132 updates are security updates.

Last login: Wed Jan 24 17:40:08 2018 from 192.168.1.100
rikirobot@robot:~$
```

图 35.4 远程登录到树莓派

(3) 在其中一个 Terminal 窗口中运行以下命令, 可得到类似如图 35.5 所示内容, 这些内容表示的是各电机转速的编码输出。保持该窗口该命令运行。

roslaunch rikirobot bringup.launch

```
[INFO] [1516786995.519695]: Encoder RearLeft: 0
[INFO] [1516786995.524463]: Encoder FrontRight: 0
[INFO] [1516786995.529613]: Encoder RearRight: 0
[INFO] [1516786995.714708]: Encoder FrontLeft: 0
[INFO] [1516786995.719597]: Encoder RearLeft: 0
[INFO] [1516786995.724361]: Encoder FrontRight: 0
```

图 35.5 运行 bringup.launch

(4) 在另一个 Terminal 窗口中运行以下命令, 可得到如图 35.6 所示 IMU 数据。

rostopic echo /imu/data

```
z: 0.013355633947
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
  x: -0.593047
  y: 0.258984
  z: 10.076016
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
header:
  seq: 398
  stamp:
    secs: 1500526643
    nsecs: 978554958
  frame_id: imu_link
orientation:
  x: 0.0236128152663
  y: 0.00875377229973
  z: 0.293314598656
  w: 0.955684232705
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
  x: 0.0985349426103
  y: -0.0848122678156
  z: 0.0169979892671
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
  x: 0.0710155
  y: 0.2199215
  z: 9.9588285
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
```

图 35.6 IMU 数据

- (5) 校准 IMU 的目标是将图 35.6 红色方框处的 z 值降低到 0.01 以下。方法如下:
- 在两个 Terminal 窗口中都按下快捷键“ctrl+c”中断当前命令;
 - 将图 35.3 中①处的值修改为图 35.6 中的 z 值(保留小数点后 6 位), 并保存;
 - 重新执行第(3)、(4)步, 并观察此时图 35.6 中的 z 值;

d. 若 z 值尚未达到要求, 从步骤 a 开始重新校准, 直至达到要求。

4.1.2 校正机器人角速度

(1)、(2): 同 4.1.1 节步骤(2)、(3)。

(3) 在另一个 Terminal 窗口中运行以下命令, 可得到如图 35.7 所示结果。

`roslaunch rikirobot_nav calibrate_angular.py`

```
rikirobot@robot:~$ roslaunch rikirobot_nav calibrate_angular.py
[INFO] [1500526839.637347]: Bring up rqt_reconfigure to control the test.
```

图 35.7 校准角速度

(4) 再打开一个 Terminal 窗口 (不用登录到树莓派), 执行以下命令, 得到可视化界面如图 35.8 所示。

`roslaunch rqt_reconfigure rqt_reconfigure`

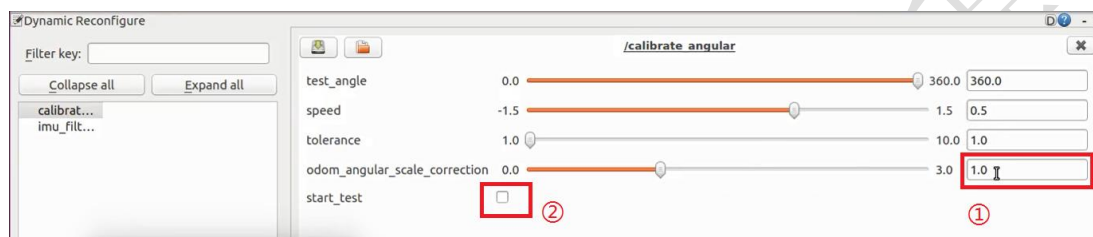


图 35.8 校准角速度

(5) 校准角速度的目标是使得机器人能够旋转较为准确的 360° 。方法如下:

- 点击图 35.8 中②处按钮, 机器人会开始原地旋转;
- 等待机器人旋转完毕, 观察它旋转的角度;
- 如果旋转角度大于 360° , 则将图 35.8 中①处的值改小, 反之改大 (建议以 0.1 为步进改动);
- 反复执行步骤 a 至 c, 直至达到要求;
- 达到要求后, 将图 3 中②处的值修改为图 35.8 中①处的值, 并保存。



图 35.9 校准机器人角速度

4.1.3 校正机器人线速度

校准线速度与校准角速度步骤基本一致, 不同之处在于:

- 第(3)步中执行的命令改为:

`roslaunch rikirobot_nav calibrate_linear.py`

- 第(4)步中的结果改为如图 35.10 所示:

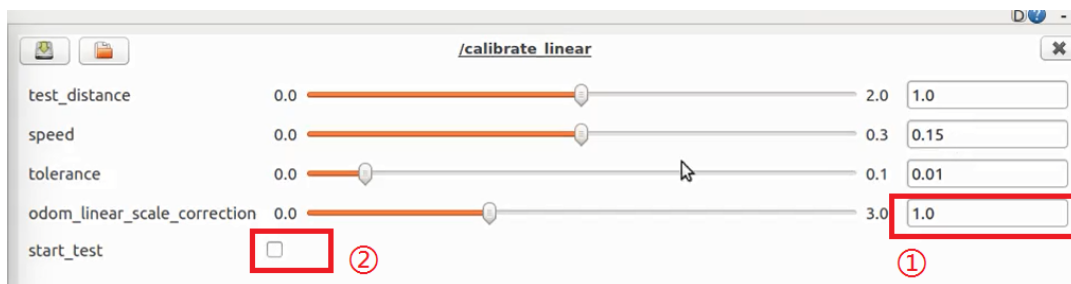


图 35.10 校准线速度

- 校准线速度的目标是使得机器人能够较为准确地直线行走 1m 距离。
- 达到校准目标后, 将图 35.3 中③处的值修改为图 35.10 中①处的值。

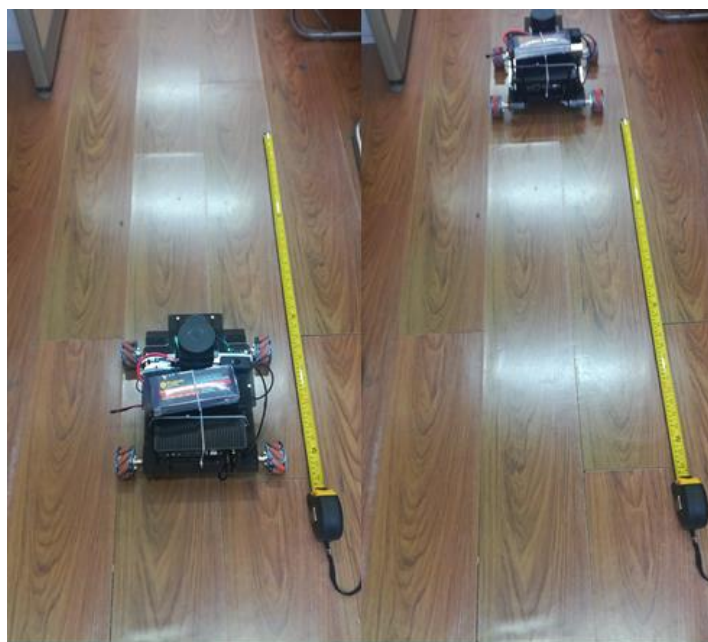


图 35.11 校准机器人线速度

4.2 基于 gmapping 包的地图构建过程

gmapping 是一个比较完善的地图构建开源包, 它使用激光雷达和里程计的数据来生成二维地图。

使用 gmapping 包构建地图需要在实现控制机器人运动的基础上进行, 因此构建地图前先完成实验 34 第 4.3 节内容, 并且保持两个 Terminal 中命令保持运行状态。后续步骤如下:

(1) 再打开一个 Terminal 窗口, 同样通过 ssh 方式远程登录到树莓派, 再执行以下命令运行激光雷达的启动文件, 直到出现如图 35.12 所示字样, 激光雷达启动成功。

`roslaunch rikirobot lidar_slam.launch`

```
[ INFO] [1500522665.025542473]: Sim period is set to 0.20
[ INFO] [1500522665.921997358]: Recovery behavior will clear layer obstacles
[ INFO] [1500522666.129248633]: Recovery behavior will clear layer obstacles
[ INFO] [1500522666.383381175]: odom received!
```

图 35.12 运行激光雷达启动文件

(2) 再打开一个 Terminal 窗口 (不用登录到树莓派), 执行以下命令启动 rviz。rviz 是 ROS 一款 3D 可视化工具, 利用它可以显示机器人构建的地图。

`roslaunch rviz rviz`

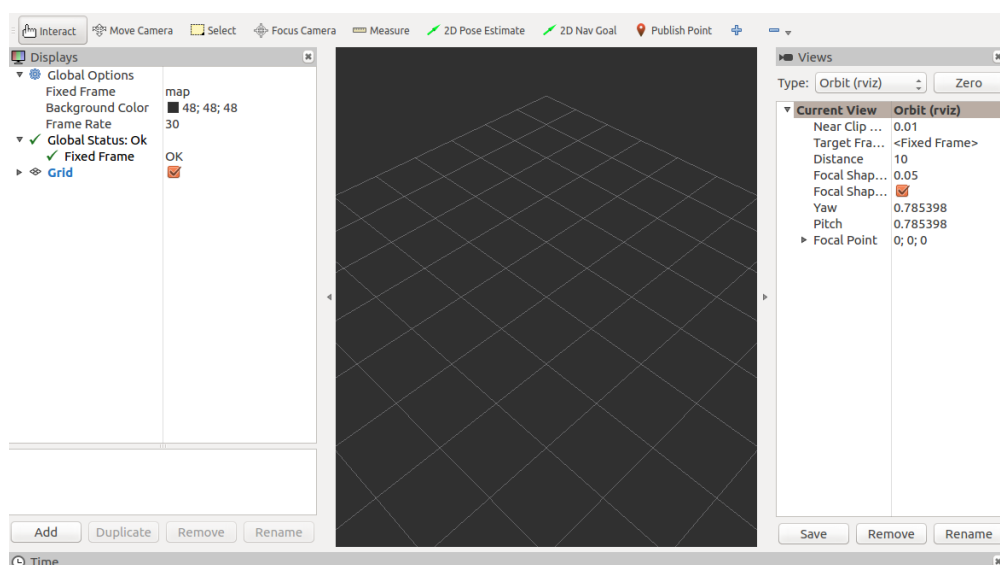


图 35.13 rviz 界面

(3) 点击左上角“File-Open Config”按钮，在弹出的窗口中选择/home/wnt-robot/catkin_ws/rikirobot_project/rikirobot/rviz/slam.rviz 文件，并打开。

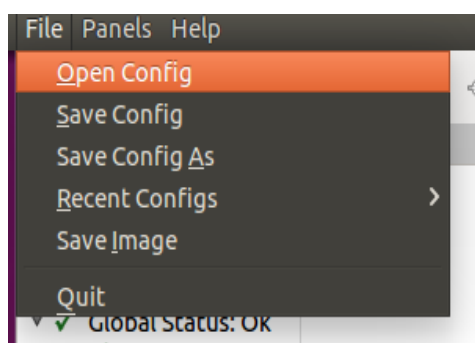


图 35.14 点击“File-Open Config”按钮

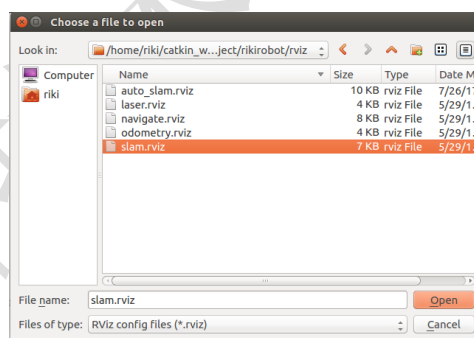


图 35.15 打开 slam.rviz 文件

(4) 打开 slam.rviz 文件后，机器人开始创建地图。如图 35.16 所示，红色边缘部分是激光雷达当前扫描到的障碍物，黑色部分是已经确定的地图边缘，灰色块状部分则是有效的地图区域。

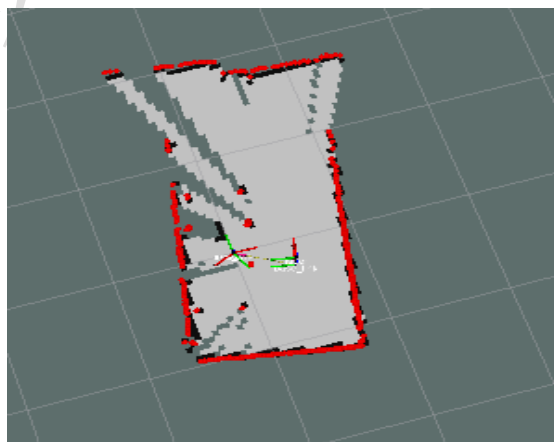


图 35.16 开始构建地图

(5) 在实验 34 第 4.3 节打开的键盘控制窗口中，控制机器人在房间内运动，使得机器人能够扫描到房间内全部区域，从而构建出较为完整的室内地图，如图 35.17、35.18 所示。

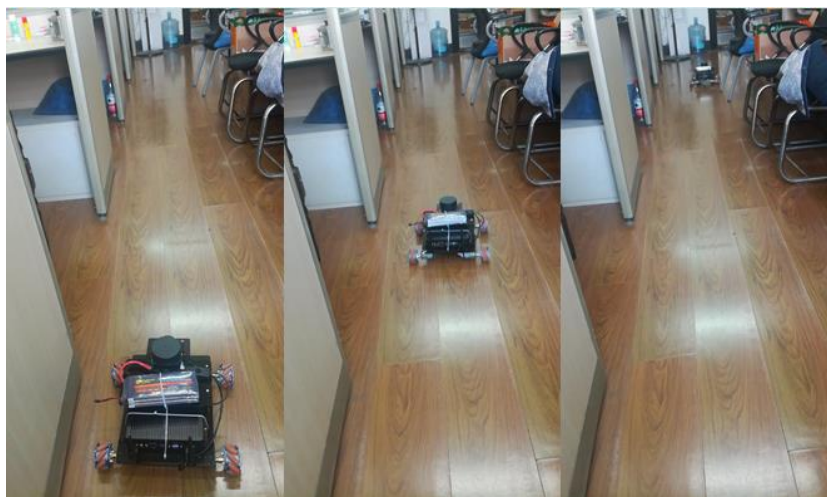


图 35.17 控制机器人运动构建地图



图 35.18 构建地图

4.3 修正地图与保存地图

根据上述步骤创建的地图可以用于导航等用途,但可能会存在一些问题使得导航效果不理想,此时便需要修正地图。下面列出最常见的三个问题,在图 35.18 中均存在,详见图 35.18 红色标记处:

- a. 地图中部分区域黑色边界不全;
- b. 部分灰色区域存在空白;
- c. 部分黑色边界线粗乱。

对于问题 a、b,修正方法是控制机器人走遍房间中的每个区域,并且在每个区域的停留时间稍长一点;对于问题 c,当机器人停留在某一区域时不要旋转机器人,因为激光雷达本身就在不断旋转,少量区域出现该问题时可以忽略,太多时则需要重新建图。

修正完毕得到完善的地图后,不要关闭 rviz 工具,接下来可保存地图以待备用,步骤如下:

- (1) 再打开一个 Terminal 窗口,执行下述命令进入地图保存目录。

```
cd catkin_ws/src/rikirobot_project/rikirobot/maps
```

- (2) 执行下述命令,运行保存地图脚本文件,结果如图 35.19 所示。

./map.sh

```
[ INFO] [1500522998.329660714]: Waiting for the map
[ INFO] [1500522998.653439573]: Received a 1984 X 1984 map @ 0.050 m/pix
[ INFO] [1500522998.653698137]: Writing map occupancy data to house.pgm
[ INFO] [1500522999.058150782]: Writing map occupancy data to house.yaml
[ INFO] [1500522999.058841156]: Done
```

图 35.19 保存地图

(3) 执行下述命令, 查看地图文件。其中.pgm 和.yaml 均为保存后的地图文件, 后续实验可直接使用该文件。

ll -h

5 结果分析与讨论

本实验利用机器人完成了 SLAM 功能, 构建并保存了室内地图。

6 注意事项

本实验中部分命令是通过 ssh 方式登录到树莓派后再运行的, 另一部分是直接在虚拟机中运行, 请读者注意区分。

实验三十六 基于 WiFi 的无人船控制实验

1 实验目的和要求

- 理解 WiFi 的无线通信原理;
- 掌握使用 WiFi 通信模块远程控制电机设备。

2 实验背景与原理

2.1 实验背景

WiFi、蓝牙的背景知识请参考教材第3章;
本实验涉及 WiFi 模块的使用, 请先完成实验十一。

2.2 实验原理

(1) 整体控制过程

通过 ESP8266 WiFi 模块将移动手机端 APP 上的控制指令发送至 Arduino 开发板 (附于无人船上), Arduino 开发板根据接收到的指令驱动电机与舵机运转。其中, 电机控制无人船前进动力, 舵机控制无人船前进方向。控制原理如图 36.1 所示。

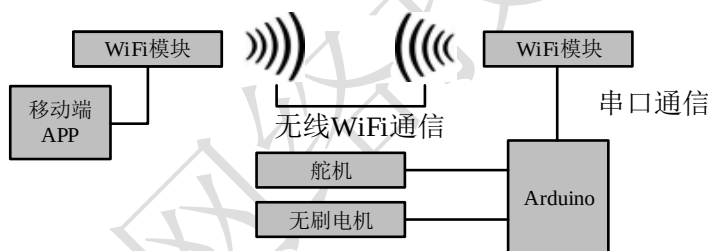


图 36.1 WiFi 控制无人船原理框图

(2) 舵机控制原理

舵机模块总共有 3 根引线: 电源线、地线和信号线。Arduino 开发板通过引脚 9 输出周期为 20ms 的脉冲波, 舵机接收到不同占空比的脉冲波会转动到相应角度。不同型号舵机的脉冲波占空比与转动角度的对应关系可能会不同, 故以实际测量为准。

(3) 无刷电机控制原理

无刷电机需要配备相应的驱动器 (也称电子调速器或简称电调)。其控制方法和舵机类似, 不同占空比的输入脉冲波对应着不同转速。具体对应关系以实际测量为准。

3 实验环境与资源

(1) 环境资源

PC 1 台、Arduino Mega 2560 单片机模块 1 个、ESP8266 WiFi 模块 1 个、杜邦线若干、12V 电池 1 个、Android 系统手机 1 部、无人船模型 1 架、Arduino IDE。

(2) 使用说明

- 各类硬件如图 36.2 至图 36.6 所示, 详细使用方法见实验步骤;
- 本实验器材中包含螺旋桨, 实验过程中务必注意安全, 人身与螺旋桨保持安全距

离, 实验完毕应第一时间切断电源;

- 本实验使用 ESP8266 模块作为 WiFi 通信模块, 相关操作请参考实验十一;
- 本实验提供移动端 APP 相关源代码及 apk 安装包, 供读者学习与二次开发, 详见实验资源包 WNTLab/experiments/exp36。



图 36.2 Arduino MEGA 2560 单片机模块

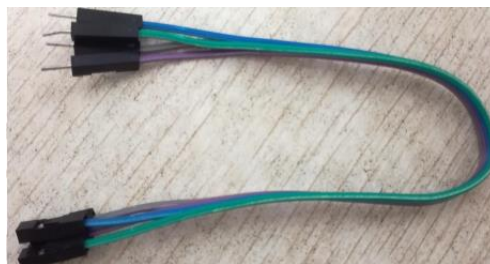


图 36.3 杜邦线



图 36.4 12V 电池

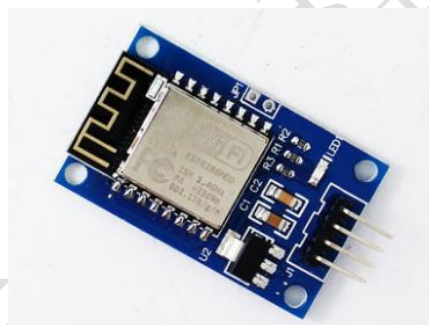


图 36.5 WiFi 模块

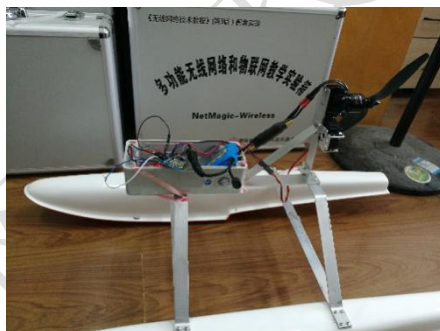


图 36.6 无人船模型

4 实验步骤与演示

4.1 开发板程序烧录

在实验箱中取出 Arduino Mega 2560 主控板 1 个, 烧录本实验提供的主控板代码。参考实验十六 4.1 节 Arduino 代码烧录过程, 打开 Arduino IDE, 编译并烧录代码到 Arduino Mega 2560 主控板中。如图 36.7 所示。



图 36.7 主控板代码烧录

4.2 下位机硬件连接

(1) 根据图 36.8 所示, 使用公对母杜邦线连接 ESP8266 WiFi 模块、Arduino Mega 2560 单片机模块、电机与舵机, 并仔细检查线路连接情况。

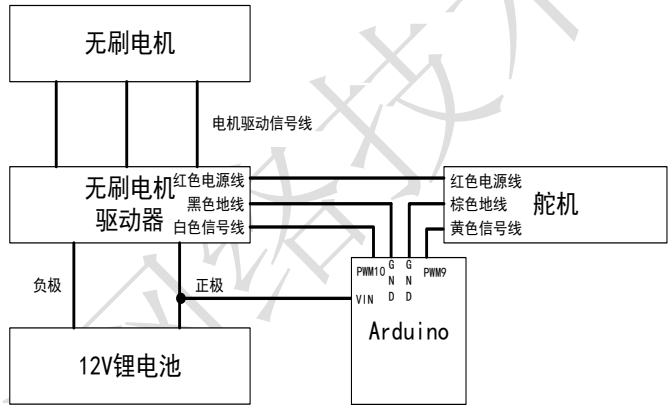


图 36.8 各模块连接线路图

(2) 将电池与各电路模块连接后固定至无人船模型上, 由于电调(电子调速器)连线较复杂, 读者可参考以下电路实际连接进行操作, 如图 36.9 所示(注意: 接通电源前应保持与螺旋桨之间一定安全距离)。

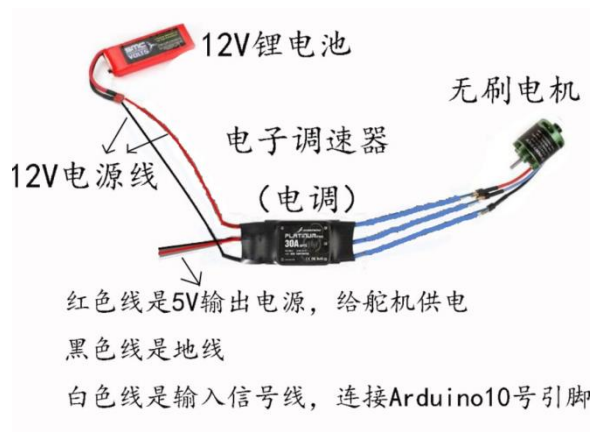


图 36.9 电调实际线路连接图

4.2 移动端 APP 安装

根据本实验提供的 apk 安装包, 进行移动端软件安装。读者需自备一部安卓智能手机。在手机中下载配套的 APP 安装包, 点击安装并等待安装完成, 如图 36.10 和图 36.11 所示 (**需注意, 该APP 要求手机操作系统为Android5.0 及以上版本。**)

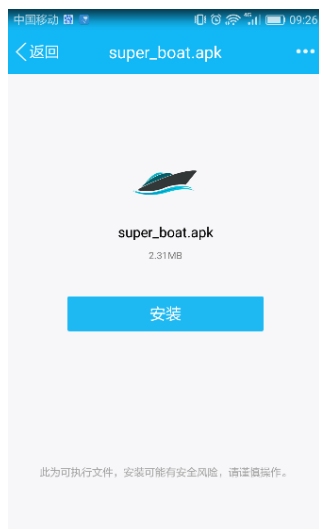


图 36.10 开始安装

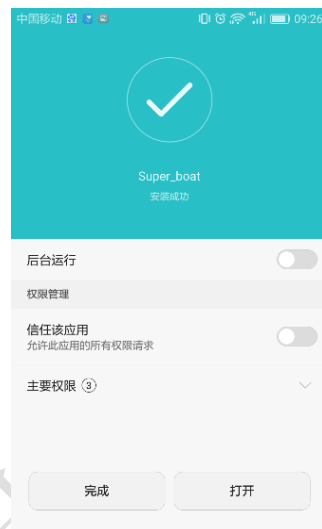


图 36.11 安装完成

4.3 水面环境下系统实测

(1) 接通锂电池电源, 电路发出“滴滴滴”就绪声音后, 并将无人船模型平稳放置于水面上, 如图 36.12 所示 (**需注意: 接通电源前务必与螺旋桨保持安全距离**);



图 36.12 平稳放置无人船模型

(2) 连接名为“Superboat”的 WiFi 信号, 打开移动端 APP。点击“开始”按钮后跳出对话框, 输入 WiFi 模块的 IP 地址和端口号(默认 IP 为“192.168.4.1”, 端口号为“5000”)建立连接, 如图 36.13 和 36.14 所示;



图 36.13 移动端 APP 界面



图 36.14 连接对话框

(3) 进入主控界面后先点击“锁定”复选框解锁无人船。按住“前进”控制无人船前进, 松开则停止。按住“左右”可让无人船左右转向, 松开则螺旋桨回正。滑动滑杆可控制速度。勾选“锁定”复选框则无人船保持当前状态, 不再接收控制信号(解锁信号除外)。控制界面如图 36.15 所示;



图 36.15 主控界面

(4) 由于螺旋桨转速较高, 实验过程中存在安全风险, 接通电源时学生务必与螺旋桨保持安全距离, 使用完后请及时切断电源。

5 移动端 APP 源码分析

本实验将移动端 APP 功能模块分为三个部分, 设计为 Control、Intro 和 MainActivity 类, 分别实现无人船远程控制、软件操作帮助及建立 WiFi 通信功能。功能模块如图 36.16 所示。

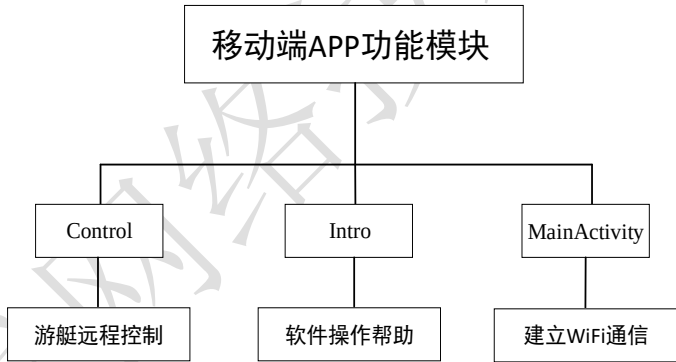


图 36.16 功能模块

其中 Intro 仅实现了文字显示功能, 代码较为简单, 不再深入说明。

(1) MainActivity 类中建立 WiFi 通信核心代码如下:

```
class ConnectSeverThread extends Thread
{
    public void run()
    {
        while(ConnectFlage)
        {
            try
            {
                ipAddress = InetAddress.getByName(Ipstring);
                socket = new Socket(ipAddress, portint);
                ConnectFlage = false;
                tcpClientCountDownTimer.cancel();
            }
            catch (Exception e)
            {
                // Exception handling
            }
        }
    }
}
```

```

        runOnUiThread(new Runnable()
        {
            public void run()
            {
                progressBar21.setVisibility(-1);//关闭滚动条
                AlertDialog21.cancel();//关闭提示框
            }
        });
        inputStream = socket.getInputStream();//获取输入流

        ReadDataFlage = true;

        ReadDataThread readDataThread = new ReadDataThread();
        readDataThread.start();

        intentcontrol.setClass(MainActivity.this, Control.class);
        MainActivity.this.startActivity(intentcontrol);
    }
    catch (IOException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}
}
}
}

```

以上代码通过读取对话框中的 IP 地址及端口号, 创建 socket 通信线程, 请求服务端 WiFi 连接。若连接成功建立, 则将控制流程切换至 Control 类。

(2) 控制界面中各按键对应的事件响应函数如下:

/** 前进按钮**/

```

private OnTouchListener imageButton31Touch = new OnTouchListener() {
public boolean onTouch(View v, MotionEvent event) {
    if (event.getAction()==MotionEvent.ACTION_DOWN) {
        sendbyte[0]=(byte)101;//无人船接收到 101 则前进
        sendflag=true;
        imageButton31.setImageResource(R.drawable.qianjindown);
        vibrator.vibrate(new long[]{0,40}, -1);//震动
    }
    if (event.getAction()==MotionEvent.ACTION_UP) {
        sendbyte[0]=(byte)102;//无人船接收到 102 则停止
        sendflag=true;
        imageButton31.setImageResource(R.drawable.qianjin);//改变背景
    }
}
}

```

```

        }
        return false;
    }
};

/** 左转按钮**/
private onTouchListener imageButton34Touch = new onTouchListener() {
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN) {
            sendbyte[0]=(byte)103;//无人船接收到 103 则左转
            sendflag=true;
            imageButton34.setImageResource(R.drawable.zuozhuandown);
            vibrator.vibrate(new long[]{0,40}, -1); //震动
        }
        if (event.getAction()==MotionEvent.ACTION_UP) {
            sendbyte[0]=(byte)105;
            sendflag=true;
            imageButton34.setImageResource(R.drawable.zuozhuan);//改变背景
        }
        return false;
    }
};

/** 右转按钮*****/
private onTouchListener imageButton33Touch = new onTouchListener() {
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction()==MotionEvent.ACTION_DOWN) {
            sendbyte[0]=(byte)104;
            sendflag=true;
            imageButton33.setImageResource(R.drawable.youzhuandown);
            vibrator.vibrate(new long[]{0,40}, -1); //震动
        }
        if (event.getAction()==MotionEvent.ACTION_UP) {
            sendbyte[0]=(byte)105;//无人船接收到 105 则右转
            sendflag=true;
            imageButton33.setImageResource(R.drawable.youzhuan);//改变背景
        }
        return false;
    }
};

```

按下或松开各个按键后, 各按键的事件处理函数会把 `sendbyte` 这个待发送变量设置为各个控制事件对应的数值, 然后把 `sendflag` 变量设为 `true`, 告诉 `SendMsgThread` 线程有数据要发送。

(3) `SendMsgThread` 线程负责发送数据, 其核心代码如下:

`class SendMsgThread extends Thread{`//这是一个负责发送数据的线程

```
public void run(){
    while(true){
        if(sendflag){//循环检测是否有数据要发送, 若有则发送数据
            netSend(sendbyte);//发送数据
            sendflag=false;//发送完设置记号, 避免重复发送
            try {
                Thread.sleep(100);//延时 100ms
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

当控制界面的按钮按下或松开时, sendflag 变量会被设置为 true, SendMsgThread 线程检测到后便会向无人船发送控制信号。

6 结果分析与讨论

本实验介绍了基于移动端 APP 和 WiFi 模块进行远程遥控无人船, 该实验可通过 WiFi 通信协议远程控制无人船运动。本实验着重说明了无人船各个功能模块间的连接以及上位机设计, 硬件驱动部分请读者自行研究相关源码。

目前 WiFi 设备较普及, 开发难度较低, 适于多数的无线通信场景。但由于传输距离较短, 室外环境下的实际应用还受到一定限制, 故在后续实验中, 我们引入低功耗广域网物联网 LoRa, 扩大了无人船控制距离, 拓展了实际应用范围。

(实验过程中注意保障人身安全!!)

实验三十七 基于 LoRa 的远距离 无人船控制实验

1 实验目的和要求

- 了解 LoRa 技术的通信原理；
- 熟悉 LoRa 通信模块实现远程控制无人船的过程；
- 利于基于 C# 的 PC 端串口控制软件的基本步骤。

2 实验背景与原理

2.1 实验背景

在实验三十六中, 已经介绍了基于移动端 APP 和 WiFi 通信对无人船进行遥控, 通过 WiFi 通信协议控制无人船运动。本实验则采用低功耗广域网中的 LoRa 技术代替 WiFi 实现远距离控制, 对无人船的可控距离更远, 更符合在室外水面上的应用条件。LoRa 技术的理论内容请参考教材第 9 章;

LoRa 模块的使用请参考实验三十三。

2.2 实验原理

(1) 整体控制过程

将 PC 端串口软件上的控制指令通过 LoRa 模块, 远程发送至无人船上的 Arduino 开发板, 开发板根据接收到的指令驱动电机与舵机运转, 并回传 GPS 模块采集的经纬度以及海拔高度信息。其中, 电机控制无人船前进动力, 舵机控制无人船前进方向。控制原理如图 37.1 所示。其中舵机和无刷电机控制原理已在实验三十六详细说明, 本实验不再赘述。

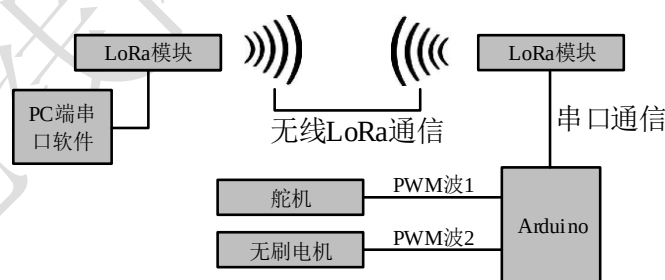


图 37.1 本实验控制原理框图

本实验分别设计了 LoRa 信号发送板(Arduino-LoRa 集成模块)、LoRa 信号接收板(Arduino-LoRa 集成模块)以及电机驱动板。通过发送板将 PC 端串口上的控制指令发送至无人船的接收板, 接收板根据接收到的指令通过串口转发至电机驱动板, 电机驱动板驱动电机和舵机运转。详细设计请参考下文第 4.2 节的电路连接图。

3 实验环境与资源

(1) 环境资源

PC 1 台、Arduino Mega 2560 单片机模块 1 个、Arduino-LoRa 集成模块 2 个、杜邦线若

干、12V 电池 1 个、USB A 型转 B 型线、无人船模型 1 台、Arduino IDE。

(2) 使用说明

- 各类硬件如图 37.2 至图 37.7 所示，详细的使用方法见实验步骤；
- 本实验器材中包含螺旋桨，实验过程中务必注意安全，与螺旋桨保持安全距离，实验完毕应及时切断电源；
- 本实验使用 Dragino LoRa Shield 作为 LoRa 通信模块，相关操作请参考实验三十三；
- 本实验提供接收板和驱动板相关源代码，供读者学习与二次开发，详见实验资源包 WNTLab/experiments/exp37。



图 37.2 Arduino MEGA 2560 单片机模块

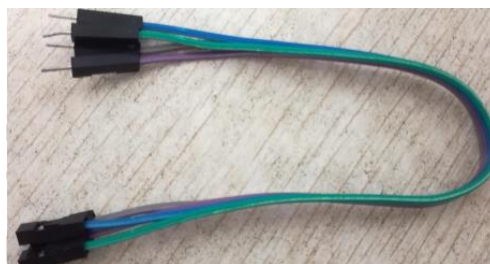


图 37.3 杜邦线



图 37.4 12V 电池



图 37.5 LoRa 模块



图 37.6 USB A 型转 B 型线

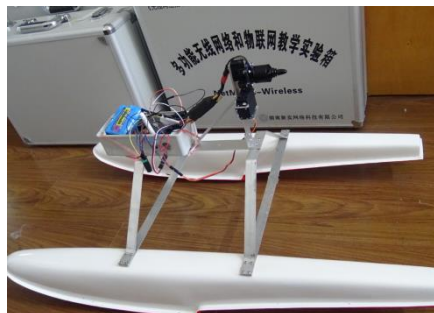


图 37.7 无人船模型

4 实验步骤与演示

4.1 开发板程序烧录

在实验箱中取出 Arduino Mega 2560 单片机模块 1 个、Arduino-LoRa 集成模块 2 个，分别烧录本实验提供的发送板、接收板、驱动板代码。参考实验十六 4.1 节 Arduino 代码烧录过程，打开 Arduino IDE，编译本实验代码，并烧录到 Arduino Mega 2560 单片机中。如图 37.8 至图 37.10 所示。

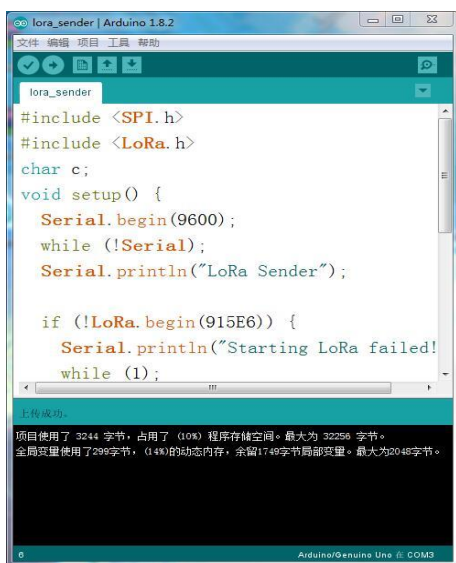


图 37.8 发送板代码烧录

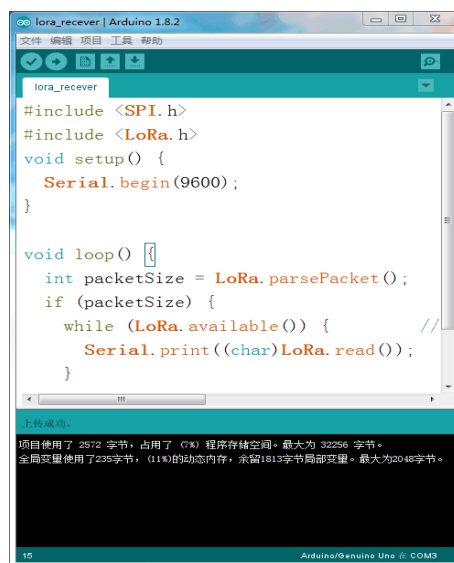


图 37.9 接收板代码烧录

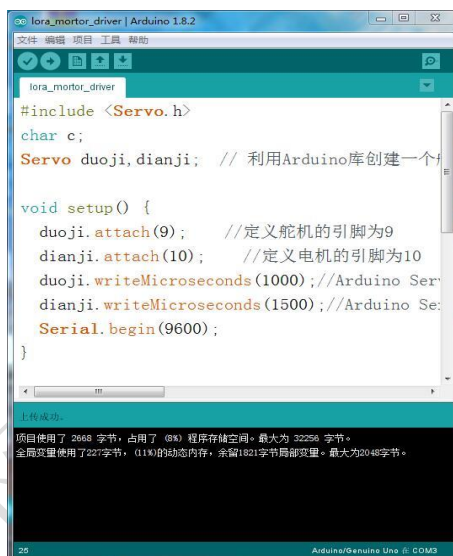


图 37.10 驱动板代码烧录

烧录程序完成后, 断开 Arduino Mega 2560 单片机模块与 PC 的连接。

4.3 开发板硬件连接

根据图 37.11 所示, 连接 LoRa 信号发送板与 PC 端, 连接 LoRa 信号接收板与电机驱动板, 连接电机驱动板与无刷电机、舵机, 并仔细检查电路连接情况。

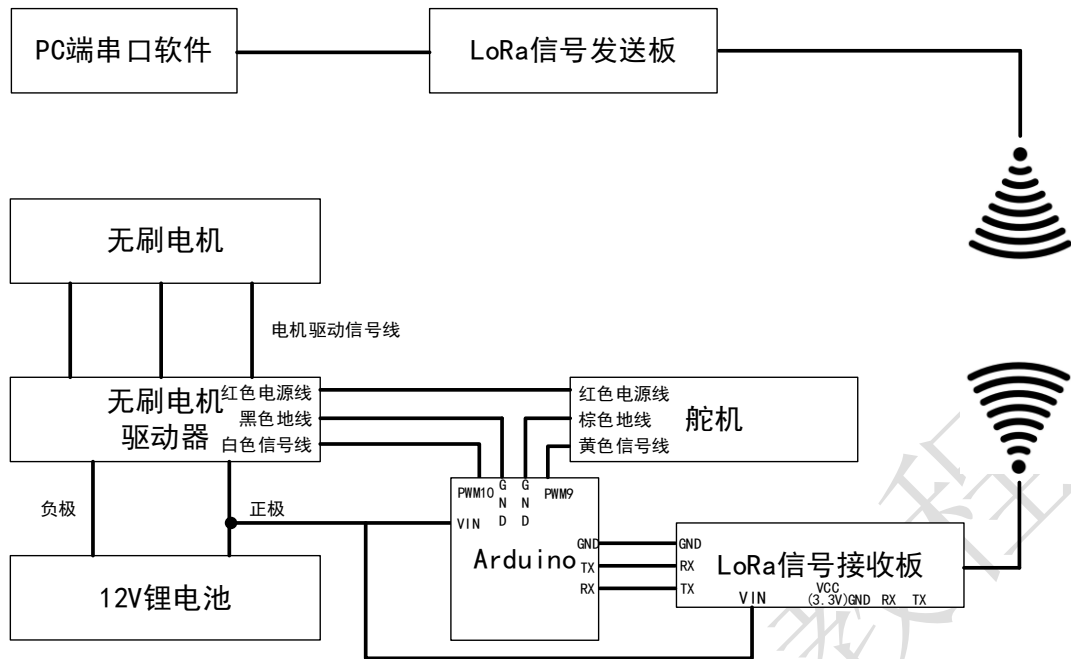


图 37.11 线路连接图

4.4 水面环境下系统实测

本次实测在室外水面环境中进行, 其中 PC 端位于教学楼处, 无人船位于周边池塘的水面上。通过 PC 端串口软件远距离控制 100 米外的无人船移动、转向以及获取 LoRa 信号接收板上 GPS 所获取的经度、纬度、海拔高度信息。实验场地如下图 37.13 所示。



图 37.13 室外实测

- (1) 接通锂电池电源并将无人船模型平稳放置水面上, 如图 37.14 所示。



图 37.14 平稳放置无人船模型

(2) 将 PC 端通过 USB A 型转 B 型线与 LoRa 信号发送板相连, 如图 37.15 所示。



图 37.15 信号发送板通过串口连接 PC

(3) 打开 PC 端控制软件, 按照软件上的教程进行操作, 如图 37.16 和图 37.17 所示。

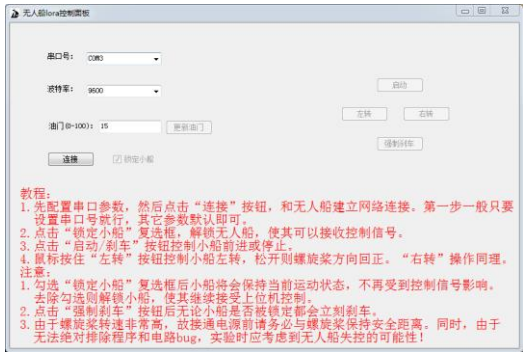


图 37.16 软件初始界面

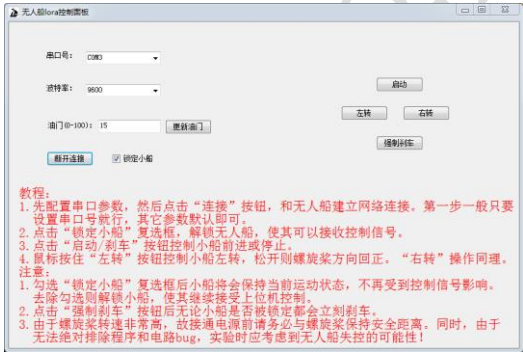


图 37.17 和无人船建立连接后界面

(4) 由于螺旋桨转速较高, 使用过程中人体务必保持安全距离, 使用完后尽快切断电源。

5 结果分析与讨论

本实验基于 VS studio IDE 开发环境用 C#语言编写了 PC 端串口控制软件, 通过串口控制 LoRa 信号发送模块发送相应的信号, 无人船上的 LoRa 接收板接到信号后通过串口转发信号至电机驱动板并控制无刷电机和舵机运转。该实验充分利用 LoRa 长距离通信的特点, 扩展了无人船在户外的控制范围, 提升了实验的实际应用价值。实验者可基于该平台在船上进一步安装传感器等将其打造成一个水上平台。本实验各部分源代码均开放, 读者可自行修改并完善代码, 扩展无人船的控制效果。