

西南林业大学
本科毕业（设计）论文
（二〇一四届）

网络用户管理模块的设计与
题 目：集成

分院系部：计算机与信息学院

专 业：计算机科学与技术专业

姓 名：温俊瑞

导师姓名：孙永科

导师职称：讲师

二〇一四年五月

网络用户管理模块的设计与集成

温俊瑞

(西南林业大学 计算机与信息学院, 云南 昆明 650224)

摘 要：如今，一走进咖啡厅、休闲娱乐场所、银行展会、美容购物等地方或者入住酒店时，几乎每个人都会先问一句：有没有免费的 WiFi 啊？WiFi 普及的速度着实令人惊叹，已经成为不少人主要的移动上网方式。国外咖啡厅，购物商场，酒店等基本都覆盖了 wifi 热点，顾客利用闲暇时间可以免费上网，同时商家又可以借助弹出的认证界面投送广告，带来商业利益。wifi 热点认证收到国外很多商家的青睐。就连不少航空公司的航班上都开始提供起 WiFi 服务，让乘客可以利用自带的电脑等电子产品联网，在空中享受发邮件、传递数据等。

目前国内 WIFI 热点认证方案主要还是通过手机短信认证、人工操作认证、采用固定密码或者直接不认证。

手机短信认证方式是这样的，用户接入 WIFI 热点后手机会弹出认证界面，输入手机号码后点击获取密码，输入短信中显示的密码即可上网。这样的认证方式存在一些不足，比如用户自带电脑不用手机获取密码就无法上网，而且不能有效的防止蹭网现象的发生，如果用一些软件批量获取密码上网的话路由器将会不堪重负。人工操作可以防止软件恶意批量新建帐号，但是操作起来耗费人力物力，操作人员也可能因为误操作导致网络出问题。采用固定密码或者直接不认证的方式就相当于完全公开网络，同样不能防止蹭网。

经过市场调查分析，国内 WIFI 热点认证方案或配置麻烦或不够安全稳定，基本都是基于 windows 操作系统下来进行控制，人工操作环节比较多从而导致费时费力。本文介绍了一套基于 Raspberry Pi^[8] 的 WIFI 热点认证方案，通过集成于 Raspberry Pi 实现刷卡打印的功能，为人们在商业场所、酒店、机场等热点认证提供便利。通过 PYTHON 程序在后台来控制 Router OS^[5] 上的用户自动化认证，无需人工操作，用户只需要通过刷卡然后打印机自动打印上网用户名和密码，用户使用该密码便可以上网。认证弹出的界面可以投放广告，为商家带来利益。另外安装配置方便，配置完成后安全稳定，减少了工作人员的工作力度。

关键词：Raspberry Pi，刷卡，打印，Router OS，WIFI，热点认证

Design and Integration of Network User Management Module

rory

(School of Computer and Information Science, Southwest Forestry University, Kunming, 650224, Yunnan, China)

Abstract: Nowadays, when people go to Café, Entertainment, Bank Show, Beauty Shopping, hotel and other Entertainment places, almost everybody will ask if there is free WIFI? Universal speed of the WIFI is indeed amazing, and has become a major way of mobile Internet. Foreign cafes, shopping malls, hotels and other entertainment venues are basically covered with WIFI hotspots, customers can access free in their spare time, while businesses can make use of pop-up and delivery of advertising authentication interface to get commercial interests. WIFI hotspot authentication is very popular between businesses in foreign countries. Even on many airlines have begun to offer services of WIFI hotspot, so that passengers can surf the internet with their computers or other electronic products, enjoy e-mail in the air, passing data and so on.

Currently WIFI hotspot authentication scheme mainly via SMS certification, certification with manual operation, using a fixed password or no certification. SMS authentication method is that after the user accessed WIFI, hotspot authentication interface will pop up the phone, enter the phone number after clicking Obtain the password, enter the password to access Internet. There are some deficiencies of this kind of authentication, such as a user can not access Internet without a Phone, and can not effectively prevent the occurrence of rubbing network, if some malware bulk access to the Internet, then the router will be overwhelmed. Manual operation can prevent malicious software, but the operation cost of manpower and resources, the operator may cause network problems because of misuse. Fixed password or no authentication provides an easy way for everyone to rub network.

After market research analysis, domestic WIFI hotspot authentication schemes either complicated in configuration or insufficient in security and stability, which are mostly based on windows operating System, manual operation links leading to more time-consuming. This paper introduces a set based On Raspberry Pi^[8] of WIFI hotspot authentication scheme, Raspberry Pi are integrated with brushing-card printing function, providing people with conve-

nient hotspot certification in commercial establishments, hotels and airports. Router OS^[5] is controlled by PYTHON program for the automate user authentication in the background. Without manual operation, Users only need to Brush card and then the printer automatically prints the username and password, the user will be able to access by this username and password. Authentication pop-up interface can advertise, which Benefits businesses. It is easy to install and configure, the system is highly security and stability, reducing the efforts of the staff.

Key words: Raspberry Pi, Brush cards, Router OS, WIFI, hotspot

目 录

1	前言	1
1.1	研究背景	1
1.2	系统概述	2
1.3	系统特色	2
2	研究方案	4
2.1	问题研究	4
2.2	思路做法	4
2.3	创新点	5
3	方案实施	7
3.1	方案要求	7
3.2	材料准备	7
3.3	设计流程	7
4	配置 ROS 路由器	8
4.1	ROS 介绍	8
4.2	基本功能介绍	8
4.3	硬件准备	9
4.4	软件设置	9
4.4.1	Router OS 路由器各种登录方式	10
4.4.2	winbox 图形操作界面	10
4.4.3	MAC 层访问 (telnet 与 winbox)	11
4.4.4	常用命令信息	11
4.5	配置步骤	12

5	网络拓扑设计	13
5.1	设计要求	13
5.1.1	拓扑设备	13
5.1.2	拓扑结构	13
6	软件方案设计	16
6.1	软件需求分析	16
6.2	关于 Raspberry Pi	16
6.2.1	快速指南	16
6.3	PYTHON 模块	18
6.3.1	流程图	18
6.3.2	系统整合	19
6.3.3	功能函数	20
7	芭堤酒店实施	22
7.1	机房	22
7.2	前台	23
7.3	Raspberry Pi	23
7.4	实施心得	24
	指导教师简介	26
	致 谢	27
A	SD 卡兼容列表	28
A.1	工作的 SD 卡	28
B	配置文件	31
B.1	主程序配置文件 (conf.cfg)	31
B.2	ROS 配置脚本 (backup.rsc)	31

C 程序代码清单	34
C.1 pyhook 函数引用	34
C.2 主程序模块	34
C.3 打印模块	37
C.4 ros 模块	37
C.5 readcard 模块	46
C.6 myconf 模块	47

插图目录

4-1 winbox 登录界面	10
5-1 酒店拓扑	15
6-1 树莓派实物图	17
6-2 树莓派结构图	18
6-3 主流程图	19
7-1 机房	22
7-2 前台	23
7-3 树莓派	24

表格目录

6-1 函数说明	20
6-1 函数说明（续）	21
A-1 SD 卡工作表	28

1 前言

1.1 研究背景

“提供 WiFi 服务每年我们每一家分店额外增加的客流量是 15000。”

— *John Wooley, president and CEO of Schlotzsky's Deli (From Chainleader.com)*

“WiFi 服务，有助于吸引顾客和提高非高峰期的客流量。这些客人往往是“高收入”人士，他们习惯经常光顾，并停留较长的时间。”

— *Lovina McMurchy, director*

在国外一些酒店、餐厅、咖啡厅、商业中心或其他的商家，我们经常可以看到可以使用笔记本或手机使用商家提供的无线 WiFi 网络上网。这是一个非常好的手段，客人得到了便利，商家赢得了客流。任何咖啡店、酒店等的成功经营，关键在于吸引新客户和留住现有客户。提供无线上网服务，可帮助你达到这两个目标。很多商务人士、旅客和学生，即使他们在休假中，都往往需要跟外界保持各种各样的定期接触和联络。越来越多的人，如销售人员和客户代表等等，倾向于离开办公室，在一个较轻松自在的环境下，跟他们的客户会面。提供 WiFi 服务绝对给他们一个额外的理由光顾你的场所，并且留下来作更多的消费。WiFi 无线上网服务，可吸引更多的企业来到你的场所举行早餐或午餐聚会，同时也可吸引商务旅客到此用膳或逗留这样，你的场所就可作为这些公司及其雇员一个重要聚脚点，让他们轻松聚会的同时又可维持有一个高效率的工作环境。

目前国内的很多商业场所提供免费 WIFI 服务，例如酒店、咖啡店、餐饮店、机场等等，但是国内的这些场所提供的部分 WIFI 热点认证存在认证繁琐、人工操作、不够安全稳定等问题。认证繁琐确实让人头疼，就以北京的首都机场为例，它为旅客提供了以下几种获得 WiFi 上网账号的认证方式：在航站楼各处自助一体机上获取，在首都机场 WiFi 主页输入国内手机号码通过短信获取，使用腾讯微博或 QQ 账号登录，中国电信和中国联

通手机用户从独立的认证窗口输入手机号码及事先获得的 Wi-Fi 上网密码。方式似乎有很多，但了解起来却不是那么轻松，而且即使在正确无误地输入密码后，系统有时也反复提示密码错误，几次失败的尝试足以破坏用户使用 Wi-Fi 的兴致。

看起来使用 Wi-Fi 的用户似乎格外挑剔。要知道，Wi-Fi 一直被视作一种灵活、带宽大、使用率高的技术，“无缝覆盖”更是广为流传。因此，用户就自然对 Wi-Fi 期望颇高：方便、便宜、快捷。提高 Wi-Fi 的易用性，显然是必须提前做好的准备功课。

1.2 系统概述

该系统集成了 Raspberry Pi+ 刷卡器 + 打印机，使用一套 PYTHON 程序控制 MikroTik ROS 路由器实现 hotspot^[1] 认证功能，客户刷卡后打印机自动打印含有用户名密码的纸条，客户接入网络后会弹出认证界面，输入纸条上的用户名密码上网。用户如果长时间不上网便会自动下线，时间可以自己定义，下次还需要上网只需要再输入纸条上的用户名密码，有效利用了网络资源。比如入住酒店的用户在入住的时候就可以使用房卡刷一次，取纸条上网，只要客户不退房，可以凭纸条一直上网，当客户离开酒店的时候只需要再刷一次，自动下线纸条就作废了，酒店管理人员不用手工操作，非常方便。

1.3 系统特色

经过了一定的市场调查和分析后，我们针对国内 Wi-Fi 热点认证的方案的优缺点设计出了一套基于 Raspberry Pi 的网络用户认证管理系统，主要有以下特色：

自动化 基于 Raspberry Pi 集成的 PYTHON 程序实现自动认证功能，认证的方式无需工作人员手动操作，不需要手动添加上网帐号和密码，只需要把 Raspberry Pi 接入咖啡厅、酒店、或者机场的网络即可。

刷卡打印认证 顾客上网只需要刷卡，商家可以自定义用卡的种类来进行认证，比如说用购物卡、酒店房卡、公交卡、等等都可以用来认证，然后打印机自动打印纸条，凭借纸条上的用户名密码，客户接入热点信号自动弹出认证界面，输入纸条上给的用户名密码上网。

配置部署 该系统集成于 Raspberry Pi，程序存储与 SD 卡内，部署的时候只需要将 SD

卡插入树莓派，ROS 路由器接入商家的网络，修改一些参数即可成功部署，可以快速部署。

稳定 如果 Raspberry Pi 损坏了，不会影响网络的使用，只是暂时不能刷卡，只需跟换一个新的 Raspberry Pi 然后插入 SD 卡即可恢复。如果遇到断电或者人为把电拔了，恢复供电等 10 秒钟就又可以继续刷卡认证了。商家可以自定义卡的种类，因此使用其他卡是不能认证的，更安全，有效防止蹭网和恶意软件的攻击。

有线认证 比如酒店、学校等场所，不仅可以同时实现无线热点的认证，还可以实现有线网络的热点认证，该系统使用 MikroTik ROS^[10] 作为认证路由器，不仅可以支持无线认证，同时也支持有线认证。

WEB 认证 WiFi 热点认证的路的窗口可以作为商家的广告招牌，可以为商家自定义个性化广告界面。

自动下线 可以在 ROS 路由器内定义自动下线时间。如果用户想要自己注销，只需要再刷一次卡（下线打印机不出纸条）即可下线。

安全 输入方式只能通过刷卡，只有商家自己定义的卡才能刷，其他卡不能刷，有效防止了恶意软件消耗网络资源的行为。

2 研究方案

2.1 问题研究

- 该研究主要是：研究酒店行业 WIFI 解决方案，咖啡店行业 WIFI 解决方案，餐饮行业 WIFI 解决方案等。
- 重点解决的问题是：

Raspberry Pi 自动监听 使用 unix posix 标准的 `termios`^[12] 监听刷卡输入信号。

基于 **Raspberry Pi** 的网络用户身份认证功能 使用 **Raspberry Pi** 集成的刷卡功能获取卡号来判断用户身份，判断用户合法性等。

基于 **Raspberry Pi** 的刷卡打印功能 使用 **Raspberry Pi** 控制刷卡打印模块的工作。

程序模块的设计 读卡模块、打印模块、ROS 控制模块、监听模块、检验模块、主模块的设计与编码。

系统集成 集成 **Raspberry Pi**+ 刷卡器 + 打印机 +ROS，程序模块调用各个设备工作。

- 预期的结果是：提供一套完善的 WIFI 解决方案，不需要人工干预，完全集成于开发板，部署配置方便，自动化处理，更加安全稳定。在认证界面投放广告，给商家带来宣传的媒介和商业利益。

2.2 思路做法

前期进行市场调查，查看国内的 WIFI 解决方案现状，将来的趋势和市场的需要。决定产品的设计方案和研发进度。搜集技术信息，设计网络拓扑图，配置 ROS^[10] 实现 hotspot 认证，采用 Python 实现刷卡打印认证功能，使用 `wxpython`^[11] 实现用户接口，管理员能够通过该界面管理 ROS，测试成功无误后在 7 天酒店试运行，几个月来运

行稳定。后来又在芭堤酒店部署了另外一套，运行了一段时间后，由于管理员反馈 GUI 界面需要人工干预，操作麻烦，窗口需要在最前端运行才可以刷卡，给酒店的其他工作带来不便，于是设想出了第二套方案，使用 `pyhook`（见附录C.1）函数调用 `windows` 自动监听键盘接收刷卡字符串，这样的话 GUI 界面就不用每次在窗口最前运行了，也不会影响到酒店电脑的其他工作了，回来之后花了几个星期开始编码实现该功能。完成后在芭堤连锁酒店环城北路店投入使用，当时正好在做令一个关于 `Raspberry Pi` 的公交系统的项目，于是突发奇想，不如把程序移植到 `Raspberry Pi` 上实现相同的刷卡认证功能，并且让该系统实现自动化认证，去掉原来的 GUI 管理界面，使该系统完全集成在 `Raspberry Pi` 的开发板上，与酒店的管理系统分隔开来，这样的话就会更稳定也更方便。第三个版本设想完成后就开始了移植的工作，由于是基于 `Raspberry Pi` 的移植，因此就得重新设计刷卡器的读取方式，重新设计采用 `unix posix` 标准的 `termio`^[12] 信号来实现自动读取功能，然后再用 `rasprint` 模块实现 `pos` 打印机的打印驱动功能，然后重新进行 `python` 编码，删除了 `wxpython` 的 GUI 界面，界面的功能函数设置成监听触发，设置了一个持续监听的功能，并且上了锁（防止用户操作），将刷卡器，打印机和 `Raspberry Pi` 集成起来，几个月后第三个版本就完成了，之后投入到芭堤连锁酒店南亚风情点使用，部署时间大概半小时就可以完成，接下来的几个星期，由于路由器硬件的故障以及 SD 卡的不兼容，出现了几次问题，但是在后期的维护中都找出了这些问题，并且修复了这些问题，现在这套系统已经很稳定了，酒店管理员也很满意并且替换了芭堤酒店环城北路店的系统，使用到现在为止没出现过问题。特别要感谢我的指导老师孙永科一直指导我，我从售前到研发再到售后整个产品研发流程当中学会了不少东西，从而做出了这套产品，然后做成熟。

2.3 创新点

- 首次基于 `Raspberry pi` 集成，之前几个版本都是基于 `windows` 下的软件实现认证，现在转到了基于 `Raspberry Pi` 的 `linux` 系统下来进行用户认证，把刷卡打印功能都移植到了 `linux` 环境下，并且长期使用下来很稳定。出问题或者坏掉只需跟换 SD 卡即可，更安全稳定。
- 对比国内的认证方法，还没有使用刷卡打印来进行 `hotspot` 认证，该方法更是很酒店、咖啡厅、商场等商业场所，使用下来用户体验不错。

- 自动化操作，不需要人工操作和干预，给酒店网络管理人员减轻了很大工作量，过去酒店的工作人员都需要手工添加用户来给用户使用，要么就是给一个固定密码，这样的话很容易被蹭网。反而不安全了。自动化操作的功能使得工作人员可以不用关心给用户认证，只需要刷卡就可以自动认证。
- 配置方便快捷，每次部署只需要连接好设备，配置好 ROS 网关和路由器密码，把 Raspberry Pi+ 刷卡器 + 打印机接入内网即可工作。
- 不依赖操作系统，PYTHON 程序 + 打印机 + 刷卡器集成到了 Raspberry Pi 上，整套设备一起工作，不依赖于酒店的设备和其他系统。
- 独立操作，不影响酒店其他设备，同时也不会受其他设备影响。即使 Raspberry Pi 断电了，酒店网络不受影响，只是暂时不能刷卡，恢复供电后又可以刷卡了。经过多次断电插电试验，系统很稳定。
- 操作简单易于理解
 - 用户入住酒店刷卡，打印机打印上网纸条，用户凭纸条上网。
 - 用户退房刷卡，自动注销，打印机不打印纸条。
 - 根据刷卡是否处置即可判断是入住还是退房。
 - 开关式的上线下线。

3 方案实施

3.1 方案要求

- 树莓派实现自动控制系统。
- 接入 hotspot 后打开任意网站显示欢迎界面。
- 欢迎界面显示自定义广告。
- 客户刷卡后打印机自动出纸，在欢迎界面输入用户名密码登录即可上网。
- 客户再次刷卡后注销用户，上网纸条失效。
- Raspberry+PYTHON 程序 + 打印机 + 刷卡器 +ROS 系统整合。
- 对于不同身份用户进行不同限制, 如限速，上网时间, 防火墙策略等等。

3.2 材料准备

- Raspberry Pi 一个（见图 6-1）
- 与 Raspberry Pi 兼容的 SD 卡一张（见附录 A.1）
- MikroTik ROS 路由器一台
- IC 卡刷卡器一个
- IC 卡若干（测试用）
- POS 打印机一台

3.3 设计流程

1. 网络拓扑设计
2. 配置 ROS 路由
3. 软件方案设计
4. PYTHON 编码

4 配置 ROS 路由器

4.1 ROS 介绍

MikroTik RouterOS^[10] 是一种路由操作系统，并通过该软件将标准的 PC 电脑变成专业路由器，在软件的开发和应用上不断的更新和发展，软件经历了多次更新和改进，使其功能在不断增强和完善。特别在无线、认证、策略路由、带宽控制和防火墙过滤等功能上有着非常突出的功能，其极高的性价比，受到许多网络人士的青睐。RouterOS 在具备现有路由系统的大部分功能，能针对网吧、企业、小型 ISP 接入商、社区等网络设备的接入，基于标准的 x86 构架的 PC。一台 586PC 机就可以实现路由功能，提高硬件性能同样也能提高网络的访问速度和吞吐量。完全是一套低成本，高性能的路由器系统。

4.2 基本功能介绍

- 完整的防火墙设置功能，隧道协议和 Ipsec 功能；
- 多线路支持 (负载均衡、双线路由、静态路由、策略路由)；
- P2P 传输过滤功能高效的 VRRP(虚拟路由冗余协议) 强大的 QoS 功能带宽流量控制；
- 网桥功能并支持 STP；
- 支持 WEP 的超高速 802.11a/b/g 无线协议；WDS 和虚拟 AP 特征；
- 支持即插即用 UPNP；
- RIP, OSPF, BGP 路由控制协议；
- 提供 1G 以太网接口；
- 同步支持 V.35, X.21, T1/E1 系列；
- 实现 Hotspot 的 web 网络认证管理功能；
- 支持 PPPoE 和 PPTP 客户端与服务器；
- Radius 计费功能；IP 电话功能 (H323)；

- 远程 WinBox 和 Web 管理;
- telnet/ssh/serial 管理控制台;
- 实时配置和监控;

4.3 硬件准备

1. 首先下载软路由的 ghost 硬盘版,如果没有,从<http://www.nbwq.com/download/ros297.rar>下载
2. 释放后, ghost 至一个小硬盘 (20G 以下), 注意, 是整盘 GHOST 而不是分区。
3. 将该硬盘挂在要做路由电脑上, 注意必须接在第一个 IDE 并且是主硬盘接口。插上一张网卡, 这是接内网的 LAN。开机。

4.4 软件设置

1. 开机, 出现登陆提示。用户:admin 密码: 空
2. 输入 setup 再按两次 A
3. 在 ether1 后面输入你的内网 IP, 如: 192.168.0.254/24 (这里/24 是 24 位掩码与 255.255.255.0 一样)
4. 输入完 ip 后, 按两次 x 退出, 现在可以可以 ping 通 192.168.0.254 了, 也可用 winbox 在图形界面下访问路由了。
5. 关机, 插上另一张网卡, 这个是接外网的, 即 WAN, 现在可以去掉软路由电脑的显示器和键盘了。
6. 开机, 运行 winbox 以 admin 身份登陆
7. 添加外网网卡。在 ip—address 里按 +, address 输入你的外网 ip 和掩码位, 比如 218.56.37.11/29。network 和 BROADCAST 不填, INTERFACE 里选择 ethr2
8. 增加外网网关。ip-routes 按 +, Destination 用默认的 0.0.0.0/0 , Gateway 输入外网网关, 比如 218.56.37.10
9. 实现 NAT 转发: IP-FIREWALL 在 NAT 里点 +, 在 ACTION 里选 masquerade

10. 现在该路由已经做好雏形，可以正常上网了。

4.4.1 Router OS 路由器各种登录方式

Mikrotik routerOS 内能通过远程配置各种参数, 包括 Telnet,SSH,winbox 和 Webbox。

winbox 运行界面

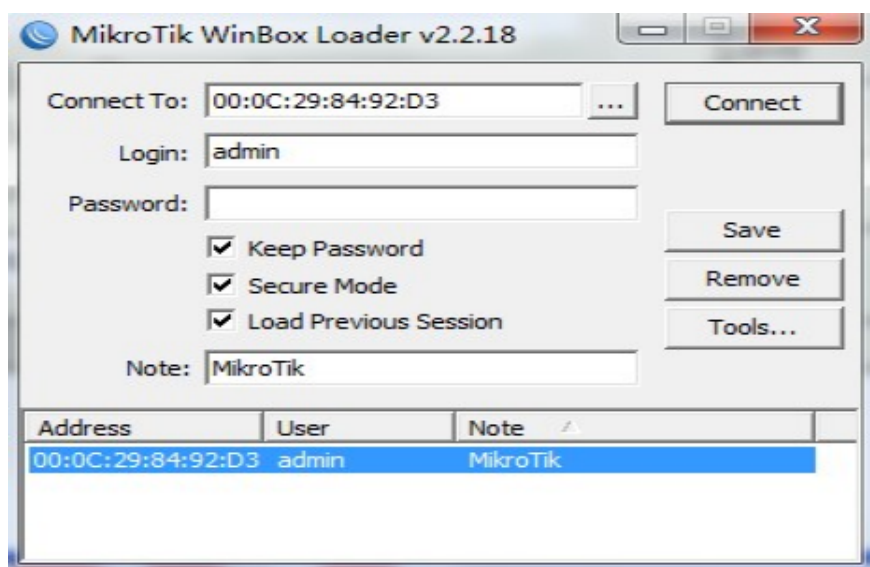



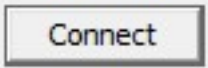
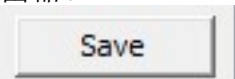
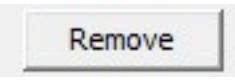
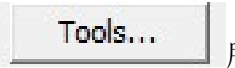
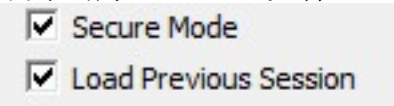
图 4-1 winbox 登录界面

MAC-telnet 是在路由器没有 IP 地址的情况下或者配置防火墙参数后无法连接, 通过路由器网卡 MAC 地址登陆的方式远程连接到路由器, MAC-telnet 仅能使用在来自同一个广播域中 (因此在网络中不能有路由的存在), 路由器的网卡应该被启用。注意, 在 winbox 中通过 MAC 地址连接路由器的功能, 并内置了探测工具。这样在管理员忘记或复位了路由器后, 同样可以通过 MAC 登录到 routerOS 上, 进行图形界面操作

4.4.2 winbox 图形操作界面

Winbox 控制台适用于 mikrotik routerOS 的管理和配置, 使用图形管理和配置, 使用它型管理接口 (GUI)。通过连接到 mikrotik 路由器的 HTTP(TCP 80 端口) 欢迎界面

下载 Winbox.exe 可行文件, 下载并保存在你的 windows 中, 之后直接在你 windows 电脑上运行 Winbox.exe 文件。

-  搜索和显示 MNDP (mikrotik Neighbor discovery protocol) 或 CDP(cisco discovery protocol) 设备, 可以通过该功能键搜索同一子网内 mikrotik 和 cisco 设备, 并能通过 MAC 地址登陆到 mikrotik routerOS 进行操作。
-  通过指定的 IP 地址 (默认端口为 80, 不许特别指定, 如果你修改了端口需要对具体访问端口做自定) 或 MAC 地址 (如果路由器在同一子网内) 登陆路由器。
-  保存当前连接列表 (当需要运行它们时, 只需双击)。
-  删除从列表中选择的项目。
-  所有列表中的项目, 清除在本地的缓存, 从 Winbox 文件导入地址或导出为 Winbox 文件。
-  Secure mode (安全模式): 提供保密并在 Winbox 和 routerOS 之间使用 TLS(transport layer security) 协议; Keep password(保存密码): 保存密码到本地磁盘的文本文件中。

4.4.3 MAC 层访问 (telnet 与 winbox)

通过 MAC 地址进行链接是用来访问没有设置 IP 地址的 routerOS 路由设备, 这种连接类似于 IP 地址连接, 通过 MAC 地址仅在限于 2 台 mikrotik routerOS 路由器之间进行, 操作路径: /tool/mac-server。

4.4.4 常用命令信息

在任何操作目录使用 ‘?’ 都可用获在当前目录中的命令信息。

log/ — 系统日志

Quit — 退出控制台

Radius/ — 客户端设置

Certificate/ — 授权管理

Special-login/ —特殊登陆用户

Ping-ping —命令

Setup —基本的系统设置

Interface/ —接口配置

Password —修改密码

Undo —撤销以前操作

Port/ —串口控制

User/ —用户管理

System/ —系统信息和应用程序

Ip/-ip —选项

Tools/ —工具

Routing —各种路由协议设置

.. —回到根目录

Address/ —地址管理

Route/ —路由管理

Firewall/ —防火墙管理

DHCP-client —DHCP 客户端设置

DHCP-server/ —DHCP 服务设置

Hotspot/-hotspot —管理

Pool-ip —地址池

一个指令或一个变量参数不需要完整的输入, 如是含糊不清的指令或变量参数需要完整的输入 `interface` 时, 你只要输入 `in` 或 `int`, 需要显示完整的指令可以使用 `[tab]` 键通过指令的组合, 可以在当前的目录执行在不同目录操作

4.5 配置步骤

基本配置完成以后需要配置我们需要的 ROS, 对于酒店的特殊需求, 我自己配置了一份 ROS 配置文件, 这样的好处是可以批量配置路由器, 不用每次手工配置, 步骤如下:

1. 使用 `winbox` 登录 ROS, 输入用户名密码。
2. 鼠标点击 `New Terminal`, 出现一个终端, 复制配置文件 `backup.rsc` (见附录 B.2)。
3. 鼠标放到终端上点击粘贴。
4. 配置成功。
5. 配置文件详见附录 B.2。

5 网络拓扑设计

5.1 设计要求

酒店的外网接 ROS 的 eth0 口内网接 eth1 口，对所有交换机使用 hotspot 认证，酒店员工办公室的电脑、前台电脑、特殊用户电脑不需要认证，其他普通用户电脑都需要进行认证。该集成的系统放到酒店一楼的前台处给客户刷卡取纸条，系统只需要在前台接入酒店内网即可。对酒店普通用户限速 400K，办公室限速 600K，特殊帐号不限速。用户两小时内不上网自动下线。

5.1.1 拓扑设备

- Raspberry Pi（树莓派），刷卡器，POS 打印机三者集成为一个整体，接到一楼网络的前台处。
- 只需把树莓派接入酒店内网即可工作。
- 总交换机即为我们的 ROS 路由，通过该总路由即可管理整个酒店网络
- 比如一楼的用户要上网，通过刷卡然后取走纸条即可上网
- 树莓派实时控制 ROS 路由

5.1.2 拓扑结构

这里显示了一楼前台处的树莓派 + 刷卡器 + 打印机三者为一个整体，连接外网的是 ROS，ROS 的 eth1 接入酒店内网交换机。

其他终端用户上网都需要经过 ROS 的认证，通过到前台刷卡然后如果是入住酒店打印机打印出一张纸条，客户取走纸条后接入酒店网络弹出认证界面，按照纸条上的用户名密码输入上网。

客户退房的时候在刷一次卡，这次打印机不会出纸条，听到嘀的一声后就注销了，纸条上的用户名和密码作废。

通过在 ROUTER OS 中设置 IP→Hotspot→Ip Bindings 设置不需要认证的 ip, 但是对于要批量设置的话可以使用其中一个不需要认证的 ip 接入办公室的路由器, 办公室就不用经过认证了, 办公室限速 600K 只需要限制这一个 ip 的网速就可以了。对于普通用户的限速在 Queues 里面写规则限制。

在 Queues 里面写限速规则

1. ROS 单 IP 限速

- (a) 首先登陆 WINBOX, 点击 QUEUES, 这个就是限速窗口, 现在是空白的, 再点击“红色 + 号”新建一个限速规则。
- (b) 名字自己起, 最好是一看就明白的, target upload: 目标上传; target download: 目标下载。这里要说明一下, 在 ROS 里表达的流量值单位是位 (bit), 而我们平时说的申请了 2M 的 ADSL, 这个 2M, 单位也是位 (bit), 但是我们说的下载速度多少 K, 指的单位是字节, 它们之间的比例是: 8 位 (bit) = 1 字节, 所以, 上传 $128k/8 = 15K$ (请注意大小写: $k * 8 = K$)。下面还可以设置每个星期几限速, 全打上就是每一天都限的意思。
- (c) 当我们设置完了限制的速度后, 我们还要设置要限制的电脑 IP, 才能实现效果哦。点击 advanced (高级) 栏, 在 dst. address 处输入你要限制速度的 IP 地址, 然后点 OK 完成。
- (d) ROS 限速的确很好用, 限了多少就是多少, 非常准确的。

2. ROS 网段限速

- (a) 首先登陆 WINBOX, 点击: system-scripts
- (b) 通过脚本来对整个网段限速, 先点“红色 + 号”, 新建一个脚本, 名字你喜欢, 在 SOURCE 处粘贴源代码

```
1      :for myip from 2 to 250
2      do={
3          /queue simple add name=(“限 192.168.1.” . $myip . “”)
4          target-address=(“192.168.1.” . $myip . “/32”)
5          limit-at=0/0 max-limit=150000/1024000
6      }
```

- (c) 即可以 192.168.1.2-250 限速

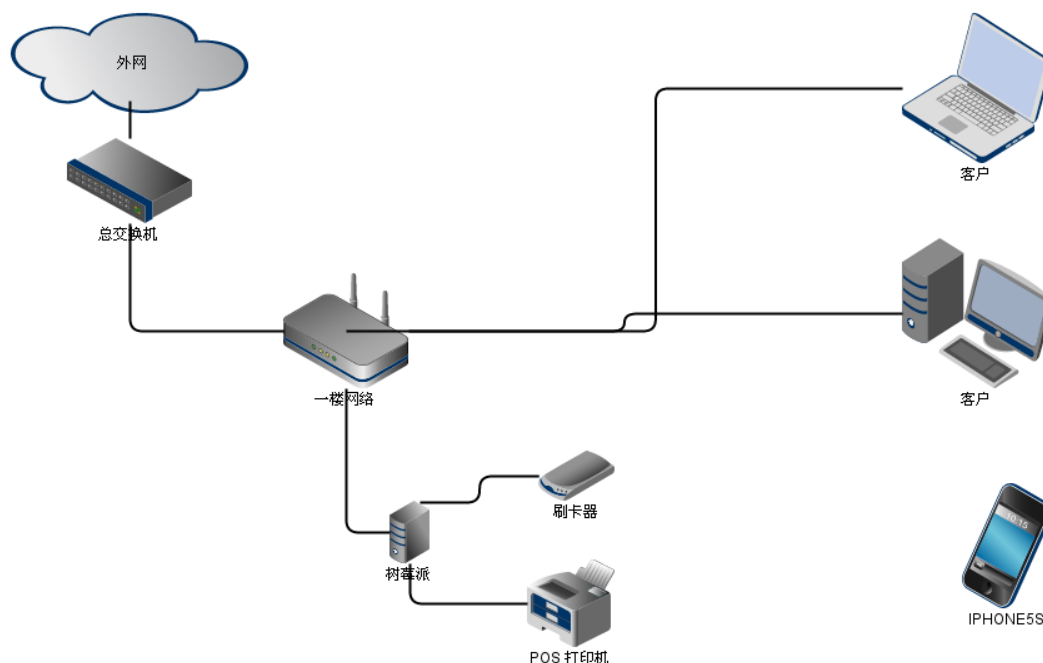


图 5-1 酒店拓扑

(d) 创建了脚本，还要运行才能生效。选中要运行的脚本，再点击 RUN...，这样整个网段都可以实现统一限速了。

酒店网络拓扑如图 5-1 所示。注意，这里仅说明了酒店的一楼网络，其他楼层同样需要经过认证才可以上网，树莓派接入酒店内网任意位置都可以进行啥卡认证，客户的笔记本的电脑、PC、智能手机都需要认证上网，办公室的位置没有标出。

6 软件方案设计

6.1 软件需求分析

通过查阅资料 and 进行实地勘察发现,WIFI 网络的管理处于一种非常不稳定的现象,蹭网现象的频繁发生,网络资源的大量的浪费,配置的繁琐。消费客户入住酒店过程中,询问帐号和密码的次数过多,给前台工作人员带来了工作力度,也给消费客户带来了诸多的不便。针对这些情况,我们提出了设计一种能够达到自动化管理 WIFI 上网,防止蹭网和减少工作力度的软件,该软件还可以带来广告效应和增值服务。

6.2 关于 Raspberry Pi

Raspberry Pi 是一个信用卡大小的电脑,可接入 TV 和键盘。这是一个能够在电子项目中使用的小容量 PC,同时可以做到像你的普通笔记本电脑一样的事情,诸如电子表格,文档处理和游戏。它还能播放高清晰度视频。我们希望看到全世界的孩子都使用它学习编程^[8]。

6.2.1 快速指南

摘自《树莓派快速入门手册》(Raspberry Pi Quick Start Guide, <http://www.raspberrypi.org/help/quick-start-guide/>):

需要的硬件

图 6-1所示就是我们使用的树梅派。连接树梅派的还有如下一些硬件设备:

- SD 卡一张(我们选用的是兼容树莓派的 SanDisk 卡见附录 A.1)
- HDMI 高清线, VGA 转接头, 网线
- 键盘鼠标



图 6-1 树莓派实物图

- 电源

树莓派的物理结构如图 6-2所示。

树莓派的硬件连接

1. 首先将 SD 卡插入在树莓派上的 SD 卡插槽，只有一种方式插入。
2. 接下来，把 usb 键盘和 usb 鼠标插入树莓派自带的两个 usb 口上。
3. 确保你的显示器是开着的，接着你需要选择你的接入方式（HDMI 或者 DVI 等等）。
4. 接着把使用树莓派的 HDMI 高清线接入树莓派，使用转接头接显示器的 VGA 插口。
5. 如果你希望你的树莓派能够联网，插入网线到网口，就在 usb 口旁边，如果不要联网则跳过此步。
6. 这时你应当很高兴你已经接好了所有的线以及插入了需要的 SD 卡，最后一部就是接入电源，树莓派就起来了。

登录树莓派

1. 当你的树莓派已经完成启动进程后，一个登录提示符将会显示出来。默认登录树莓派的用户名是 `pi` 密码是 `raspberrypi`。注意：当你输入密码的时候你看不到密码，这是 linux 系统的安全特性。

RASPBERRY PI MODEL B

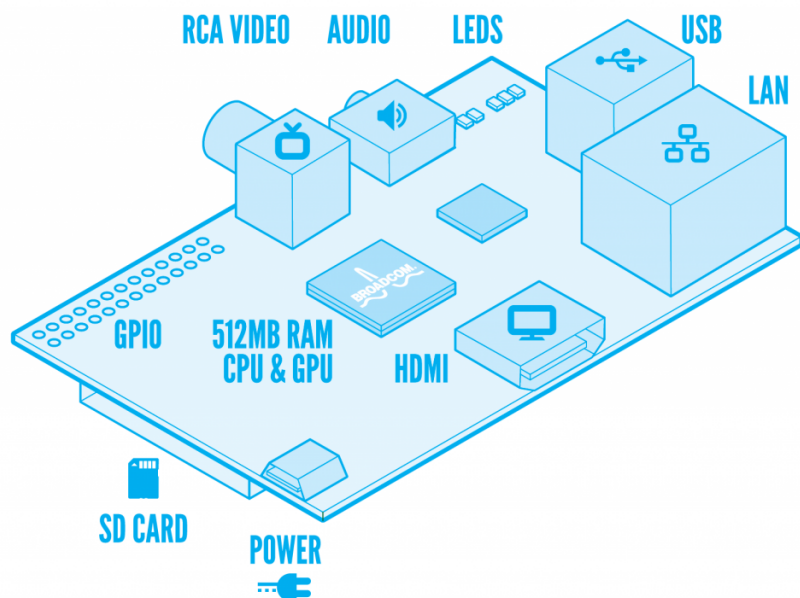


图 6-2 树莓派结构图

2. 在你成功登录以后，你将会看到这样的命令提示符 `pi@raspberrypi~$`。
3. 要进入图形化的操作界面，输入 `startx` 然后回车，一个漂亮的用户界面就出来了。

6.3 PYTHON 模块

6.3.1 流程图

整个系统的工作流程如图6-3所示。

1. 主程序开始 → 进入 `main` 函数等待用户的输入（只有刷卡作为输入）。
2. 接收用户输入判断是否等于 `matriux`（用于中断主函数等待输入的状态，便于调试）。
3. 如果输入等于预设字符串 `matriux`，则中断监听状态，进行调试。
4. 如果输入不等于 `matriux`，则假定输入是用户刷卡的输入。
5. 判断该输入是否为空，如果为空则返回重新等待输入。

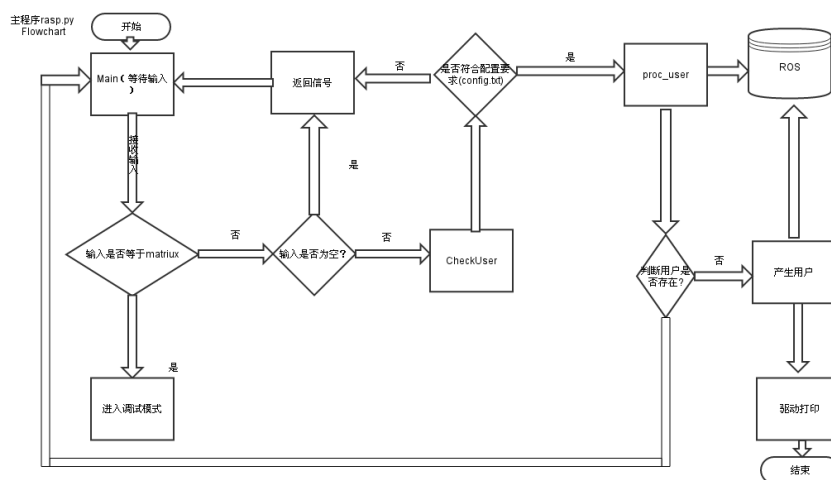


图 6-3 主流程图

6. 如果不为空，使用 `CheckUser` 检查用户合法性（合法性的定义在 `config.cfg` 文件中，见代码 B.1。）
7. 经判断不符合 `config.cfg` 则返回重新等待输入，若符合要求则调用 `proc_user`。
8. `proc_user` 首先查询 `ROS` 路由器中是否已经存在该用户，若存在则踢用户下线。若不存在则生成新用户信息，存入 `ROS`。
9. 存入 `ROS` 成够后调用打印机打印用户名和密码给用户上网使用。
10. 继续等待其他输入 → 结束。

6.3.2 系统整合

1. 安装 PyUSB

- (a) 从 Sourceforge¹上面下载最新的tarbell
- (b) `unzip pyusb*.zip`
- (c) `cd pyusb*`
- (d) `python setup.py build`
- (e) `sudo python setup.py install`

2. 安装escpos

- (a) `wget http://python-escpos.googlecode.com/files/python-escpos-`

¹<http://sourceforge.net/>

- ```
1.0.tgz
```
- (b) `tar zxvf python-escpos-1.0.tgz`
  - (c) `cd python-escpos-1.0`
  - (d) `python setup.py build`
  - (e) `sudo python setup.py install`
3. 安装其他软件:`apt-get install python-imaging python-usb python2.7-usbtc08 python-serial`
  4. PYTHON 编码完成后在 Raspberry Pi 上测试
  5. 各个程序模块已经能够在 Raspberry Pi 的 linux 系统下运行
  6. 定制系统开机脚本, 开机自动运行 PYTHON 程序
  7. 测试系统稳定性

### 6.3.3 功能函数

表 6-1列出了程序中主要的一些函数以及函数对应的用法。

表 6-1 函数说明

| 函数                             | 功能                                     |
|--------------------------------|----------------------------------------|
| <code>getAdminInfo</code>      | 获取 <b>admin</b> 用户的信息                  |
| <code>getNewPwd</code>         | 通过给定字符串随机构造新密码                         |
| <code>checkUser</code>         | 检查用户的合法性, 通过 <b>config</b> 文件检查, 返回用户名 |
| <code>proc_user</code>         | 写入 <b>ros</b> 数据库, 产生用户名和密码            |
| <code>getpass</code>           | <b>termios</b> 监听程序, 持续监听刷卡获得的卡号       |
| <code>getCardNum</code>        | 刷卡后获得卡号                                |
| <code>termios.tcgetattr</code> | 获取文件描述符                                |
| <code>termios.tcsetattr</code> | 设置文件描述符                                |
| <code>ros.getUserlist</code>   | 获取用户列表                                 |
| <code>ros.delUser</code>       | 删除用户                                   |
| <code>ros.addNewUser</code>    | 添加用户                                   |
| <code>rasprint.raspout</code>  | 打印输出                                   |

表 6-1 函数说明（续）

|                            |                                                                                              |
|----------------------------|----------------------------------------------------------------------------------------------|
| Printer.Usb(0x04b8,0x0202) | 获取打印机标识、用于本地打印<br>用命令“lsusb”获取到当前打印机的”<br>Vendor ID”和” Product ID”<br>假如获取到的是 0x04b8:0x0202, |
| main                       | 主程序                                                                                          |

## 7 芭堤酒店实施

### 7.1 机房

机房实景如图 7-1所示。我们的 ROS 放在机架最上面，负责整个酒店的用户身份认证功能，用下来性能还不错。**eth0** 口连接酒店的外网，**eth1** 连接酒店的总交换机，总交换机连接每层楼的交换机，每层楼的终端都要经过 ROS 路由器的认证才能够上网。

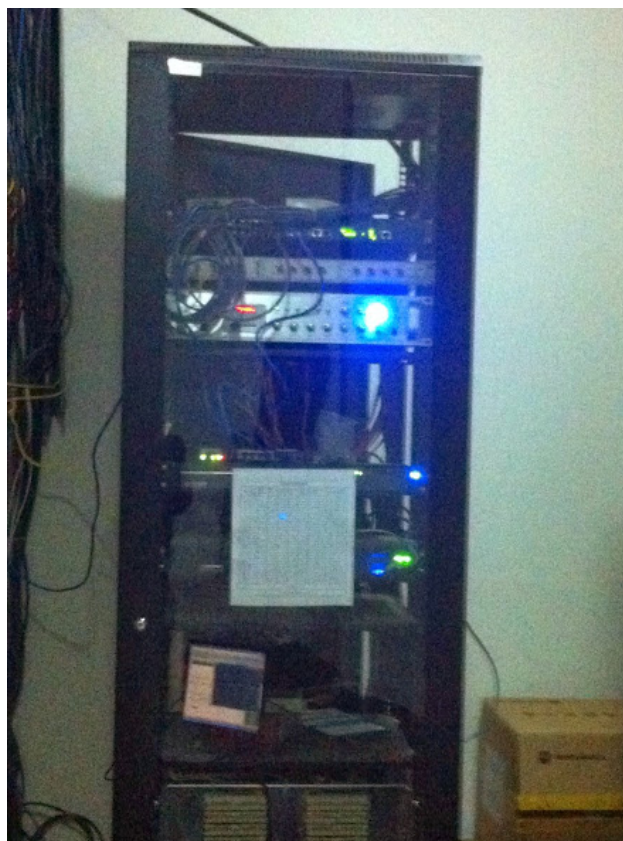


图 7-1 机房



## 7.2 前台

前台实景如图 7-2所示。我们的树莓派 + 打印机 + 刷卡器，树莓派不需要人接触，直接放在显示器后面。只要露出刷卡器和打印机给用户刷卡取纸就行了。刷卡器就是键盘旁边那个，打印机放在旁边。

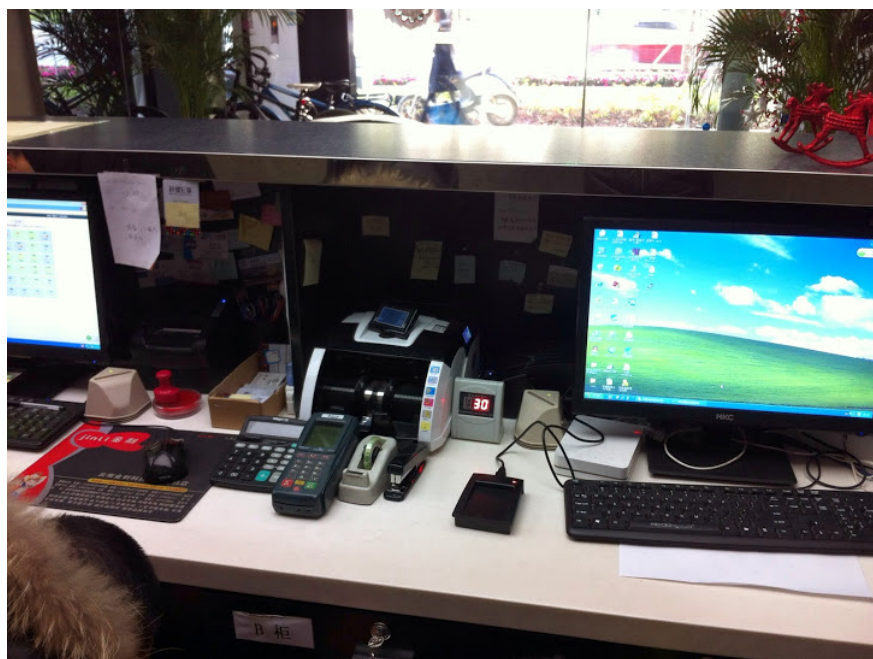


图 7-2 前台

## 7.3 Raspberry Pi

图 7-3所示就是躲在角落里的树莓派，负责整个酒店的网络用户认证，总共有两个usb 接口，刷卡器接在树莓派的usb1 口上，放在外面给人们刷卡，打印机接在树莓派usb2 口上，放在外面打印用户名密码，客户拿到纸条后安装纸条上的说明打开浏览器输入任意网址便会弹出带有 badi 酒店 logo 以及广告的广告的登录界面，然后输入用户名密码即可上网，客户离开的时候再去刷一次卡就自动下线了。另外使突然断电也不影响酒店的网络使用，仅仅是暂时不能刷卡，只要恢复供电就又可以刷卡了。





图 7-3 树梅派

### 7.4 实施心得

通过在芭堤酒店的实施我得到了锻炼，我发现自己平时在学校只顾开发，从来没有和客户交流的概念，这样的话设计出来的软件就会没有市场，我学会了要和客户交流，根据客户提出的一些建议改进系统，比如客户提出希望操作简单方便，不要影响前台的电脑使用，我们就设想出了单独集成一套 **Raspberry Pi** 的方案，简化了认证方式，使得工作人员不必手工去添加用户，同时不会影响前台电脑的使用。随着使用时间的过去，根据反馈一点点改进，现在已经很稳定了，客户也很满意，产品也更成熟了。

## 参考文献

- [1] Geier, Eric. Wi-Fi Hotspots. Cisco Press, 2006.
- [2] Richardson, Matt, and Shawn Wallace. Getting Started with Raspberry Pi. O'Reilly Media, Inc., 2012.
- [3] Halfacree, Gareth, and Eben Upton. Raspberry Pi User Guide. John Wiley & Sons, 2012.
- [4] Monk, Simon. Programming the Raspberry Pi: Getting Started with Python. McGraw-Hill, 2013.
- [5] 崔北亮. 《Router OS 全攻略》, 电子工业出版社. 2010.
- [6] Beazley, David M. Python essential reference. Addison-Wesley Professional, 2009.
- [7] Rappin, Noel, and Robin Dunn. wxPython in Action. Manning, 2006.
- [8] Raspberry Pi Quick Start Guide, <http://www.raspberrypi.org/help/quick-start-guide/>
- [9] Wikipedia: Raspberry Pi, [http://en.wikipedia.org/wiki/Raspberry\\_Pi](http://en.wikipedia.org/wiki/Raspberry_Pi)
- [10] Wikipedia: MikroTik, <http://en.wikipedia.org/wiki/MikroTik>
- [11] Wikipedia: WxPython, <http://en.wikipedia.org/wiki/WxPython>
- [12] Termio Manual, <http://man7.org/linux/man-pages/man7/termio.7.html>

## 指导教师简介

孙永科，男，1980 年出生，硕士，讲师，计算机科学与技术专业。主要从事软件开发和计算机网络的教学工作，主讲课程为《计算机网络》，《网络程序设计与开发》。参加科技部重大科技工程“西南野生生物物种质资源库”子课题“西南野生生物物种质资源库信息系统”的开发工作；参加科研项目《运用现代教育技术的实践与成果》获云南省二等奖；参加科研项目《树木学教育教学建设与研究》获云南省二等奖。

## 致 谢

非常感谢在大学期间的各位老师教授了我很多东西，并且在学习和生活中给予了我很大帮助，在这四年期间，我学到了很多在学校外面学不到的东西，在何鑫老师和辛宏老师的启蒙下我开始喜欢上编程，接触到了面向对象、MVC 的思想，打好了编程的基础，非常感谢他们。还要感谢王晓林老师让我很幸运地接触到了世界上最好玩的系统和编辑器。还要感谢孙永科老师，在您的带领下我试着自己尝试创造很多有趣的东西，通过动手实践我学会了很多网络知识，喜欢上了 PYTHON，学会了把书本知识应用到每一个有趣的项目里，从研发产品，再到售前售后懂得了如何制定研发计划，和客户沟通改善用户体验，非常感谢您。另外还有很多我们学院的优秀老师给予了我很大的帮助，谢谢你们。

## 附录 A SD 卡兼容列表

### A.1 工作的 SD 卡

表 A-1 SD 卡工作表

| OK | Manufacturer | Type | Size | Class | Model                 |     |
|----|--------------|------|------|-------|-----------------------|-----|
| ok | SanDisk      | SD   | 2    | ?     | Extreme               | III |
|    |              |      |      |       | (BE0715105083B)       |     |
| ok | SanDisk      | SD   | 2    | 2     | BE0816113150D         |     |
|    |              |      |      |       | writes at 3.5 Mb/s    |     |
| ok | SanDisk      | SD   | 2    | 4     | Ultra 15 MB/s         |     |
| ok | SanDisk      | SD   | 2    | 4     | Ultra                 | II, |
|    |              |      |      |       | BE0719111366D         |     |
| ok | SanDisk      | SD   | 2    | 6     | Extreme               | III |
|    |              |      |      |       | (BE0804212046D)       |     |
|    |              |      |      |       | 20 MB/s               |     |
| ok | SanDisk      | SD   | 2    | ?     | SanDisk for Wii       |     |
| ok | SanDisk      | SDHC | 4    | 2     | BH0820113475D         |     |
| ok | SanDisk      | SDHC | 4    | 4     | SDSDB-004G-B35,       |     |
|    |              |      |      |       | BH1210821913G,        |     |
|    |              |      |      |       | SDSDB-004G-BT35       |     |
| ok | SanDisk      | SDHC | 4    | 4     | Ultra II              |     |
| ok | SanDisk      | SDHC | 4    | 6     | Extreme III (30 MB/s) |     |
| ok | SanDisk      | SDHC | 4    | 6     | Ultra SDSDH-          |     |
|    |              |      |      |       | 004G-U46              | -   |
|    |              |      |      |       | BH1136121837G,        |     |
| ok | SanDisk      | SDHC | 4    | 10    | Extreme               |     |
|    |              |      |      |       | (BH10297143382G)      |     |
| ok | SanDisk      | SDHC | 4    | 10    | Extreme (SDSDX-       |     |
|    |              |      |      |       | 004G-X46)             |     |
| ok | SanDisk      | SDHC | 8    | 4     | writes at 1.5MB/s     |     |
| ok | SanDisk      | SDHC | 8    | 4     | Ultra BI1024716014G   |     |

<sup>0</sup>参见 Working / Non-working SD cards - [SD 卡工作表](#)

<sup>0</sup>本表未完全包括所有 SD 卡，仅列出 SanDisk

表 A-1 SD 卡工作表 (续)

|    |         |           |    |    |                                                         |
|----|---------|-----------|----|----|---------------------------------------------------------|
| ok | SanDisk | SDHC      | 8  | 4  | SDSDB-008G-B35S /<br>SDSDB-008G-B35                     |
| ok | SanDisk | SDHC      | 8  | 4  | SDSDB3-008G-A16BJ                                       |
| ok | SanDisk | SDHC      | 8  | 6  | Ultra SDSDH-008G-<br>U46                                |
| ok | SanDisk | SDHC      | 8  | 10 | Extreme                                                 |
| ok | SanDisk | SDHC      | 8  | 10 | Extreme                                                 |
| ok | SanDisk | SDHC      | 8  | 10 | Extreme Pro                                             |
| ok | SanDisk | SDHC      | 8  | 10 | Extreme Pro                                             |
| ok | SanDisk | SDHC      | 8  | 10 | Ultra SDSDU-008G-<br>U46 (30 MB/s)                      |
| ok | SanDisk | SDHC      | 8  | 10 | Ultra SDSDU-008G-<br>A11 (30 MB/s)                      |
| ok | SanDisk | SDHC      | 8  | 10 | Ultra SDSDU-008G-<br>T11 (30 MB/s)                      |
| ok | SanDisk | SDHC      | 8  | 10 | Ultra SDSDU-008G-<br>UQ46 (30 MB/s)                     |
| ok | SanDisk | SDHC      | 16 | 4  | SDSDB-016G-B35                                          |
| ok | SanDisk | SDHC      | 16 | 4  | SDSDB-016G-A11                                          |
| ok | SanDisk | SDHC      | 16 | 6  | Ultra (30MB/ s)<br>(BL1133921933G)                      |
| ok | SanDisk | SDHC      | 16 | 10 | Extreme                                                 |
| ok | SanDisk | SDHC      | 16 | 10 | Extreme Pro                                             |
| ok | SanDisk | SDHC      | 16 | 10 | Ultra(30MB/ s)<br>(SDSDU-016G-U46)<br>(SDSDU-016G-U46S) |
| ok | SanDisk | microSDHC | 16 | 10 | Ultra microSDHC I<br>UHS-I                              |
| ok | SanDisk | SDHC      | 16 | 10 | EXTREME SDHC UHS-<br>I (SDSDX-016G-X46)                 |
| ok | SanDisk | SDHC      | 32 | 4  |                                                         |
| ok | SanDisk | SDHC      | 32 | 6  |                                                         |
| ok | SanDisk | SDHC      | 32 | 10 | Extreme (45MB/ s<br>UHS-I) (SDSDX-032G-<br>X46)         |
| ok | SanDisk | SDHC      | 32 | 10 | Ultra (30 MB/s)                                         |

表 A-1 SD 卡工作表 (续)

|    |         |           |    |    |                                     |
|----|---------|-----------|----|----|-------------------------------------|
| ok | SanDisk | SDHC      | 32 | 10 | Elevate (30 MB/s)<br>SDSDU-032G-T11 |
| ok | SanDisk | SDXC      | 64 | 10 | Extreme (SDSDX-064G-X46)            |
| ok | SanDisk | SDXC      | 64 | 10 | Ultra SDXC UHS-I FFP (3A114807)     |
| ok | SanDisk | microSDHC | 4  | 2  |                                     |
| ok | SanDisk | microSDHC | 4  | 4  |                                     |
| ok | SanDisk | microSDHC | 8  | 2  |                                     |
| ok | SanDisk | microSDHC | 8  | 4  |                                     |
| ok | SanDisk | microSDHC | 8  | 4  | SDSDQM-008G-B35                     |
| ok | SanDisk | microSDHC | 8  | 10 | Ultra (SDSDQU-008G-U46) 30 MB/s     |
| ok | SanDisk | microSDHC | 16 | 10 | Mobile Ultra (SDSDQUA-016G-U46A)    |
| ok | SanDisk | microSDHC | 32 | 4  |                                     |
| ok | SanDisk | microSDHC | 32 | 10 | SDSDQU-032G-U46A                    |
| ok | SanDisk | microSDXC | 64 | 6  | SDSDQY-064G-A11A                    |

## 附录 B 配置文件

### B.1 主程序配置文件 (**conf.cfg**)

```
[router]
ip=172.16.0.1
port=8728
user=admin
pwd=
[card]
prefix=152414
ignore=badi
[wifi]
ssid=badi***
[admin]
pwd=123456
```

### B.2 ROS 配置脚本 (**backup.rsc**)

```
may/05/2014 16:32:33 by RouterOS 6.12
software id = UELR-USC3
#
/interface bridge
add l2mtu=2290 name=bridge1
/interface wireless
set [find default-name=wlan1] band=2ghz-b/g/n channel-width=|
 20/40mhz-ht-below country=china disabled=no frequency=2462
 l2mtu=2290 \
 mode=ap-bridge wireless-protocol=802.11
/interface wireless security-profiles
set [find default=yes] supplicant-identity=MikroTik
/ip hotspot
add disabled=no idle-timeout=none interface=bridge1 name=server1
/ip hotspot profile
set [find default=yes] http-cookie-lifetime=30m login-by=cookie
```



```
,http-pap \
 use-radius=yes
/ip hotspot user profile
set [find default=yes] idle-timeout=none keepalive-timeout=2h \
 mac-cookie-timeout=3d shared-users=3
/ip pool
add name=dhcp ranges=192.168.220.100-192.168.220.150
/ip dhcp-server
add address-pool=dhcp disabled=no interface=bridge1 lease-time=30m
name=dhcp1
/interface bridge port
add bridge=bridge1 interface=wlan1
/ip address
add address=192.168.220.1/24 interface=wlan1 network=192.168.220.0
add address=192.168.120.13/24 interface=ether1 network=192.168.120.0
/ip dhcp-client
add dhcp-options=hostname,clientid interface=ether1
/ip dhcp-server network
add address=192.168.220.0/24 dns-server=\
 221.3.131.11,8.8.8.8,202.203.132.1,8.8.4.4 gateway=192.168.220.1
 netmask=\
 24
/ip dns
set servers=221.3.131.11,8.8.8.8
/ip firewall filter
add action=passthrough chain=unused-hs-chain comment=\
 "place hotspot rules here" disabled=yes
/ip firewall nat
add action=passthrough chain=unused-hs-chain comment=\
 "place hotspot rules here" disabled=yes to-addresses=0.0.0.0
add action=masquerade chain=srcnat out-interface=ether1 to-addresses=\
0.0.0.0
/ip route
add distance=1 gateway=192.168.120.1
/ip service
set www port=88
/ip upnp
set allow-disable-external-interface=no
```

```
/radius
add address=202.203.132.242 secret=radius service=\
 ppp,login,hotspot,wireless,dhcp timeout=6s
/system clock
set time-zone-name=Etc/GMT-8
/system clock manual
set time-zone=+08:00
/system identity
set name=CS-AP-2.4G-1
/system ntp client
set enabled=yes primary-ntp=216.171.120.36 secondary-ntp=64.4.10.33
/system scheduler
add interval=1d name="Reboot Router" on-event="/system \
 \nreboot" policy=\
 ftp,reboot,read,write,policy,test,winbox,password,sniff,sensitive
 ,api \
 start-date=jan/01/1970 start-time=06:00:00
```

## 附录 C 程序代码清单

### C.1 pyhook 函数引用

### C.2 主程序模块

```
1 # -*- coding: utf-8 -*-
2 #
3 # by oldj
4 # http://oldj.net/
5 #
6 import sys
7 import random
8 import ros
9 import myconf
10 import rasprint
11
12 keystr = ''
13 cardid = ''
14 PWDSTR='0123456789'
15
16 cardPrefix,ignore,ESSID,adminPwd=myconf.getAdminInfo()
17
18 def getNewPwd():
19 lst=list(PWDSTR)
20 slice = random.sample(lst, 6)
21 return ''.join(slice)
22
23 def checkUser(_user):
24 _user="".join(_user.split());
25 ret=''
26 #try:
27 # t=int(_user)
28 # return _user
29 #except:
30 # return ''
31 #print "The length of card",len(_user)
```

```
32
33 try:
34 if len(_user)==10:
35 # 检查卡号的前几位
36 l=len(cardPrefix)
37 #print "compare to ",_user[:6],cardPrefix
38 if _user[:6]!=cardPrefix: return ''
39 #print "success"
40 t=int(_user[-4:])
41 #ret= "%04x" %t
42 return str(t)
43 elif len(_user)==4:
44 int(_user,16)
45 return _user
46 else:
47 return ''
48 except:
49 return ''
50
51 def proc_user(cardInput=''):
52 #print "enter-----"
53 #2. 修改数据库中的用户状态
54 _user=str(cardInput)
55 #print "proc useing",_user,cardPrefix,ignore,ESSID
56 if _user == '': return
57 u=checkUser(_user)
58
59 if u == '':
60 return
61
62 current_user=u
63
64 newPwd=getNewPwd()
65 # 判断文本框中的用户是否已经存在
66
67 #showUserInfo
68 user_list=ros.getUserList(ignore)
69 #print "success getUserList"
70 #return
71 user_blocked_List=[]
72
```

```
73
74 uPosition = -1
75 for i in range(0,len(user_list)):
76 if user_list[i][1]==current_user:
77 uPosition = i
78 break
79 #print "uPos %d" %uPosition
80
81 if uPosition != -1:
82 # 存在, 则删除用户, 禁止使用
83 userid=user_list[uPosition][0]
84 user=user_list[uPosition][1]
85 ros.delUser(userid,user)
86 # 存在则修改密码
87 #ros.changePwd(userid,newPwd)
88 #winsound.Beep(4000,300)
89 else:
90 # 若不存在, 添加用户并启用
91 ros.addNewUser(current_user,newPwd)
92 #winsound.Beep(3000,300)
93 #myPrint.send_to_printer(ESSID,current_user,newPwd)
94 #print "This is printer",ESSID,current_user,newPwd
95 rasprint.raspout(ESSID,current_user,newPwd)
96
97 def getpass():
98 import termios, sys # man termios
99 fd = sys.stdin.fileno()
100 old = termios.tcgetattr(fd)
101 new = termios.tcgetattr(fd)
102 new[3] &= ~termios.ISIG & ~termios.ECHO # lflags
103 try:
104 termios.tcsetattr(fd, termios.TCSADRAIN, new)
105 cardid = raw_input()
106 finally:
107 termios.tcsetattr(fd, termios.TCSADRAIN, old)
108 return cardid
109
110 def main():
111 while 1:
112 keystring = getpass()
113 proc_user(keystring)
```

```

114 if keystring == "matriux":
115 break
116 print keystring
117
118 if __name__ == "__main__":
119 main()

```

### C.3 打印模块

```

1 from escpos import *
2
3 def raspout(essid="wifiname",username="joshua.w",password="matriux"):
4 #Epson = escpos.Escpos(0x0416,0x5011,0)
5 Epson = printer.Usb(0x0416,0x5011,0)
6 Epson.text("ESSID: %s\nusername: %s\npassword: %s"
7 % (essid,username,password))
8 #Epson.image("logo.gif")
9 Epson.barcode
10 Epson.barcode('\n1324354657687','EAN13',64,2,'','')
11 Epson.cut()
12
13 if __name__ == '__main__':
14 raspout()

```

### C.4 ros 模块

```

1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import sys, time, md5, binascii, socket, select
5 import threading
6 import myconf
7
8
9 class ApiRos:
10 "Routeros api"
11 def __init__(self):
12 self.sk = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13 r=myconf.getconf()

```

```
14 self.sk.connect((r[0],int(r[1])))
15 self.login(r[2], r[3]);
16 self.currenttag = 0
17
18 def login(self, username, pwd):
19 for repl, attrs in self.talk(["/login"]):
20 chal = binascii.unhexlify(attrs['=ret'])
21 md = md5.new()
22 md.update('\x00')
23 md.update(pwd)
24 md.update(chal)
25 self.talk(["/login", "=name=" + username,
26 "=response=00" + binascii.hexlify(md.digest())])
27 def talk(self, words):
28 if self.writeSentence(words) == 0: return
29 r = []
30 while 1:
31 i = self.readSentence();
32 if len(i) == 0: continue
33 reply = i[0]
34 attrs = {}
35 for w in i[1:]:
36 j = w.find('=', 1)
37 if (j == -1):
38 attrs[w] = ''
39 else:
40 attrs[w[:j]] = w[j+1:]
41 r.append((reply, attrs))
42 if reply == '!done': return r
43
44 def writeSentence(self, words):
45 ret = 0
46 for w in words:
47 self.writeWord(w)
48 ret += 1
49 self.writeWord('')
50 return ret
51
52 def readSentence(self):
53 r = []
54 while 1:
```

```
55 w = self.readWord()
56 if w == '': return r
57 r.append(w)
58
59 def writeWord(self, w):
60 #print "<<< " + w
61 self.writeLen(len(w))
62 self.writeStr(w)
63
64 def readWord(self):
65 ret = self.readStr(self.readLen())
66 #print ">>> " + ret
67 return ret
68
69 def writeLen(self, l):
70 if l < 0x80:
71 self.writeStr(chr(l))
72 elif l < 0x4000:
73 l |= 0x8000
74 self.writeStr(chr((l >> 8) & 0xFF))
75 self.writeStr(chr(l & 0xFF))
76 elif l < 0x200000:
77 l |= 0xC00000
78 self.writeStr(chr((l >> 16) & 0xFF))
79 self.writeStr(chr((l >> 8) & 0xFF))
80 self.writeStr(chr(l & 0xFF))
81 elif l < 0x10000000:
82 l |= 0xE0000000
83 self.writeStr(chr((l >> 24) & 0xFF))
84 self.writeStr(chr((l >> 16) & 0xFF))
85 self.writeStr(chr((l >> 8) & 0xFF))
86 self.writeStr(chr(l & 0xFF))
87 else:
88 self.writeStr(chr(0xF0))
89 self.writeStr(chr((l >> 24) & 0xFF))
90 self.writeStr(chr((l >> 16) & 0xFF))
91 self.writeStr(chr((l >> 8) & 0xFF))
92 self.writeStr(chr(l & 0xFF))
93
94 def readLen(self):
95 c = ord(self.readStr(1))
```



```
96 if (c & 0x80) == 0x00:
97 pass
98 elif (c & 0xC0) == 0x80:
99 c &= ~0xC0
100 c <<= 8
101 c += ord(self.readStr(1))
102 elif (c & 0xE0) == 0xC0:
103 c &= ~0xE0
104 c <<= 8
105 c += ord(self.readStr(1))
106 c <<= 8
107 c += ord(self.readStr(1))
108 elif (c & 0xF0) == 0xE0:
109 c &= ~0xF0
110 c <<= 8
111 c += ord(self.readStr(1))
112 c <<= 8
113 c += ord(self.readStr(1))
114 c <<= 8
115 c += ord(self.readStr(1))
116 elif (c & 0xF8) == 0xF0:
117 c = ord(self.readStr(1))
118 c <<= 8
119 c += ord(self.readStr(1))
120 c <<= 8
121 c += ord(self.readStr(1))
122 c <<= 8
123 c += ord(self.readStr(1))
124 return c
125
126 def writeStr(self, str):
127 n = 0;
128 while n < len(str):
129 r = self.sk.send(str[n:])
130 if r == 0: raise RuntimeError, "connection closed by remote end"
131 n += r
132
133 def readStr(self, length):
134 ret = ''
135 while len(ret) < length:
136 s = self.sk.recv(length - len(ret))
```

```
137 if s == '': raise RuntimeError, "connection closed by remote end"
138 ret += s
139 return ret
140
141 def getUserList(ignoreAccount=''):
142 apiros = ApiRos();
143 #apiros.login('admin', '');
144 inputsentence=['/ip/hotspot/user/print']
145 inputsentence.append('=.proplist=.id,name,password,disabled')
146 apiros.writeSentence(inputsentence)
147 il=len(ignoreAccount)
148 ret=[]
149 while 1:
150 x = apiros.readSentence()
151 if x[0] == '!done': break
152 del x[0]
153 #print ignoreAccount
154 #print il
155 ipre = x[1].split('=')[2][:il]
156 if ipre == ignoreAccount:
157 # print "ignore"
158 continue
159 for i in range(len(x)):
160 s=x[i].split('=')
161 x[i]=s[2]
162 #print x[6]
163 if x[3] == 'false':
164 x[3]=u' 启用'
165 else:
166 x[3]=u' 禁用'
167 x[2]='----'
168
169 ret.append(x)
170
171 mySort2(ret,1)
172 return ret
173
174 def mySort2(A, indexLie=0):
175 NLie = len(A[0])
176 if indexLie<0:
177 indexLie=0
```

```

178 elif indexLie>=NLie :
179 indexLie = 0
180 if indexLie!= 0:
181 for H in range(len(A)):
182 (A[H][0],A[H][indexLie]) = (A[H][indexLie],A[H][0])
183 A.sort()
184 if indexLie!= 0:
185 for H in range(len(A)):
186 (A[H][0],A[H][indexLie]) = (A[H][indexLie],A[H][0])
187
188 def addNewUser(userid,newPwd):
189 apiros = ApiRos()
190 inputsentence=['/ip/hotspot/user/add']
191 inputsentence.append('=name='+userid)
192 inputsentence.append('=password='+newPwd)
193 inputsentence.append('=server=hotspot1')
194 inputsentence.append('=profile=default')
195 inputsentence.append('=disabled=false')
196
197 apiros.writeSentence(inputsentence)
198 apiros.readSentence()
199
200 def changePwd(userid,newPwd):
201 apiros = ApiRos()
202 inputsentence=['/ip/hotspot/user/set']
203 inputsentence.append('=.id='+userid)
204 inputsentence.append('=password='+newPwd)
205 apiros.writeSentence(inputsentence)
206 apiros.readSentence()
207
208 def delUser(userid,user):
209 apiros = ApiRos()
210 inputsentence=['/ip/hotspot/user/remove']
211 inputsentence.append('=.id='+userid)
212 apiros.writeSentence(inputsentence)
213 apiros.readSentence()
214 #tickFromLine
215 inputsentence=['/ip/hotspot/active/print']
216 inputsentence.append('=.proplist=.id,user')
217 inputsentence.append('?user='+user)
218 apiros.writeSentence(inputsentence)

```

```
219 #print "user:" + user
220 ret=[]
221 while 1:
222 x = apiros.readSentence()
223 #ret=apiros.readSentence()
224 if x[0] == '!done': break
225 ret.append(x)
226
227 for x in ret:
228 t=x.index("=user="+user)
229 #if t>0:
230 acid=x[t-1]
231 acid=acid.split('=')[2]
232 #print "acid remove:" + acid
233 inputsentence=['/ip/hotspot/active/remove']
234 inputsentence.append("=.id="+acid)
235 apiros.writeSentence(inputsentence)
236 while 1:
237 xx = apiros.readSentence()
238 if xx[0] == '!done': break
239
240
241
242 def enabledUser(userid,newPwd):
243 apiros = ApiRos();
244 inputsentence=['/ip/hotspot/user/set']
245 inputsentence.append('=.id='+userid)
246 inputsentence.append('=password='+newPwd)
247 inputsentence.append('=disabled=false')
248 apiros.writeSentence(inputsentence)
249 apiros.readSentence()
250
251 def activeUserInfo(user,ulist):
252 ret=[]
253 for t in range(0,len(ulist)):
254 if ulist[t][2]=='=user='+user:
255 ret.append(ulist[t])
256
257 return ret
258
259 #while 1:
```

```
260 # x = apiros.readSentence()
261 # if x[0] == '!done': break
262 # ret.append(x)
263 #return ret
264
265 #apiros.readSentence()
266 def listActiveUser():
267 apiros = ApiRos();
268 inputsentence=['/ip/hotspot/active/print']
269 inputsentence.append('=.proplist=.id,user,address,mac-address,uptime
270 ,idle-timeout')
271 apiros.writeSentence(inputsentence)
272 #print "Treading start"
273 ret=[]
274 while 1:
275 x = apiros.readSentence()
276 if x[0] == '!done': break
277 ret.append(x)
278
279 #self.acu_list = ret
280 #print self.acu_list
281 return ret
282
283
284 def disabledUser(userid,user):
285 apiros = ApiRos();
286 inputsentence=['/ip/hotspot/user/set']
287 inputsentence.append('=.id='+userid)
288 inputsentence.append('=disabled=true')
289 apiros.writeSentence(inputsentence)
290 apiros.readSentence()
291 #tickFromLine
292 inputsentence=['/ip/hotspot/active/print']
293 inputsentence.append('=.proplist=.id,user')
294 apiros.writeSentence(inputsentence)
295 #print "user:" + user
296 ret=[]
297 while 1:
298 x = apiros.readSentence()
299 if x[0] == '!done': break
300 ret.append(x)
```

```
301
302 for x in ret:
303 t=x.index("=user="+user)
304 if t>0:
305 acid=x[t-1]
306 acid=acid.split('=')[2]
307 #print "acid remove:"+acid
308
309 inputsentence=['/ip/hotspot/active/remove']
310 inputsentence.append("=.id="+acid)
311 apiros.writeSentence(inputsentence)
312 while 1:
313 xx = apiros.readSentence()
314 if xx[0] == '!done': break
315 break
316
317
318
319 def main():
320 apiros = ApiRos();
321 #apiros.login(sys.argv[2], sys.argv[3]);
322
323 inputsentence = []
324
325 while 1:
326 #r = select.select([s, sys.stdin], [], [], None)
327 #if s in r[0]:
328 # something to read in socket, read sentence
329
330
331 #if sys.stdin in r[0]:
332 # read line from input and strip off newline
333 l = sys.stdin.readline()
334 l = l[:-1]
335
336 # if empty line, send sentence and start with new
337 # otherwise append to input sentence
338 if l == '':
339 apiros.writeSentence(inputsentence)
340 inputsentence = []
341 else:
```

```
342 inputsentence.append(l)
343
344 x = apiros.readSentence()
345 print x
346
347 if __name__ == '__main__':
348 main()
```

## C.5 readcard 模块

```
1 #!/usr/bin/env python
2 -*- coding: utf-8 -*-
3
4 import ctypes
5 from ctypes import *
6 #import thread,sys,time
7
8 dll=ctypes.windll.LoadLibrary('btlock55L.dll')
9
10 global roomid
11 def getCardNum():
12 bPort=1
13 strID='QTJDJHC'
14 strPWD='201108'
15
16 iCardNo=create_string_buffer(10)
17 iGuestSN=create_string_buffer(10)
18 iGuestIdx=create_string_buffer(3)
19 strDoorID=create_string_buffer(6)
20 strSuitDoor=create_string_buffer(12)
21 strPubDoor=create_string_buffer(8)
22 strBeginTime=create_string_buffer(14)
23 strEndTime=create_string_buffer(14)
24
25 dll.Read_Guest_Card.argtypes = [ctypes.c_int,ctypes.c_char_p,
26 ctypes.c_char_p,ctypes.c_char_p,ctypes.c_char_p,
27 ctypes.c_char_p,ctypes.c_char_p,ctypes.c_char_p,
28 ctypes.c_char_p,ctypes.c_char_p,ctypes.c_char_p]
29 dll.Read_Guest_Card.restypes = ctypes.c_int
30
```

```

31 result=dll.Read_Guest_Card(bPort, strID, strPWD, iCardNo,
32 iGuestSN, iGuestIdx, strDoorID, strSuitDoor,
33 strPubDoor, strBeginTime, strEndTime)
34 if result == 0:
35 return strDoorID.value
36 else:
37 return ''
38
39 # def myThread():
40 # global roomid
41 # while 1:
42 # t = getCardNum()
43 # if roomid != t:
44 # roomid =t
45 # print roomid
46 # time.sleep(2)
47
48 # def main():
49 # global roomid
50 # try:
51 # roomid=''
52 # thread.daemon=True
53 # thread.start_new_thread(myThread,())
54 # except Exception,e:
55 # pass
56
57
58 # if __name__=='__main__':
59 # #chk_card()
60 # main()

```

## C.6 myconf 模块

```

1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 import ConfigParser
4 def getconf():
5 ret=[]
6 config = ConfigParser.SafeConfigParser()
7 config.read("conf.cfg")

```



```
8 ret.append(config.get("router", "ip"))
9 ret.append(config.get("router", "port"))
10 ret.append(config.get("router", "user"))
11 ret.append(config.get("router", "pwd"))
12 return ret
13
14 return r
15
16 def getAdminInfo():
17 ret=[]
18 config = ConfigParser.SafeConfigParser()
19 config.read("conf.cfg")
20 ret.append(config.get("card", "prefix"))
21 ret.append(config.get("card", "ignore"))
22 ret.append(config.get("wifi", "ssid"))
23 ret.append(config.get("admin", "pwd"))
24 return ret
25
26 def getCardPrefix():
27 ret=''
28 config = ConfigParser.SafeConfigParser()
29 config.read("conf.cfg")
30 ret=config.get("card", "prefix")
31 return ret
32 def getESSID():
33 ret=''
34 config = ConfigParser.SafeConfigParser()
35 config.read("conf.cfg")
36 ret=config.get("wifi", "ssid")
37 return ret
38
39 def __(_Name):
40 if _Name == 'user':
41 return u' 帐号'
42 elif _Name=='address':
43 return u'IP 地址'
44 elif _Name=='mac-address':
45 return u' 物理地址'
46 elif _Name=='uptime':
47 return u' 上网时间'
48 elif _Name=='idle-timeout':
```

```
49 return u' 空闲时间'
50 elif _Name=='.id':
51 return 'ID'
52
```