JS9331 开发板
OpenWrt 入门教程

v1.20(2018.11.18)

杭州卓钛科技有限公司

网站：www.zhuotk.com

前言

这份《JS9331 开发板 OpenWrt 入门教程》是卓钛科技编写的，主要介绍基于 JS9331 开发板的 OpenWrt 入门学习。里面介绍了有关 OpenWrt、嵌入式 linux 的一些知识，同时还有一些项目实例，适用于 OpenWrt、linux 的初学者。看完本教程，你将学会如何配置 OpenWrt 系统，利用 OpenWrt 做一些好玩高级的应用以及开发简单的应用程序、驱动程序。由于时间仓促以及作者水平有限,本教程错漏缺点在所难免,希望读者批评指正，提出你们的需求和建议。

目录

1. OpenWrt 介绍

1.1. 什么是 OpenWrt

OpenWrt（官网 www.openwrt.org）可以被描述为一个嵌入式的 Linux 发行版，目前常用在路由器上，但是作为基于 linux 系统的它，其实可以做更多的事情。

它是一个高度模块化、高度自动化的嵌入式 Linux 系统，拥有强大的网络组件和扩展性，还可被用于工控设备、电话、小型机器人、远程监控、智能家居以及 VOIP 设备中。

它不同于其他许多用于路由器的发行版（主流路由器固件有 dd-wrt,tomato,OpenWrt 三类），它是一个从零开始编写的、功能齐全的、容易修改的路由器操作系统。实际上，这意味着您能够使用您想要的功能而不加进其他的累赘，而支持这些功能工作的 linux kernel 又远比绝大多数发行版来得新（linux 内核中很多源代码都是由 OpenWrt 社区提供的）。

1.2. 为什么学 OpenWrt

你不需要对 MIPS 处理器有很深入的了解 ,也不用懂得如何去设计一个 ARM 或 MIPS 处理器专用的 linux 内核 ,因为这些移植工作在 OpenWrt 里已有人为你做好, 你只需懂得如何安装和使用 OpenWrt 就行了,不过你也可以去 http://www.linux-mips.org 找到相关的资料。如果你对 Linux 系统有一定的认识 , 并想学习或接触嵌入式 Linux 的话 , OpenWrt 很适合你 ,你将学会一些无线路由器的基本知识 ,以及一般嵌入式 Linux 的开发过程。但凡做过或者了解过嵌入式开发的人 ,都知道无论是 ARM, PowerPC 或 MIPS 的处理器 ,都必需经过以下的开发过程：

1、创建 Linux 交叉编译环境
2、建立 Bootloader
3、移植 Linux 内核
4、建立 Rootfs (根文件系统)
5、安装驱动程序
6、安装软件

采用上面传统的方法进行嵌入式开发，费时费力，但是可以你通过 OpenWrt 快速构建一个应用平台，OpenWrt 从交叉编译器，到 linux 内核，再到文件系统甚至 bootloader 都整合在了一起，形成了一个 SDK 环境。其多达 3000 多种软件包（数量还在增加），囊括从工具链(toolchain)，到内核(linux kernel)，到软件包(packages)，再到根文件系统(rootfs)整个体系，使得用户只需简单的一个 make 命令即可方便快速地定制一个具有特定功能的嵌入式系统来制作固件，大大减少了嵌入式软件开发的工序。当你熟悉这些嵌入式 Linux 的基本开发流程后你不再局限于 MIPS 处理器和无线路由器 ,你可以尝试在其它处理器 ,或者非无线路由器的系统移植嵌入式 Linux, 定制合适自己的应用软件 , 并建立一个完整的嵌入式产品。

OpenWrt 的成功之处还在于它的文件系统是可写的，开发者无需在每一次修改后重新编译系统，并且可以像 PC 机上的 linux 系统一样，用命令安装一些安装包，不用手动配置，这些都令它更像一个小型的 Linux 电脑系统。

1.3. OpenWrt 版本发展史

OpenWrt 项目由 2004 年 1 月开始 ,第一个版本是基于 Linksys 提供的 GPL 源码及

uclibc 中的 buildroot 项目 ,这个版本称为“stable”版 ,在网上至今仍有很多项目用这个版本 ，较为有名 Freifunk-Firmware 和 Sip@Home。到了 2005 年初,一些新的开人员加入了这项目 ，几个月后他们释出了第一个“experimental”版本 ,这和以前版本不同的是 ，这版本差不多完全舍弃了 Linksys 的 GPL 源码 ，使用了 buildroot2 作为核心技术 ,将 OpenWrt 完全模块化，OpenWrt 使用 Linux 正式发行的核心源码（ 2.4.3x ）加上了一些补丁和网络驱动 ，开发队伍更为OpenWrt添加了许多免费的工具，你可以直拉把 Image 写入Flash(mtd)里面，设定无线功能 和 VLAN 交换功能，这个版本名为“White Russian”，而 1.0 版本大概 于 2005 年底公布。2006-2009 年是 OpenWrt 迅猛发展的时间，这个时候的 OpenWrt 所支持的平台不仅仅限于 broadcom 的 SoC，它开始支持 Intel IXP 为首的 ARM 平台，以及 PowerPC,MIPS 24K R2,x86 等各种新平台。在软件应用上出现了以 LuCi 跟 Webif 为首的 UI 以及各种更新软件包。

由于 openwrt 项目组策略改变，现在推出了一个基于原来 openwrt 的新版本“LEDE”，绝大部分开发人员也转移到了 LEDE 上面进行开发，所以我们的教程以后将基于 LEDE 进行编写。读者可以到“https://www.openwrt.org/”了解更多的资料。目前 openwrt 和 LEDE 两者用法几乎没有区别，只是 LEDE 拥有更多的开发人员，更新速度更快，功能以后会更多。

版本时间轴

版本号	发布日期	代号
测试版本		
	持续更新	LEDE
稳定版本		
15.05	2015 年 9 月	Chaos Calmer
14.07	2014 年 10 月	Barrier Breaker
12.09	2013 年 4 月	Attitude Adjustment
10.03.1	2011 年 12 月	Backfire
10.03	2010 年 4 月	Backfire
8.09.2	2010 年 1 月	Kamikaze
8.09.1	2009 年 6 月	Kamikaze
8.09	2008 年 9 月	Kamikaze
7.09	2007 年 9 月	Kamikaze
7.07	2007 年 7 月	Kamikaze
7.06	2007 年 6 月	Kamikaze
0.9	2007 年 1 月	White Russian0.9
0.x	2006 年 11 月	White RussianRC6
0.x	2006 年 3 月	White RussianRC5
0.x	2005 年 11 月	White RussianRC4
0.x	2005 年 9 月	White RussianRC3
0.x	2005 年 7 月	White RussianC2
0.x	2005 年 6 月	White RussianRC1
0.x	2005 年 2 月	Before experimental

2. 开发学习概述

在这里，我们将采用 OpenWrt 的应用场景大致分为以下两种。

一种是直接使用 OpenWrt 现有的固件和安装包，做一些应用和配置。这种应用情景对人员的技术水平要求不是很高，一般对 linux 有了解的，甚至是未接触过 linux 的人员也可以。

另一种是需要对 OpenWrt 进行一些功能的定制，以实现现有 OpenWrt 版本未实现的功能或修改现有功能的情况，这种情况要求开发人员有一定的 linux 基础知识和开发能力。

以下“入门开发篇”主要针对 OpenWrt 初级用户，“深入开发篇”主要针对希望深入学习 OpenWrt 的用户。

3. 入门应用篇

在本篇中，我们并不介绍如何编译 OpenWrt、敲命令等，因为这些知识一开始就要求初学者掌握似乎比较困难，对于只是想用 OpenWrt 现有功能的用户来说，这些知识都是没必要的。下面我们就以 JS9331 开发板为平台，开始学习如何使用 OpenWrt。

3.1. 进入 OpenWrt 系统

首先给 JS9331 开发板上电，连接网线。开发板启动完成后，在浏览器地址栏中输入 192.168.1.251，在登录界面用户名和密码都输入“root”，登录 OpenWrt 系统（如不清楚如何启动开发板，请查看《JS9331 开发使用手册》“开机测试”一节，其中有更详细的启动步骤，这里不再赘述）。

登录后的 OpenWrt 网页界面，如下图所示

在登录 OpenWrt 后的首个页面是“总览”页面。在这里，我们可以看到各种系统信息，包括主机名、linux 内核版本号、固件版本、内存大小等。

3.2. 系统菜单

通过 OpenWrt 配置页面的系统顶部菜单，我们可以配置 OpenWrt 一部分功能（用户还可以通过串口、ssh、telnet 进入系统 shell 进行更高级的系统配置，这些将在“深入开发篇”中进行介绍）。

系统顶部菜单栏如下图所示。

其中有“状态”、“系统”、“网络”、“退出”这几项，这些是系统的初始菜单项，有时新增一些功能，会增加一些新的菜单项。

下面我们就按照这几个主菜单项的顺序依次介绍它们，以及其中部分的各子菜单项功能。

3.2.1. “状态”菜单项

“状态”菜单项包含以下子菜单项。

1) “总览”子菜单，
显示的即进入 OpenWrt 系统后显示的第一页面，前面已经介绍过了。
2) “防火墙”子菜单
显示的是根据菜单项“网络”->“防火墙”中设置的防火墙规则，当前网络的流量情况。
3) “路由表”子菜单
显示的是当前设备（JS9331 开发板）的路由表。
4) “系统日志”子菜单
显示的是系统守护进程打印的一些信息，当系统出错时，用户可以根据里面的一些提示也许能找到问题。

```
Sat Jun 20 13:41:13 2015 kern.info kernel: [   28.340000] br-lan: port 2(wlan0) entered forwarding state
Sat Jun 20 13:41:13 2015 kern.info kernel: [   28.340000] br-lan: port 2(wlan0) entered forwarding state
Sat Jun 20 13:41:13 2015 kern.info kernel: [   28.350000] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
Sat Jun 20 13:41:13 2015 daemon.notice netifd: Network device 'wlan0' link is up
Sat Jun 20 13:41:15 2015 kern.info kernel: [   30.340000] br-lan: port 2(wlan0) entered forwarding state
Sat Jun 20 13:42:01 2015 daemon.warn odhcpd[783]: DHCPV6 SOLICIT IA_NA from 0001000118042685308 5a99a55b6 on br-lan
```

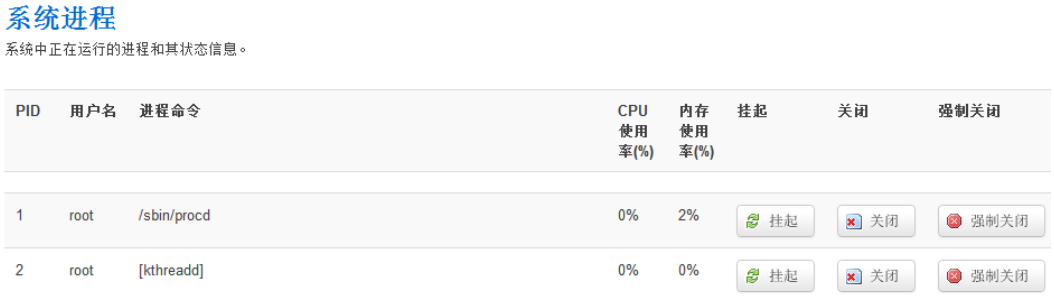
5) “内核日志”子菜单

显示的是内核的打印信息（同串口的内核打印信息），同样，当系统出错时，用户可以根据里面的一些提示信息也许能找到问题。

```
[   26.010000] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
[   26.020000] IPv6: ADDRCONF(NETDEV_CHANGE): br-lan: link becomes ready
[   26.820000] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[   26.830000] device wlan0 entered promiscuous mode
[   28.000000] br-lan: port 1(eth1) entered forwarding state
[   28.340000] br-lan: port 2(wlan0) entered forwarding state
[   28.340000] br-lan: port 2(wlan0) entered forwarding state
[   28.350000] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[   30.340000] br-lan: port 2(wlan0) entered forwarding state
```

6) “系统进程”子菜单

显示的是 OpenWrt（linux）系统中正在运行的进程和其状态信息，有点像 windows 任务管理器。用户可以决定是否关闭挂起进程，如果用户不熟悉系统，不推荐做这些操作。



7) “实时信息”子菜单，用图表显示系统当前的负载、流量等情况。

3.2.2. “系统”菜单项

1) “系统”子菜单

显示的是一些系统的基本设置，包括“主机名”、“系统语言”、“系统主题”等。

2) “管理权”子菜单

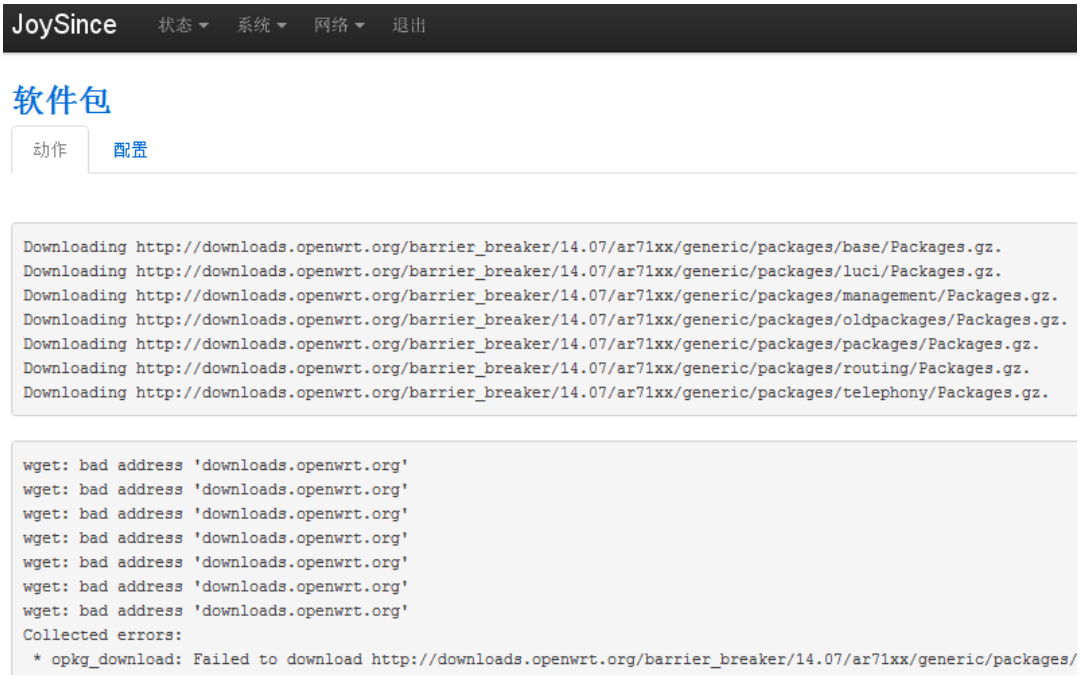
可以设置系统的管理员密码（登录密码）。ssh 访问权限也在这里设置。

3) “软件包”子菜单

我们可以查看、安装、卸载安装包，OpenWrt 的软件功能拓展就可以在这里实现。下面我们着重介绍一下这个页面。该页面如下图所示。

我们可以在“已安装软件包”中，查看已安装的软件包列表。如果用户想安装新的软件包，需要安装如下步骤进行。

1) 需要点击图中的“刷新列表”按钮，但此时，可能会出现下图的提示



出现以上提示是因为系统（板子）没设置好网关、DNS，导致开发板无法和外网进行通信。用户可以到系统顶部菜单“网络”->“接口”->“LAN”->“修改”->“基本设置”里面修改“IPv4 网关”和“DNS”为用户所在网络的路由器 IP 地址（这里假设为 192.168.1.1）。如下图所示。

最后别忘了点击当前页面下方的“保存&应用”。

这样，开发板就可以和外网进行通信了。

2) 然后再点击“可用软件包”，系统会从“配置”中设置好的地址下载软件包列表。
3) 选择需要安装的软件包，点击“安装”。随后安装自动完成

注意：由于 OpenWrt 网站服务器在国外，访问其页面时，会经常出现无法访问的情况，这时可能无法获取到软件安装包，这里介绍几种解决方法

a.读者可以登录 downloads.lede-project.org 进行安装包的下载，如果还是下载不了，可以挂载 VPN 再尝试下载或者将安装包的下载链接复制到迅雷里面进行下载。

b.自行编译安装包进行安装（将在“深入开发篇”进行介绍）。

4) “启动项”子菜单

显示的是开机启动的脚本。用户也可以在底部添加需要开机启动的指令。

5) “计划任务”子菜单

可以添加定时任务。

6) “LED 配置”子菜单

我们可以配置板子上任意 LED 灯所指示的内容。下面我们来仔细介绍一下 OpenWrt 的 LED 配置的功能。LED 配置页面如下图所示。

上图中的

a)“名字”我们可以任意设置，但是最好和 LED 灯指示的内容一致，以便区别记忆。

b)“LED 名称”是固件编译时指定好的，和开发板硬件已经绑定了（用户可查看开发板配套资料中的 js9331_openwrt.patch 搜索相应字段，学习如何添加修改），页面中无法修改其名称，但是可以从多个选项中选择。

c)“触发”是表示 LED 由什么事件来触发。有多种选项可供用户选择，如“timer（定时器）”、“netdev（网络设备）”等。具体如何配置，用户可以参考开发板的 LED 配置页面。

wifi 指示灯的配置，如下图所示

系统运行正常指示，如下图所示

具体运行效果，请用户查看开发板实物。

7) “备份/升级”子菜单

这也是一个比较重要的页面。这里我们可以进行 OpenWrt 的系统配置备份、固件烧写、系统恢复、恢复出厂设置等操作。下面我们来仔细介绍一下该页面的功能，如下图所示。

a) “生成备份”按钮

点击该按钮，系统会自动检测我们修改过的文件，并将其打包提示下载。但是 OpenWrt 默认是比较“/etc”目录下面的文件进行备份，有时我们修改了其他目录下的文件，这时我们就需要完整的备份 OpenWrt 系统。

点击页面中的“配置”，在配置栏中增加“/overlay”，然后点击“提交”。如下图所示。

再回到刚才的页面，点击“生成备份”下载备份文件进行完整的备份。

b) “执行复位”按钮。

该按钮用于恢复出厂设置

c) “恢复配置”栏。

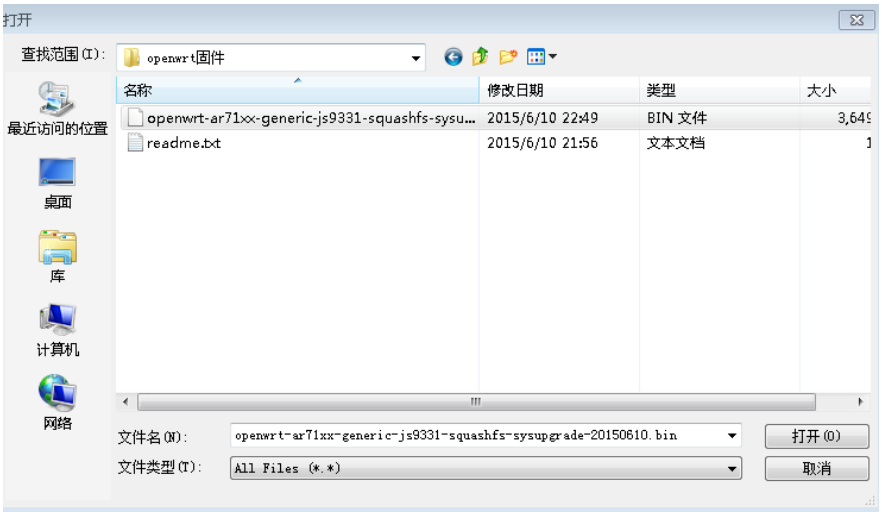
选择之前生成的备份文件，然后再点击“上传备份”，系统即可恢复到备份时的状态。

d) “刷写新固件栏”

选择页面下的“固件文件”一栏“选择文件”按钮，如下图红圈所示。

提示：点击如果勾选“保存配置”，则系统在刷新固件后，系统的会根据“配置”里面的路径，保持里面的文件不变，系统会更新内核。如果去掉“保存配置”，刷新后，将是一个全新的系统。此时也可以利用之前的“恢复配置”恢复系统。

在随后跳出的文件选择页面中，选中要上传的固件，并点击“打开”，如下图所示



点击“刷写固件”，如果固件正确，则会出现下图提示

点击上图右下角“执行”，开始烧写固件。等待几分钟后系统刷写完成，系统自动重启（注意 JS9331 恢复出厂设置后的 IP 地址是 192.168.1.251，OpenWrt 官方的是 192.168.1.1）。

8) “重启”子菜单

在该页面中可以重启系统。

3.2.3. “网络”菜单项

1) “接口”子菜单

在该页面中，我们可以设置开发板的 IP 设置，WAN 口的 PPPOE 拨号等。

2) “无线”子菜单

在该页面中，可以配置开发板的 wifi SSID、加密方式、发射功率等一些无线参数。

3) “DHCP/DNS”子菜单

可以设置 DHCP 和 DNS 一些配置，比如 DNS 转发等。

4) “主机名”子菜单

可以设置主机名和 IP 的对应关系。

5) “静态路由”子菜单

可以设置开发板的路由表。

6) “防火墙”子菜单

可以设置接口的过滤规则、端口转发等。

7) “网络诊断”子菜单

利用在这个页面中的“ping”功能，可以判断开发板是否能和外网进行通信。

3.2.4. “退出”菜单项

点击该菜单后，将直接退出登录状态。

3.3. 玩转 OpenWrt 高级功能

3.3.1. 电脑和开发板互传文件

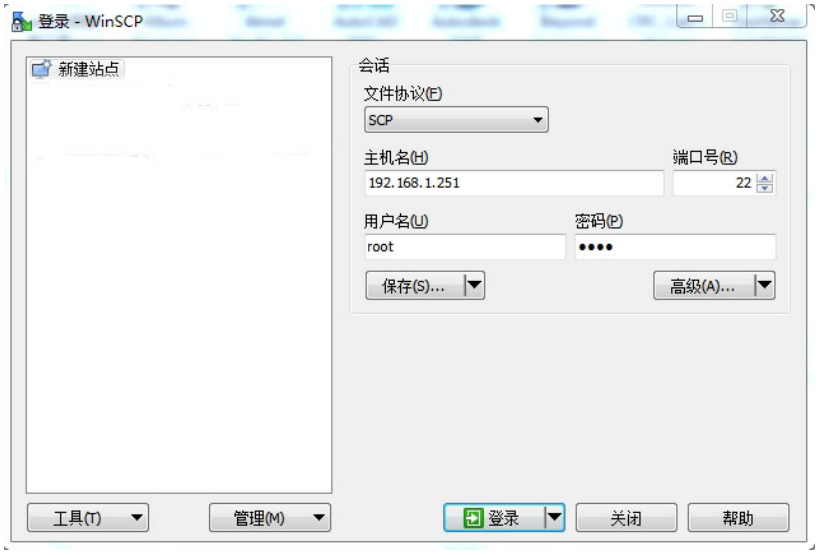
在开发使用开发板的过程中，会经常需要向开发板传输文件的功能，这里介绍 openwrt（linux）里面几种常用的文件传输方法。

3.3.1.1. 用 scp 协议传输文件

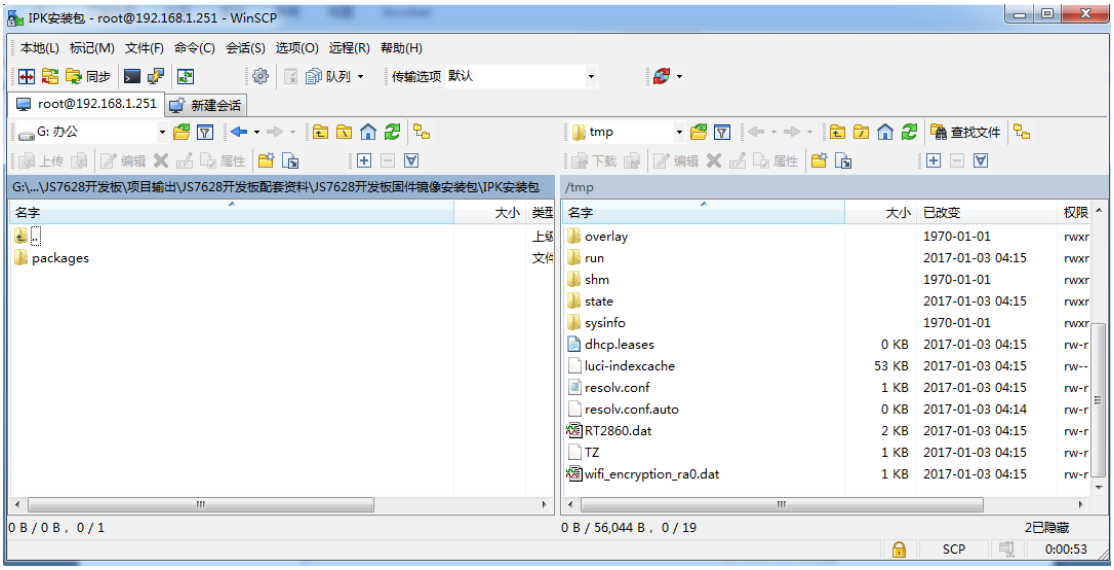
由于 openwrt 默认开启 scp 服务器，所以我们不需要在开发板上安装其他软件，即可用 scp 协议连接开发板传输文件。

在 windows 下可以用“winscp”软件进行 scp 协议文件的传输，该软件在“JS9331 开发板配套资料\开发工具软件\ winscp_V5.7.1.5235_setup.exe”请读者自行安装，安装完成后，点击图标 

进入 winscp 界面，如下图所示



输入和上图一致的配置（用户名和密码都是 root），然后点击“登录”。出现类似下图



上图表示成功连接开发板，可以互传文件了。

在 ubuntu(linux)下可以用 scp 命令进行文件的传输。

scp /path/local_filename username@servername:/path　上传本地文件到服务器，例如

scp ./test123.txt　root@192.168.1.251:/tmp　把本机当前目录下的 test.txt 文件上传到 192.168.1.251 开发板的/tmp 目录中

scp username@servername:/path/filename /tmp/local_destination　从服务器下载文件，例如

scp root@192.168.1.251:/tmp/test.txt ./　把 192.168.1.251 上的/tmp/test.txt 文件下载到电脑的当前目录下

3.3.1.2. 用 NFS 服务传输文件

NFS（Network File System）是一种在 linux 里面支持的比较好的网络文件系统。在 linux 里面，用户可以通过 NFS 客户端透明地读写位于远端 NFS 服务器上的文件，就像访问本地文件一样，非常方便。下面介绍一下如何实现开发板和 ubuntu 里面通过 NFS 文件系统传输文件，步骤如下

1) ubuntu 安装 NFS 服务

ubuntu 执行以下命令

sudo apt-get install nfs-kernel-server　　#安装 NFS 服务

接着用管理员权限（sudo）打开 ubuntu 中/etc/exports 文件，在末尾添加

/home/user/ *(rw,sync,no_root_squash)

类似如下图所示

```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/home/robinson/ *(rw,sync,no_root_squash)
```

nfs 允许挂载的目录及权限，在文件/etc/exports 中进行定义，各字段含义如下：

"/home/robinson"：要共享的目录，读者可以设置自己想要共享的目录,比如 openwrt 固件目录等，一定要是实际存在的目录

"*" ：允许所有的网段访问

"rw" ：具有读写权限

"sync"：资料同步写入内在和硬盘

"no_root_squash"：nfs 客户端共享目录使用者权限

完成以上修改后，执行命令

sudo /etc/init.d/nfs-kernel-server restart　　#重启 NFS 服务，使之前配置生效

showmount -e　　#查看修改是否生效

如下图所示

```
robinson@robinson-virtual-machine:~$ sudo /etc/init.d/nfs-kernel-server restart
[ ok ] Restarting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
robinson@robinson-virtual-machine:~$ showmount -e
Export list for robinson-virtual-machine:
/home/robinson *
```

2) 开发板挂载 NFS 文件系统

首先，安装 nfs 文件系统支持安装包"kmod-fs-nfs"，接着开发板执行命令

mkdir /mnt/nfs　　#建立一个名为"nfs"的目录，用于挂载 ubuntu 共享的 nfs 文件夹

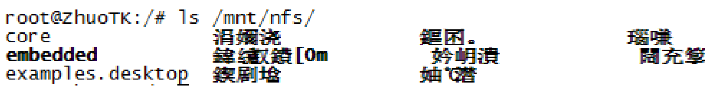
mount -t nfs -o nolock 192.168.1.8:/home/robinson/ /mnt/nfs　#挂载 ubuntu 共享的 nfs 文件夹，"192.168.1.8"为作者的 ubuntu IP，读者请自行修改为自己的 ubuntu IP。

3) 测试挂载结果

挂载成功后，开发板执行

ls /mnt/nfs　　#查看文件列表

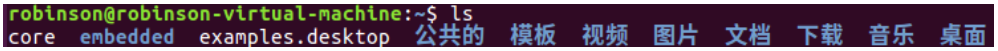
结果如下图所示



ubuntu 上执行

ls /home/robinson

结果如下图所示



从上面可知，这两个文件夹是完全同步的，现在读者可以在开发板上任意的修改 ubuntu 上共享的文件内容和直接运行编译好的程序了。

提示：由于挂载的是 NFS 远程文件系统，在 ubuntu 关机前，前先在开发板上执行

umount /mnt/nfs　#卸载 NFS 文件夹

否则有可能造成开发板访问这个目录的时候，造成开发板等待 NFS 响应而卡死。

3.3.1.3.　用 FTP 服务传输文件（搭建 FTP 服务器）

FTP 是 File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”。常用于 Internet 上的控制文件的双向传输。与大多数 Internet 服务一样，FTP 也是一个客户机/服务器系统。这里介绍一下如何实现开发板和 ubuntu、windows 通过 FTP 协议传输文件，步骤如下

1)　开发板安装“vsftpd”安装包

参考“安装 IPK 包”一节，搜索并安装“vsftpd”安装包。

2)　配置开启 FTP 服务

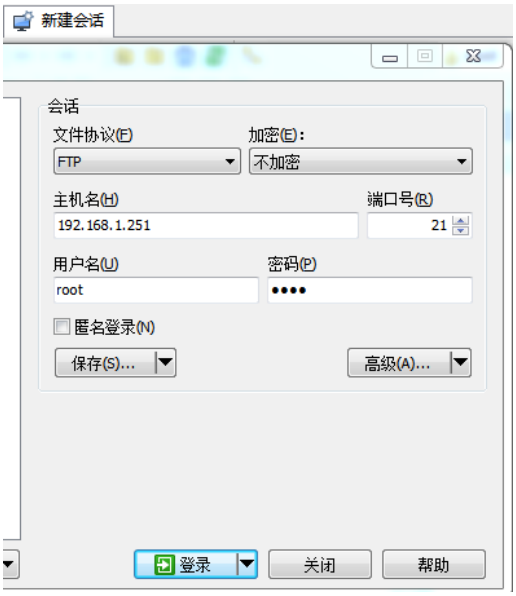
安装完成“vsftpd”后，系统自动启动 FTP 服务并设置了开机启动。

用户也可以编辑“/etc/vsftpd.conf”对 FTP 进行一些配置（一般配置无需修改），修改完成后，可以输入

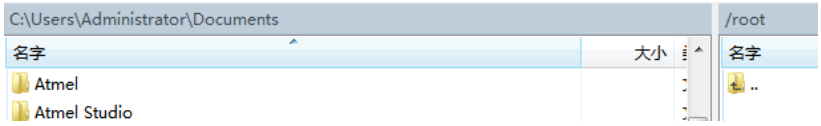
/etc/init.d/vsftpd restart　#重启 FTP 服务，使修改后的配置生效

3)　登录 FTP 服务器

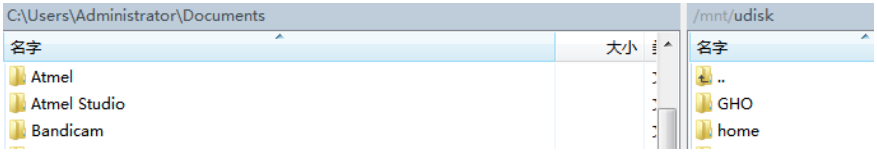
读者可以用之前介绍的 Winscp 通过 FTP 登录开发板。点击 winscp 的“新建会话”，输入如下图所示配置



上图中的用户名和密码都是“root”。然后点击“登录”按钮，即可登录开发板 FTP 服务器，如下图所示。



如果用户挂载了 U 盘，则可以进入其目录查看内容，如下图所示。



3.3.2. 安装 IPK 包

openwrt 安装软件包的方式有很多种，常见的有以下几种方法

1) 拷贝 IPK 安装包，本地离线安装。

这里以安装“usbutils”安装包为例，介绍如何进行本地离线安装，读者可以到“downloads.lede-project.org/snapshots”或者到“JS9331 开发板配套资料\JS9331 开发板固件镜像安装包\openwrt IPK 安装包”目录下，搜索“usbutils”，找到相应的安装包“usbutils_007-6_mips_24kc.ipk”，然后通过之前介绍的“电脑和开发板互传文件”方法，将该文件传输到开发板的"/tmp"目录下，调试终端执行命令

```
opkg install /tmp/usbutils_007-6_mips_24kc.ipk
```

结果如下所示

```
root@ZhuoTK:/# opkg install /tmp/usbutils_007-1_ar71xx.ipk
Installing usbutils (007-1) to root...
Collected errors:
 * satisfy_dependencies_for: Cannot satisfy the following dependencies for usbutils:
 *      libusb-1.0 *    librt *
```

以上结果提示我们还需要先安装“usbutils”依赖的“libusb-1.0”和“librt”安装包，用同样的方法找到这两个安装包并上传到开发板，并用调试终端重新执行以下命令

```
opkg install /tmp/librt_1.1.16-1_mips_24kc.ipk
opkg install /tmp/libusb-1.0_1.0.21-1_mips_24kc.ipk
opkg install /tmp/usbutils_007-6_mips_24kc.ipk
```

结果如下图所示

```
root@ZhuoTK:/# opkg install /tmp/librt_1.1.16-1_mips_24kc.ipk
Installing librt (1.1.16-1) to root...
Configuring librt.
root@ZhuoTK:/# opkg install /tmp/libusb-1.0_1.0.21-1_mips_24kc.ipk
Installing libusb-1.0 (1.0.21-1) to root...
Configuring libusb-1.0.
root@ZhuoTK:/# opkg install /tmp/usbutils_007-6_mips_24kc.ipk
Installing usbutils (007-6) to root...
Configuring usbutils.
```

上图表示软件包安装完成。“lsusb”是“usbutils”包里面的软件，我们试着在调试终端执行

```
lsusb                //显示当前系统的所有 USB 设备
```

测试一下，结果如下图所示

```
root@ZhuoTK:/# lsusb
Bus 001 Device 003: ID 1a86:7523 QinHeng Electronics HL-340 USB-Serial adapter
Bus 001 Device 002: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

该命令执行成功，进一步说明“usbutils”软件包已经安装好了。

开发板资料里面提供的安装包只包含了 openwrt 的一小部分安装包，如果读者没有找到需要的安装包，请参考“*深入开发篇*”的介绍，自行到 openwrt 源码的 make menuconfig 菜单里面选上需要的功能，然后编译出相应的安装包。

2) 通过 openwrt 网页网络下载安装

方法见之前“*软件包*”*子菜单*一节

3) 通过命令行网络下载安装

调试终端执行

opkg update //下载安装包信息，注意如果没有设置好网关、DNS 可能会报错

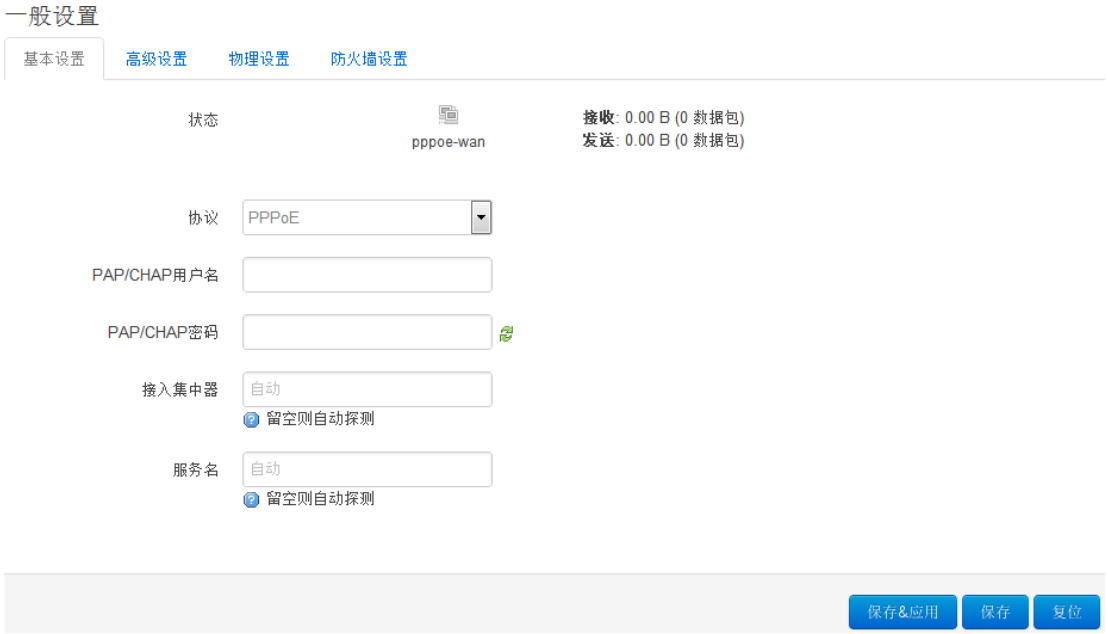
opkg install xxxx //安装软件，xxxx 为软件包名称，“kmod*”之类的安装包需要用自己编译的或开发板资料里面提供的，否则有可能会有兼容性问题。

3.3.3. 拨号上网

1) 给开发板上电，将开发板的 WAN 口与外网相连。
2) 登录开发板网页配置界面，依次点击菜单“网络”->“接口”出现“接口”配置页面。如下图所示。

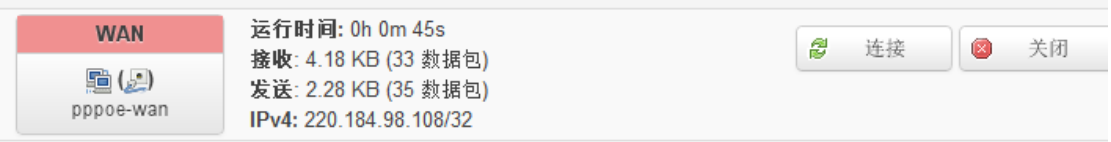
3) 点击“接口配置界面”中的“WAN”一栏中的“修改”，对 WAN 口进行配置，如下图所示。

4) 将“协议”一栏切换成“PPPoE”，然后在“PAP/CHAP 用户名”一栏中填入宽带用户名，在“PAP/CHAP 密码”一栏输入宽带密码。最后点击本页面右下的“保存&应用”按钮。



5) 如果一切顺利，我们将在“接口”页面，看到“WAN”口已经获取到 IP 地址，并

且开始收发数据了，这表示我们已经拨号成功了。用户可以通过连接 LAN 口或 wifi 上网。



3.3.4. 挂载 4G 网卡上网

随着我国各大电信运营商对4G网络大力推进,目前我国大部分地区已经支持4G网络了。注意：由于 4G 网卡的具体型号很多，并且驱动不是统一的驱动，我们无法一一测试。我们只能保证我们的开发板可以配合我们店里的 4G 网卡实现 4G 上网。不是本店购买的 4G 网卡，一般需要读者自行测试调试驱动。如读者尚未购买 4G 网卡，可到卓钛科技淘宝店进行购买 www.zhuotk.com。

下面就以开发板和本店出售的全网通 4G 网卡（如下图所示）

为例，介绍一下，如何增加开发板对 4G 网卡的支持。

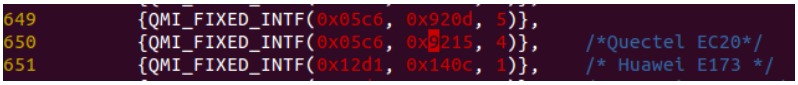
具体步骤如下：

1）添加内核本 4G 网卡的驱动支持

编辑“openwrt/build_dir/target-mips_34kc_uClibc-0.9.33.2/linux-ar71xx_generic/linux-3.10.49/drivers/net/usb/qmi_wwan.c”文件，在大概 649 行后面添加

“{QMI_FIXED_INTF(0x05c6, 0x9215, 4)}, /*Quectel EC20*/”

如下图所示。



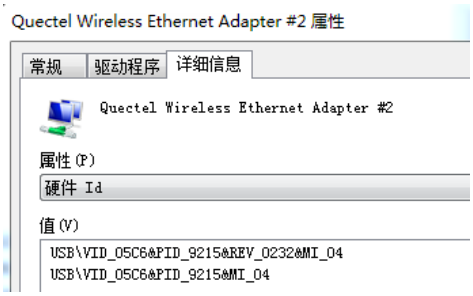
增加内核驱动对本网卡 PID、VID、接口号的识别。

注释掉大概在 786 行的

“{QMI_GOBI_DEVICE(0x05c6, 0x9215)}, /* Acer Gobi 2000 Modem device (VP413) */”

保存退出。

提示：具体“PID、VID、接口号”的查找方法，读者可以将 4G 网卡插到 windows 电脑上，然后查看该设备的属性即可，如下图所示。



2）选择并编译 4G 网卡驱动、测试工具

在 openwrt 根目录下执行 make menuconfig 进入功能配置界面。选择

Kernel modules　--->

　　USB Support　--->

　　<M> kmod-usb-net..... Kernel modules for USB-to-Ethernet convertors

　　<M>　kmod-usb-net-qmi-wwan..... QMI WWAN driver

Network　--->

　　<M> uqmi..... Control utility for mobile broadband modems

保存退出。

3）安装编译好的 4G 网卡驱动安装包和应用安装包安装

依次安装以下安装包

“kmod-mii”

“kmod-usb-net”

“kmod-usb-wdm”

“kmod-usb-net-qmi-wwan”

“uqmi”

安装完成后重启开发板。

注意：本地安装安装包的方法，已在《JS9331 开发板 openwrt 入门教程》里面有介绍，这里不再赘述。

4）测试 4G 网卡上网

　　安装了上述的安装包后并重启完成后，将 4G 手机卡插入 4G 网卡背后的卡槽中，然后将 4G 网卡插入开发板的 USB 接口，大概过 10s 左右，会出现下图的提示

```
root@JoySince:/# [   97.270000] usb 1-1.1: new high-speed USB device number 6 using ehci-platform
[   97.420000] qmi_wwan 1-1.1:1.4: cdc-wdm0: USB WDM device
[   97.420000] qmi_wwan 1-1.1:1.4 wwan0: register 'qmi_wwan' at usb-ehci-platform-1.1, WWAN/QMI devic
e, 8a:fa:bc:49:84:1b
```

上图表示系统已经正确识别 4G 网卡。同时，在"/dev"目前下会出现这个 4G 网卡的设备接口，本例中为“cdc-wdm0”，如下图红圈所示。

```
root@JoySince:/# ls /dev
bus                 mtd1ro              mtdblock2           random
cdc-wdm0            mtd2                mtdblock3           shm
console             mtd2ro              mtdblock4           tty
cpu_dma_latency     mtd3                mtdblock5           ttyATH0
```

　　执行命令

uqmi -d /dev/cdc-wdm0 --start-network internet --autoconnect　　//使网卡自动连接网络

　　//开发板重启后同样有效。

uqmi -d /dev/cdc-wdm0 --get-data-status　　//查看是否连接上

效果如下图所示

```
root@JoySince:/# uqmi -d /dev/cdc-wdm0 --start-network internet --autoconnect
"No effect"
root@JoySince:/# uqmi -d /dev/cdc-wdm0 --get-data-status
"connected"
```

从上图可知，4G 网卡已经连接到运营商的网络了。

5）给系统添加一个新接口

添加接口有两种方法，一种是通过修改配置文件，另外一种是通过网页添加。这里我们采用比较直观的网页添加方法。

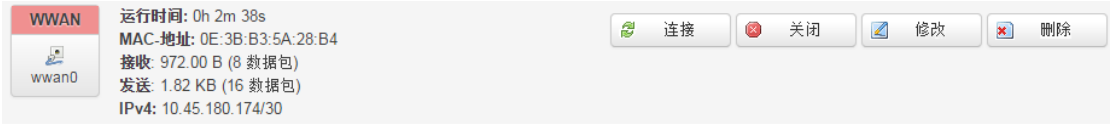
登陆 openwrt 配置页面，进入“网络”->“接口”->“添加新接口”。添加一个名为“wwan”的新接口，并选择“wwan0”适配器，如下图所示

然后点击该页面中的“提交”按钮。接着会出现下图

选择“防火墙设置”一栏，并配置成“wan”模式，如下图所示。

接着，点击右下角“保存&应用”。

回到“网络”->“接口”页面，可以看到网卡已经获取到 IP 地址，并已经有数据通信了，如下图所示。



至此，用户可以通过连接开发板的 LAN 口或者 wifi 进行 4G 上网了，开发板成为了一个 4G 路由器。

提示：有关 4G 网卡上网的其他介绍，读者可以参考“JS9331 开发板配套资料\学习资料\openwrt 学习资料\How To Use LTE modem in QMI mode for WAN connection.pdf”和“JS9331 开发板配套资料\学习资料\openwrt 学习资料\openwrt 基于 qmi 的 3G4G 拨号.pdf”。在本店购买 4G 网卡后，可以向客服索取 4G 网卡相关的驱动等资料。

3.3.5. 路由模式设置

路由模式特点：

1) 开发板开启 DHCP server
2) 外部设备（手机、电脑等）通过 wifi 或 LAN 口连接开发板
3) 开发板的 WAN 口接 ADSL（拨号、动态获取 IP、静态 IP）接入

下面介绍一下如何用 JS9331 开发板进行路由模式的配置（JS9331 开发板出厂时已经配置成路由模式）。

进入 openwrt 页面，点击顶部菜单“网络”->“接口”，点击“LAN”一栏中的“修改”。进入“LAN”口的配置界面，具体配置如下图所示。

上图中，已经配置好了路由模式。

3.3.6. AP 模式设置

AP 模式特点：

1) 开发板的 LAN 口连接局域网
2) 外部设备（手机、电脑等）通过开发板的 wifi 连接到局域网，实现上网
3) 开发板关闭 DHCP

下面介绍一下如何用 JS9331 开发板进行 AP 模式的配置。

进入“LAN”口的配置界面，具体配置如下图所示。

配置如上图所示，将 DHCP 关闭，否则开发板的 DHCP 功能有可能和现有局域网的 DHCP 服务器冲突，导致无法上网。修改完毕后，点“保存&应用”。

进入无线配置页面，配置如下图所示。

注意，开发板出厂时以上参数已设置好，一般无需更改。

3.3.7. 二级无线路由器("客户端")模式设置

二级无线路由器特点：

1) 开发板通过 wifi 连接上级无线路由器作为网络的接入
2) 电脑或其他设备连接开发板的 LAN 口或 wifi 上网
3) 开发板开启 DHCP
4) 连接开发板的设备和上级路由器处于不同网络，一般不能互通。

下面我们进行配置开发板“二级无线路由器”模式的演示。

1) 首先在开发板网页配置首页选择“网络”->“无线”，进入“无线”页面后点击“扫描”按钮，如下图所示

接着开发板会搜索周围的 AP 热点，并列出。如下图所示

开发板搜索到了作者事先用手机建立的名为“wurobinson”的 AP 热点（笔记本热点或路由器热点也可），如上图红圈所示。

2) 点击需要加入热点右侧对应的“加入网络”按钮，接着会进入“加入网络：设置”页面。如下图所示

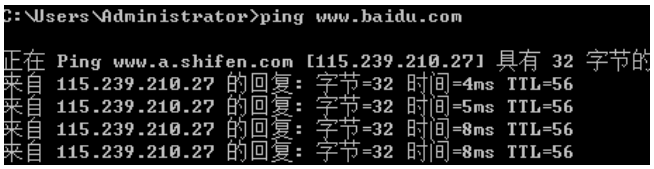
如果用户的 AP 热点设置有密码，则需在上图中的“WPA 密钥”一栏中填入该 AP 热点的密钥，其他选项一般无需配置，然后点击上图右下角的“提交”按钮即可。

3) 接着会进入“无线接口配置”页面，如下图所示。

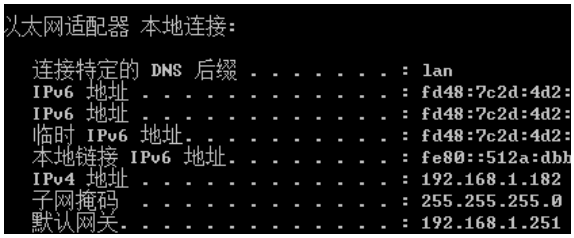
用户一般无需做其他配置，直接点击上图右下角的“保存&应用”按钮即可。在保存后，如果连接成功，在该页面中会显示出当前连接热点的信息，如连接信号强度等，如下图所示。

至此用户就可以通过开发板上网了（注意：如果读者的 wifi 热点网络网段也是“192.168.1.*”则会和开发板 LAN 网络网段冲突，导致网络不正常，需要读者自行修改开发板 LAN 网络网段或者修改 wifi 热点网段），下图是作者的台式机电脑通过开发板的 LAN 口上网。

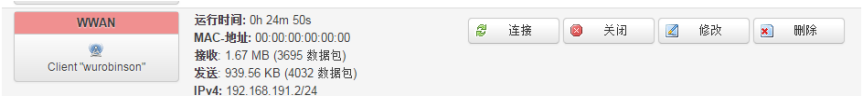
注意：开发板默认是开启 DHCP 服务的，所以无需再次开启，如果读者之前将该服务关闭，请开启。



上网设备（作者的台式机）的 IP 是由开发板分配的，并非上级设备（作者的手机）。如下图所示



而开发板的“WWAN”接口获取到的是由上级设备（作者的手机）分配到的 IP 地址（本例中是 192.168.191.2），读者可以到“网络”->“接口”里面查看，如下图。



4） 新建开发板的 wifi 热点（选做）

如果之前客户不小心删除了 wifi 热点，这时我们需要新建一个 wifi 接口，以便其他设备能通过开发板的 wifi 热点上网。

首先在开发板网页配置首页选择“网络”->“无线”，进入“无线”页面后点击“添加”按钮，进入新建无线接口的配置页面，如下图所示。

如上图所示，读者可根据自己的需要配置“ESSID”名，并在“网络”多选框里面选上“lan”，如红圈所示，其他选项不用配置，接着点击右下角“保持&应用”按钮即可。如果配置成功，读者可以用手机或其他 wifi 设备搜索到新建的 wifi 热点“Openwrt”，可连接上网。

小提示：

1）可以用同一个开发板同时建立多个 wifi 热点，方法同上。此方法可以多个开发板级联，从而实现“中继功能”，但上下级开发板的网段需要不同，同时注意修改开发板的 MAC 地址注意冲突。

2）如果用户想通过上级无线路由器访问到开发板，则需要将“WWAN”接口的防火墙修改为“LAN”模式，如下图所示。

然后，确保“LAN”和“WWAN”的 IP 网段不同，类似下图。

随后用户可以通过连接在上级路由的设备，访问 WWAN 的地址（此例中是“192.168.191.2”）进而访问到开发板。

3.3.8. 客户端（WDS）模式设置

客户端（WDS）模式特点：

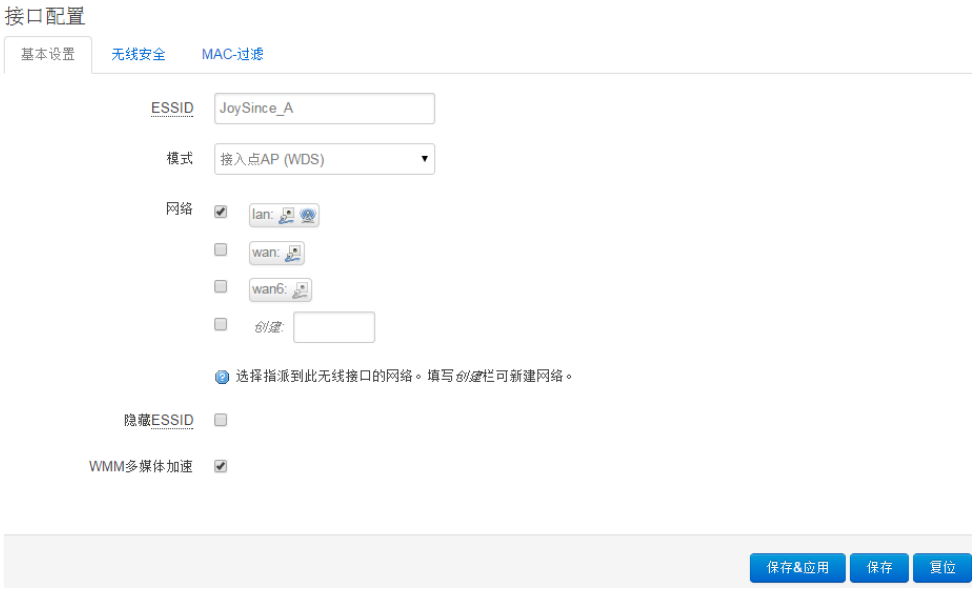
1) 作为客户端的开发板作为无线网卡使用。
2) 作为客户端的开发板关闭 DHCP，由上级网络 DHCP 服务器分配 IP。
3) 终端设备（电脑或手机等）连接作为客户端的开发板的 LAN 或 wifi 口进行上网，IP 可由上级路由器分配。
4) 终端设备和上级路由器处于同一个网络，可互通。

下面我们用 2 块 JS9331 开发板进行“客户端模式”的演示。

注意：如果是用开发板与普通路由器热点、手机热点、笔记本热点进行本实验，可能会由于 wifi 兼容性问题无法使用本模式，读者可使用上一节介绍的“二级无线路由器模式”。

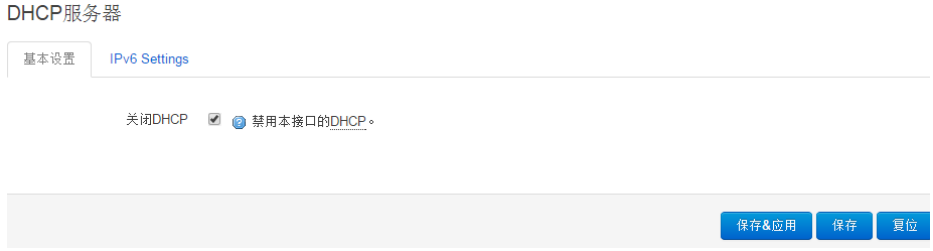
我们将一块开发板命名为 A——作为上级（主）无线路由。另一块开发板命名为 B——作为客户端（WDS）设备。进行以下步骤。

1) 首先我们 A 号开发板 IP 地址设为“192.168.1.251”，B 号开发板 IP 地址设为“192.168.1.252”，
2) 同时到 uboot 里面，使用“setmac xx:xx:xx:xx:xx:xx”修改两块开发板 MAC 地址，确保它们 MAC 不同，避免 MAC 地址冲突。比如 A 号设置为“00:ca:01:00:00:00”，B 号可设置为“00:ca:01:00:00:03”（间隔 3 个 MAC 地址）
3) 为避免混淆，修改 A 号开发板 SSID 改为“JoySince_A”，模式为“接入点 AP（WDS）”，配置如下图所示。



点击“保存&应用”。

4) 进入 B 号开发板，为避免 DHCP 服务冲突，需要关闭 B 号板的 DHCP 服务器。进入“网络”->“接口”->“LAN”->“修改”，配置如下图所示。

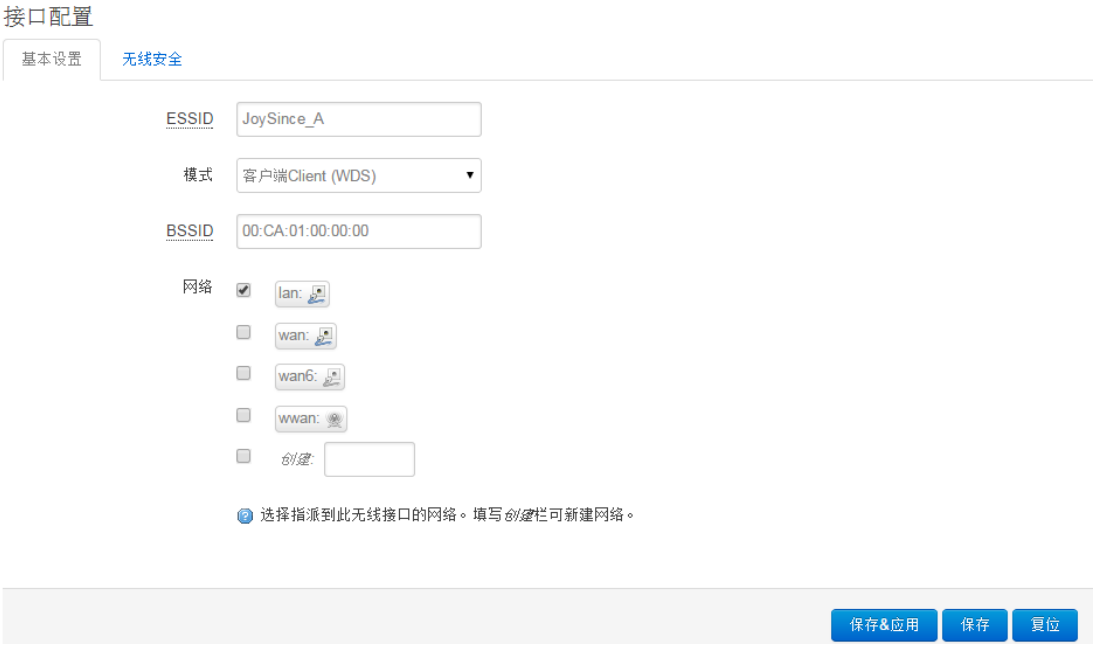


点击“保存&应用”。

5) 进入 B 号开发板“网络”->“无线”->“搜索”，找到 SSID 为“JoySince_A”的 wifi，点击“加入网络”按钮，如下图所示。

随后进入 “加入网络：设置”页面，该页面中的配置一般无须修改，直接点击“提交”即可，如下图所示。

6)　接着进入“接口配置页面”，模式一栏选择“客户端 Client(WDS)” 模式，网络一栏选择“lan”网络，如下图所示。

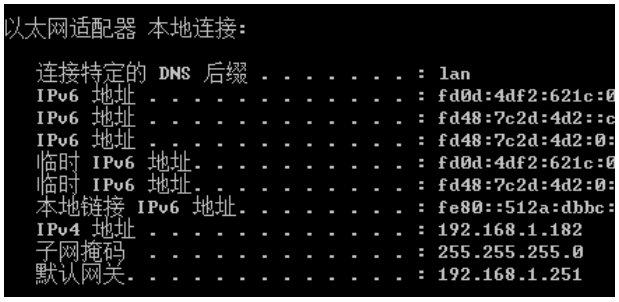


然后点击右下“保存&应用”。

9）这时，我们在当前页面可以看到 B 号板子已经通过 wifi 成功连接到 A 号板上。

至此，客户端模式设置完成。

10）测试网络是否连通。作者的电脑此时是连接在 B 号板子的 LAN 口上的，通过 A 号子的 DHCP 服务，获取到了 A 号板子分配给电脑 IP，如下图所示。



并且可以同时 ping 通 A、B 两个板子，说明电脑和 A、B 板子都处于同一个局域网络里面了。

3.3.9. 桥接（WDS）模式设置

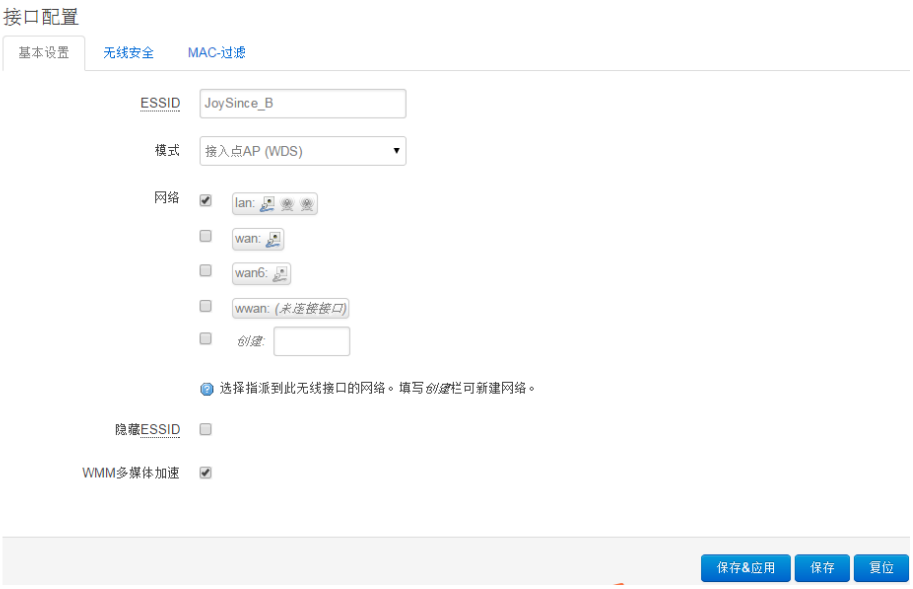
桥接模式特点：

1) 开发板用于拓展已有无线网络的覆盖范围
2) 开发板作为无线接入设备
3) 开发板关闭 DHCP
4) 开发板 SSID 与已有无线网络 SSID 不同

注意：如果是用开发板与普通路由器热点、手机热点、笔记本热点进行本实验，可能会由于 wifi 兼容性问题无法使用本模式，读者可使用上一节介绍的“二级无线路由器模式。”

“桥接模式”比“客户端模式”多了一个无线热点功能，下面我们就在上面“客户端”模式设置的基础上，在开发板 B 中加入无线热点功能。

进入开发板 B 的“网络”->“无线”->“添加”界面。模式可选择“接入点 AP（WDS）”或“接入点 AP”，两者的区别就是，前者还可以继续进行下级 WDS 桥接，后者不行。



点击“保存&应用”。这时我们可以看到在“网络”->“无线”页面中有两个接口，如下图所示。

读者可以用无线设备搜索并加入“JoySince_A”(A 号板的热点)，“JoySince_B”(B 号板的热点)

3.3.10. 中继（WDS）模式设置

中继模式特点：

1) 开发板用于拓展已有无线网络的覆盖范围
2) 开发板作为无线接入设备
3) 开发板关闭 DHCP
4) 开发板 SSID 与已有无线网络 SSID 相同

注意：如果是用开发板与普通路由器热点、手机热点、笔记本热点进行本实验，可能会由于 wifi 兼容性问题无法使用本模式，读者可使用上一节介绍的“二级无线路由器模式。”

由上述特点，我们只要将上一节“桥接模式”中的开发板 B 的 SSID 和加密方式，改为和开发板 A 相同即可。

小提示:读者可以查看"JS9331 开发板\项目输出\JS9331 开发板配套资料\学习资料\openwrt 学习资料\Atheros openwrt WDS.pdf " 这个 openwrt 官方有关 WDS 功能的介绍文档。

3.3.11. 开发板挂 VPN

VPN 是什么？VPN 全称是 Virtual Private Network，也就是虚拟专用网的意思。很多人不知道 VPN 作用是什么，其实 VPN 是通过因特网的一个临时安全连接，是一条穿越混乱公网的安全稳定的隧道，通过这条隧道可以安全地加密传输数据，常用于注重保密性的企业里。

有时候由于各种原因，导致无法直接上外服游戏或者是访问国外网站，这时也会用到它，因此它也被称为“翻墙软件”。想了解更多有关“VPN”知识的读者请自行百度一下。

下面介绍如何让开发板实现 VPN 客户端来连接远程 VPN 服务器，最终实现开发板通过 VPN 间接访问因特网。步骤如下。

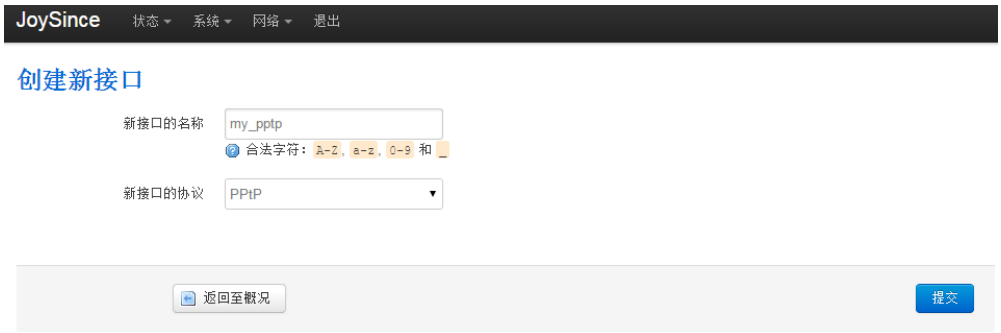
1）为了让 openwrt 支持 pptp 加密方式的 VPN，需要依次安装以下支持 VPN 的安装包

“ppp-mod-pptp”

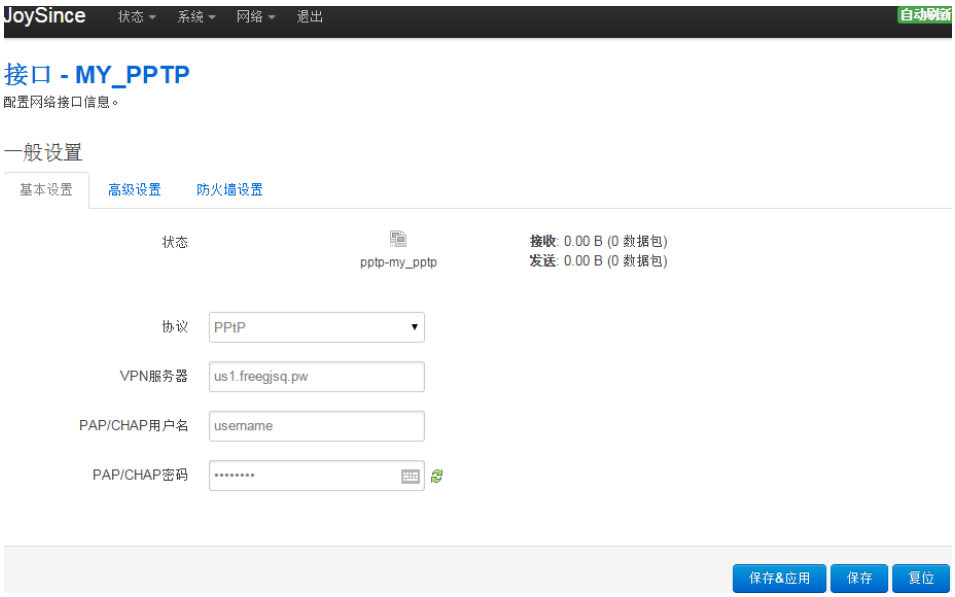
（注意：这个包依赖的其他安装包请读者自行安装，为了保证兼容性，“kmod*”的安装包需要用我们提供的或者是自己编译的。）

安装以上安装包后，为了使各个模块生效，需要重新启动一次开发板。

2）登录 openwrt 配置页面，进入“网络”->“接口”->“添加新接口”页面，如下图所示



上图中，为了便于识别接口作用，在“新接口的名称一栏”，作者填写接口名称为“my_pptp”，读者也可以命名为其他名字。在“新接口的协议”一栏选择“PPtP”，最后点击右下角的“提交”按钮。进入接口配置页面，如下图所示。



如上图，“VPN 服务器”一栏填入需要连接的 VPN 服务器地址（IP 或域名），“PAP/CHAP 用户名”一栏填写申请到的用户名，“PAP/CHAP 密码”一栏填写正确的密码。

点击右下角的“保存&应用”按钮，最后进入“网络”->“接口”页面，点击“MY_PPTP”接口一栏右边的“连接”按钮，如果成功连接，则会出现类似下图

由上图，我们可以看出，开发板已经正确的获取到了由 VPN 服务器分配的 IP 地址“10.0.0.85”，现在用终端执行 ping www.google.com，结果如下图所示

```
root@JoySince:/# ping www.google.com
PING www.google.com (216.58.199.4): 56 data bytes
64 bytes from 216.58.199.4: seq=0 ttl=51 time=323.414 ms
64 bytes from 216.58.199.4: seq=1 ttl=51 time=321.334 ms
64 bytes from 216.58.199.4: seq=2 ttl=51 time=322.967 ms
64 bytes from 216.58.199.4: seq=3 ttl=51 time=321.895 ms
64 bytes from 216.58.199.4: seq=4 ttl=51 time=320.205 ms
64 bytes from 216.58.199.4: seq=5 ttl=51 time=319.798 ms
```

而未连接该 VPN 前，无法访问 www.google.com，如下图所示

```
root@JoySince:/# ping www.google.com
PING www.google.com (216.58.197.100): 56 data bytes
```

提示：1、本例中建立的 VPN 客户端是开机自动连接的，即每次启动开发板，VPN 都是会自动连接的，如果读者只是需要的时候连接，请到“网络”->“接口”->“MY_PPTP”（接口名根据实际情况而定）->“修改”->“高级设置”页面，去掉“开机自动运行”一项，并点击右下角“保存&应用”，如下图所示。

2、如果读者遇到 PPTP 已经显示连接，却无法 ping 通国外一些网站，比如“www.google.com”、“www.twitter.com”等情况，这有可能是 DNF 解析的问题。读者可以尝试配置开发板默认的 DNS 配置文件“/etc/init.d/dnsmasq”，将里面默认的 DNS IP“127.0.0.1”修改为“114.114.114.114”或者其他 DNS 服务器 IP。效果如下

```
}
DNS_SERVERS="$DNS_SERVERS 114.114.114.114"
```

然后重启网络或重启开发板生效。

3.3.12. 网络串口透传

JS9331 开发板上拓展出了一个 RS232/TTL 串口（有关该串口介绍详见《JS9331 开发板使用手册》“开发板硬件资源”一节），下面我们就以该串口来实现“网络串口透传”功能。

1) 进入 openwrt “软件包”页面，搜索下载“ser2net”并安装，或者读者也可以到“JS9331 开发板配套资料\JS9331 开发板固件镜像安装包\openwrt IPK 安装包”搜索找到以上安装包。下面其他实验，均可用同样方法找到安装包。

2) 如果“ser2net”安装成功，在 openwrt 系统下会出现文件“/etc/ser2net.conf”，我们对这个配置文件进行编辑（建议用 vi 编辑，不能用“写字板”编辑，否则会导致 ser2net 不识别配置文件）。在其最后一行添加如下所示配置。

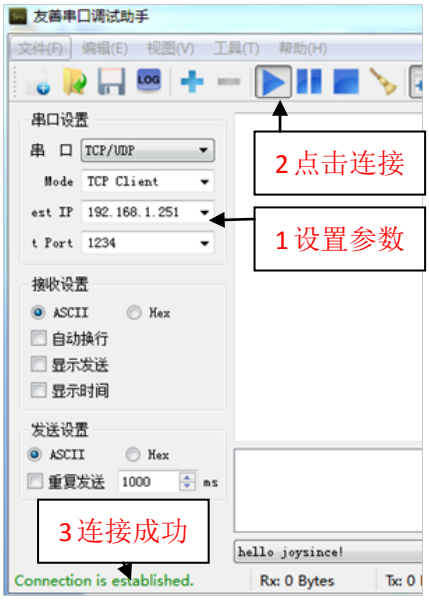
```
1234:raw:0:/dev/ttyUSB0:115200 NONE 1STOPBIT 8DATABITS  -RTSCTS
```

（具体参数含义在该配置文件里有介绍，请读者自行查看）最后保存退出。

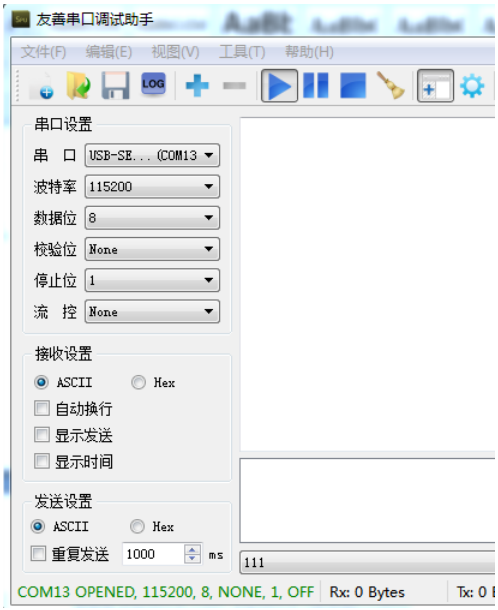
3) 在终端运行 ser2net。

```
root@JoySince:/# ser2net
```

4) 用 TCP 调试工具连接开发板 IP，端口 1234。这里我们用一款名为“友善串口调试助手”（“JS9331 开发板配套资料\开发工具\serial_port_utility_latest-v2.4.1.0516.zip”）的软件来调试。设置好后，连接开发板。如下图所示。

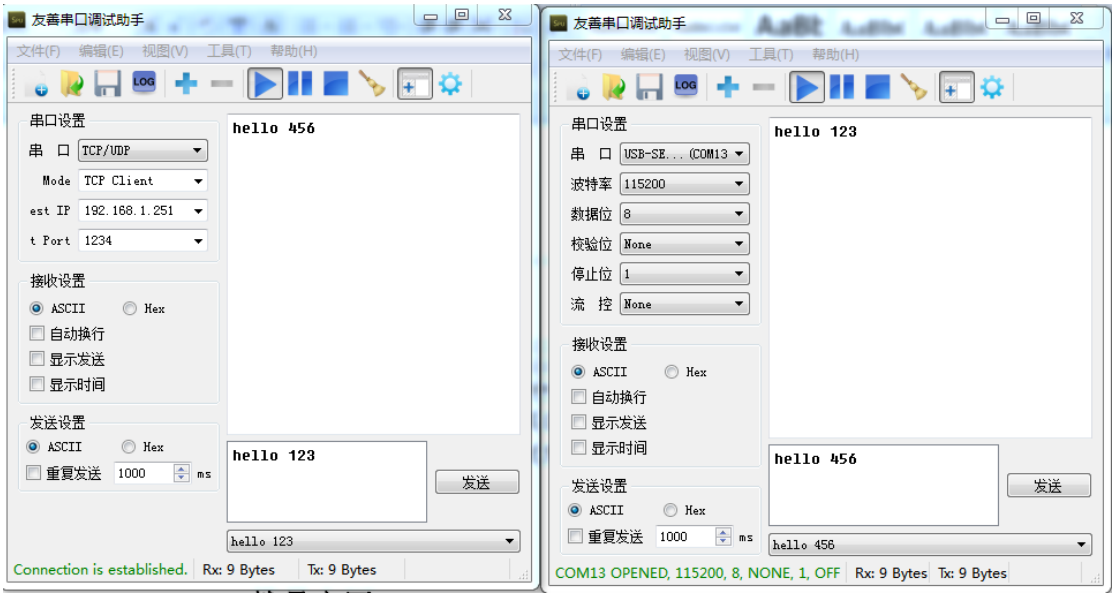


5) 用一条 USB 转 RS232 串口线连接电脑和开发板 RS232 接口，这里我们也用“友善串口调试助手”来连接串口，如下图所示。



连接步骤和上面类似，具体串口号需读者自行配置。

6) 最后我们可以在刚才开启的两个调试界面中互发数据。如下图所示。



至此“网络<--->串口”的透传功能实现。

提示：如果读者想让 ser2net 开机启动的话，可以到开发板的“/etc/rc.local”里面添加一个配置

sleep 3 #启动延迟防止 ser2net 启动不成功

ser2net&

如下图所示

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.
sleep 3
ser2net&
exit 0
```

然后保存退出，以后每次开发板开机，系统都会自动运行该程序。读者如果需要开机自启动其他的程序也可用此方法，不过需要注意的是，一些程序需要加参数才能正确运行。

3.3.13. 挂载 U 盘

如果想用开发板进行一些需要大容量存储的应用（比如在教程里介绍的“搭建 FTP 服务器”、“实现迅雷远程下载”、 “打造本地音乐播放器”等），我们可以挂载 U 盘（或移动硬盘）来实现。我们这里介绍手动挂载 U 盘和自动挂载 U 盘这两种方式

1)　手动挂载 U 盘

JS9331 开发板默认已经安装了 U 盘驱动，用户直接将 U 盘插入开发板 USB 接口，调试串口会出现类似下图提示

```
root@ZhuoTK:/# [  508.550000] usb 1-1.1: new high-speed USB device number 3 using ehci-platform
[  508.660000] usb-storage 1-1.1:1.0: USB Mass Storage device detected
[  508.680000] scsi host0: usb-storage 1-1.1:1.0
[  509.820000] scsi 0:0:0:0: Direct-Access     Kingston DT 101 G2       1.00 PQ: 0 ANSI: 2
[  509.830000] sd 0:0:0:0: [sda] 31324160 512-byte logical blocks: (16.0 GB/14.9 GiB)
[  509.840000] sd 0:0:0:0: [sda] Write Protect is off
[  509.840000] sd 0:0:0:0: [sda] No Caching mode page found
[  509.850000] sd 0:0:0:0: [sda] Assuming drive cache: write through
[  509.860000]  sda: sda1
[  509.870000] sd 0:0:0:0: [sda] Attached SCSI removable disk
```

上图说明开发板已经正确识别 U 盘，并提示 U 盘设备号是“sda1”，设备路径在"/dev/sda1"。读者参考“安装 IPK 安装包”一节，根据自己的 U 盘文件系统格式，有选择的安装以下安装包，

“ntfs-3g”　　//支持 NTFS 可读性文件系统

“kmod-fs-ext4”　　//支持 ext4 文件系统

“kmod-fs-vfat”　　//支持 vfat 文件系统

接着执行命令

```
mkdir /mnt/udisk   #建立一个用于挂载 U 盘的目录
mount /dev/sda1 /mnt/udisk   #将 U 盘设备“/dev/sda1”挂载到“/mnt/udisk”目录下
ls /mnt/udisk              #查看 U 盘内容
```

注意：如果需要拔掉 U 盘时，请先执行

```
umount /mnut/udisk    #卸载 U 盘
```

否则有可能会造成下次插入 U 盘时，设备号变成/dev/sdb1 等情况。

2) 自动挂载 U 盘

需要实现自动挂载 U 盘，首先需要根据自己的 U 盘文件系统格式，有选择的安装文件系统支持安装包（参考“手动挂载”一节进行安装，如果读者在“手动挂载”一节已经安装过相应的文件系统支持安装包了，则可以不用再次安装），接着输入以下命令

```
mkdir /etc/hotplug.d/block
vi /etc/hotplug.d/block/30-disk-mount
```

输入如下内容：

```
#!/bin/ash

    case "$ACTION" in
        add)
            for i in $(ls /dev/ | grep 'sd[a-z][1-9]')
            do
                mkdir -p /tmp/run/$i
                mount /dev/$i /tmp/run/$i
            done
            ;;
        remove)
            MOUNT=`mount | grep -o '/tmp/run/sd[a-z][1-9]'`
            for i in $MOUNT
            do
                umount $i
                if [ $? -eq 0 ]
                then
                    rm -r $i
                fi
            done
            ;;
esac
```

保存退出。这时我们可以重新拔插 U 盘，并出现正确识别 U 盘的调试信息后，脚本程序会把 U 盘自动挂载到开发板的“/tmp/run/sda1/”目录下，执行命令

```
ls /tmp/run/sda1      #查看 U 盘文件
```

注意：如果出现了“/tmp/run/sda1”文件夹，里面却没有内容（U 盘实际有内容），请检查是否安装了相应的文件系统支持安装包。资料里面也有提供现成的 U 盘自动挂载源码，在“JS9331 开发板配套资料\开发板源码\U 盘自动挂载脚本源码\30-disk-mount”。有关 vi 的使用，在“JS9331 开发板配套资料\学习资料\linux 应用开发学习资料\嵌入式 Linux 应用程序开发详解\嵌入式 Linux 应用程序开发详解-第 3 章 Linux 下的 C 编程基础.pdf”中有介绍，请读者务必熟悉这一 linux 下的常用工具。

3.3.14. 远程访问开发板

实现远程访问设备的方式很多，下面分别对“路由器端口映射访问”和“服务器中转访问”这两种方法进行说明。

3.3.14.1. 路由器端口映射访问

这里我们用“花生壳”的动态域名+路由器端口转发，实现远程访问开发板。这里假设读者将开发板连接到路由器，作为局域网设备。（用户也可以直接用开发板拨号上网，直接通过 IP 地址或域名访问开发板）

1) 注册一个域名

首先我们到“花生壳”官网 www.oray.com 注册一个花生壳用户，并申请一个免费域名。

2) 连接域名服务器

我们用 tp-link 路由器作为演示。进入 tplink 路由器管理界面->“动态 DNS”，配置如下图所示。

点击上图的，“登录”、“保存”，如果连接成功“连接状态”提示“连接成功”。

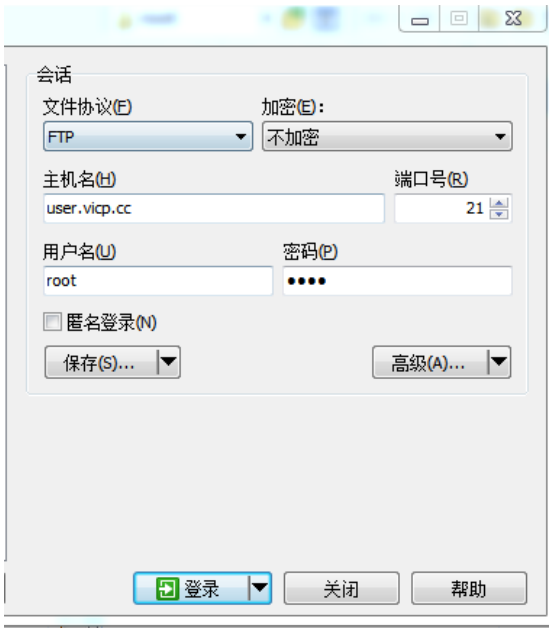
3) 设置端口映射

点击 tp-link 路由器“转发规则”->“虚拟服务器”，设置如下图所示。点击“添加新条目”增加端口服务映射。

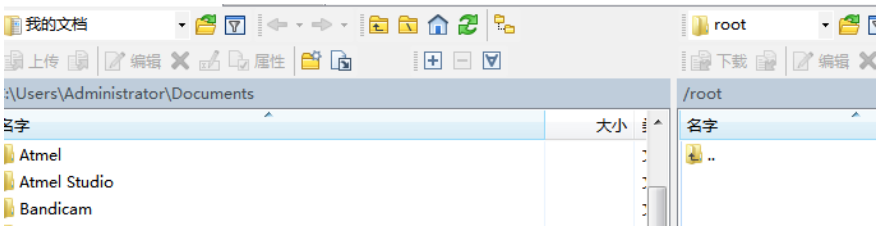
如上图所示，添加了 FTP 服务（21 端口）映射。

4) 远程访问测试

用 winscp 新建一个会话，连接 FTP 服务器，配置如下图所示。（下图中的“主机名”即申请的花生壳域名，用户名和密码均为“root”）



连接成功后，如下图所示。



注意:有些地区宽带运营商可能会封锁某些端口,比如 80、8080 端口，导致相应的一些服务比如 web 等无法正常访问，这时需要我们将该服务换一个端口解决这个问题。有一些是属于“内网宽带”，比如长城宽带，是无法使用这种端口映射的方法来实现远程访问的，读者可以百度“IP”查看自己的外网 IP，然后查看路由器 WAN 口获取到的 IP，如果这两个 IP 不一样，说明读者的网络就是属于“内网宽带”，读者可以购买“花生棒”来解决这个问题，或者采用下一节的“中转访问开发板”解决这个问题。

5) “路由器端口映射访问”优缺点

优点：不需要搭建额外的中转服务器

缺点：需要被访问的远程设备具有可以直接访问的独立公网 IP，很容易受网络环境的影响，经常无法实现直接访问。

3.3.14.2. 服务器中转访问（实现内网穿透）

“服务器中转访问”是本地设备经由公网上的服务器转发实现对远程设备（开发板）的访问。这里我们采用“域名解析”+“云服务器”+开发板+frp(内网穿透软件)实现这个功能。具体操作步骤如下

1) 购买域名注册和云服务器

本例中将采用的阿里云的域名和云服务器。读者可以到阿里云 www.aliyun.com（或其他服务提供商自行购买）进行购买。购买云服务器时，选择“按量付费、国内服务器、最低配置、ubuntu 16.04 系统”的阿里云服务器。购买域名（后面“测试 web 连接”时用到，如果只是做 TCP 应用通过 IP 访问，可以不用域名）时，读者可以选择花生壳域名或非热门的付费域名比如“.top”和，以减少前期测试费用。

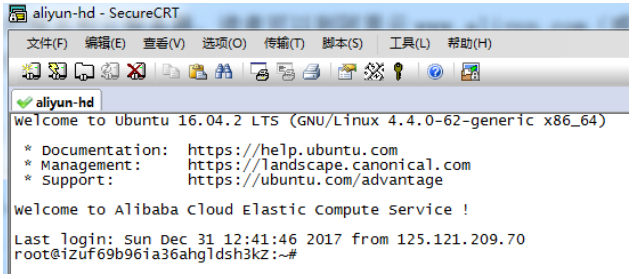
2) 配置阿里云服务器

进入“管理控制台”->“云服务器 ESC”->“实例”->　“管理”->“本实例安全组”->“配置规则”->“添加安全组规则”，设置开放 6000 至 8000 端口号（下面步骤需要用到端口，不设置这个规则会导致 frq 客户端无法访问云服务器），设置如下图所示

点击上图“确定”完成设置。

查看“管理控制台”->“云服务器 ESC”->“实例”，得到之前购买的云服务器的公网 IP，类似下图所示

我们用 secureCRT 选择“ssh”并填入在创建云服务器时填写的 ssh 登录密码和用户名，然后登录，登录后类似下图



执行

`uname -a`

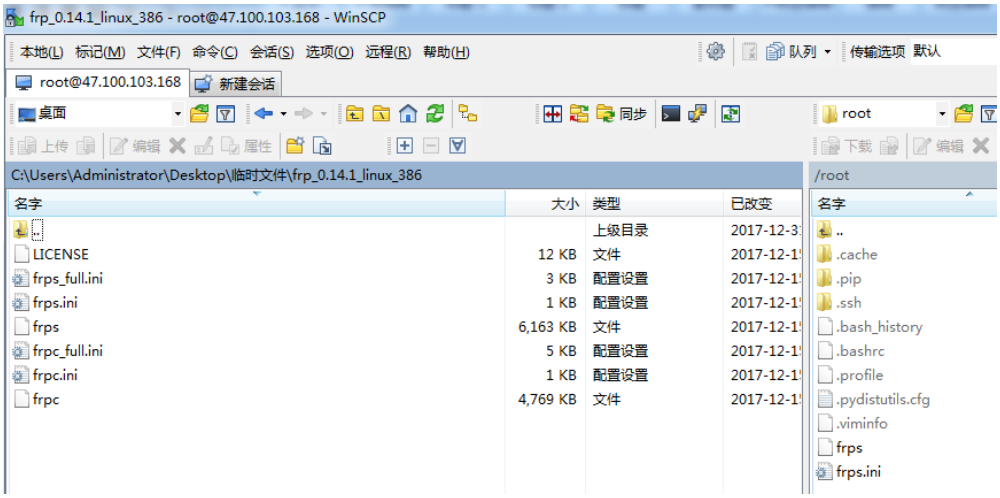
命令，获取系统构架信息，如下图所示

```
root@izuf69b96ia36ahgldsh3kz:~# uname -a
Linux izuf69b96ia36ahgldsh3kz 4.4.0-62-generic #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC 2017 x86_64
 x86_64 x86_64 GNU/Linux
```

由上图可知这个云服务器是“linux、x86_64”系统。

3)　服务器下载并运行“frp”内网穿透服务端

“frp”官方的下载地址是“https://github.com/fatedier/frp/releases”，根据上一步得到的云服务器系统信息，我们下载“frp_0.14.1_linux_386.tar.gz”（提示:在“JS9331 开发板固件镜像安装包\frp 内网穿透软件”已经有下载好的软件供读者使用），并解压其中的“frps”、“frps.ini”这两个文件，将这两个文件，用 winscp 上传到云服务器的“/root”目录下，如下图所示



云服务器执行命令

cd /root #切换到/root 目录

vi ./frps.ini　#打开并修改“frps.ini”文件，如下图

```
[common]
bind_port = 7000
```

保存退出，接着执行

chmod +x ./frps　#给 frps 添加执行权限

./frps -c ./frps.ini　#运行 frps，出现类似下图提示

```
2017/12/31 16:48:59 [I] [service.go:88] frps tcp listen on 0.0.0.0:7000
2017/12/31 16:48:59 [I] [main.go:112] Start frps success
2017/12/31 16:48:59 [I] [main.go:114] PrivilegeMode is enabled, you should pay more attention to s
ecurity issues
```

4)　开发板下载并运行“frp”内网穿透客户端

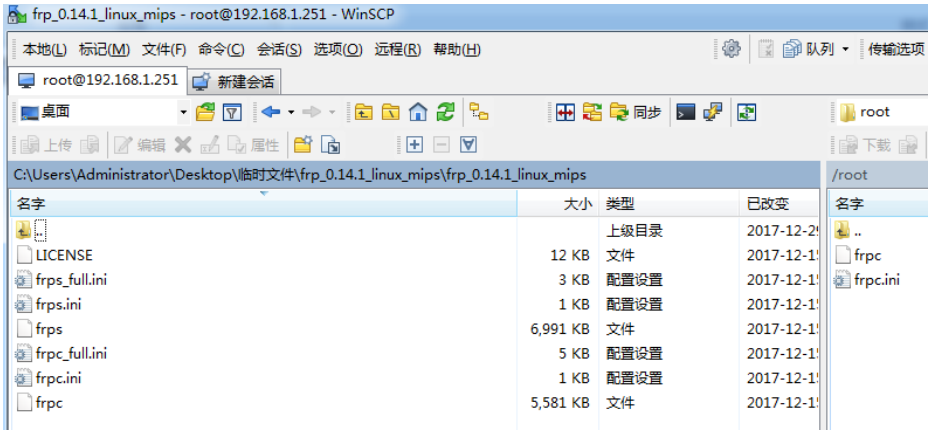
开发板上执行命令

uname -a

如下图所示

```
root@ZhuoTK:/# uname -a
Linux ZhuoTK 4.4.79 #0 Sat Aug 5 07:47:41 2017 mips GNU/Linux
```

从上图可知，开发板是“linux、mips”系统，因此我们下载“frp_0.14.1_linux_mips.tar.gz”（提示:在“JS9331 开发板固件镜像安装包\frp 内网穿透软件”已经有下载好的软件供读者使用），并解压其中的“frps”、“frps.ini”这两个文件，将这两个文件，用 winscp 上传到开发板的“/root”目录下，如下图所示



开发板执行命令

cd /root #切换到/root 目录

vi ./frpc.ini #打开并修改“frpc.ini”文件，如下图

```
[common]
server_addr = xxx.xxx.xxx.xxx
server_port = 7000
```

（上图“xxx.xxx.xxx.xxx”为云服务器 IP）然后保存退出，接着执行

chmod +x ./frpc #给 frpc 添加执行权限

在运行“frpc”之前，请先确保开发板能连接外网，否则 frpc 程序无法正常运行。开发板可以执行 ping www.baidu.com 进行测试是否联网，如果没有连接外网，请参考*“软件包”子菜单*一节设置 DNS 和网关。确保开发板可以联网后，执行

./frpc -c ./frpc.ini #运行 frpc，出现类似下图提示

```
2017/12/31 08:49:52 [I] [control.go:277] [f345d76927a5a476] login to server success, get run id [f345d76927a5a476], serv
er udp port [0]
```

说明该客户端已经连接到云服务端了。

5) 测试远程 ssh 连接

开发板退出“frpc”，配置开发板的"frpc.ini"文件，添加如下配置

[ssh_1]

type = tcp

local_ip = 127.0.0.1

local_ip = 22

remote_port = 6000

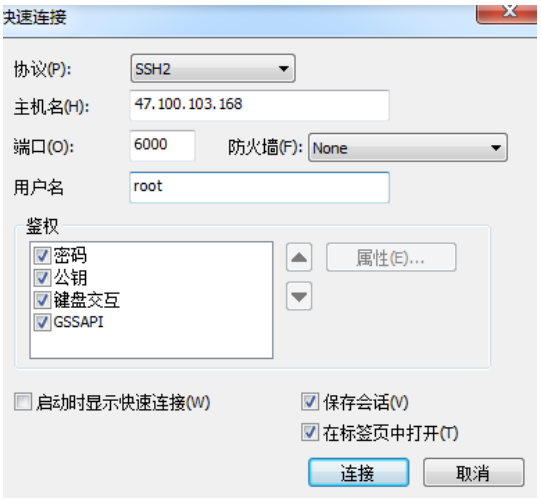
其中“local_ip = 22”是指映射到开发板 ssh 服务的 22 号端口，效果如下图

```
[common]
server_addr = xxx.xxx.xxx.xxx
server_port = 7000

[ssh_1]
type = tcp
local_ip = 127.0.0.1
local_port = 22
remote_port = 6000
```

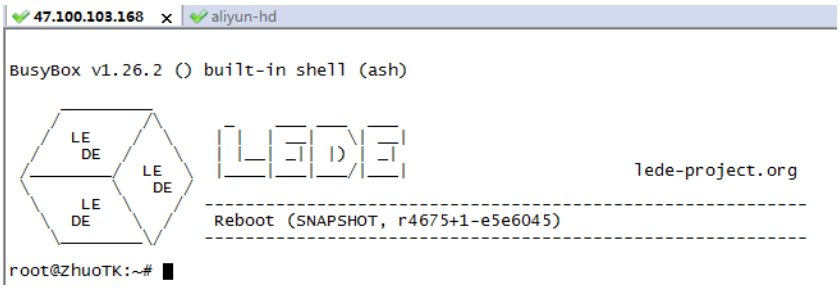
（上图“xxx.xxx.xxx.xxx”为云服务器 IP）开发板重新执行

./frpc -c ./frpc.ini

本地用 secureCRT 建立一个 ssh 终端连接，并填写云服务器 IP、开发板的用户名(root)和密码(root)、端口号 6000,如下图



顺利的话，就能连接到开发板的 ssh 控制终端了，如下图所示。



通过添加“frpc.ini”文件配置，读者还可以添加其他的一个或者多个 TCP 类型的服务，比如 FTP,添加配置类似下图

```
[ftp_1]
type = tcp
local_ip = 127.0.0.1
local_ip = 21
remote_port = 6001
.....
```

6)　测试远程 web 连接

首先配置域名的解析，将它的域名 A 记录指向云服务器 IP（记录值），类似下图

这里我们假设设置了一个域名“test1.yourdomain.com”指向服务器的 IP。

云服务器退出“frps”，修改配置文件“frps.ini”，添加如下配置

```
vhost_http_port = 8000
```

效果如下图所示

```
[common]
bind_port = 7000
vhost_http_port = 8000
```

执行命令

```
./frps -c ./frps.ini
```

重新运行服务端。

开发板退出“frpc”，配置开发板的"frpc.ini"文件，添加如下配置

```
[web_1]
type = http
local_port = 80
custom_domains = test1.yourdomain.com
```

其中“local_port = 80”是指映射到开发板 web 服务的 80 号端口,效果如下图所示

```
[common]
server_addr = 47.100.103.168
server_port = 7000

[ssh_1]
type = tcp
local_ip = 127.0.0.1
local_port = 22
remote_port = 6000

[web_1]
type = http
local_port = 80
custom_domains = test1.yourdomain.com
```

开发板重新执行

```
./frpc -c ./frpc.ini
```

此时，开发板出现如下图提示

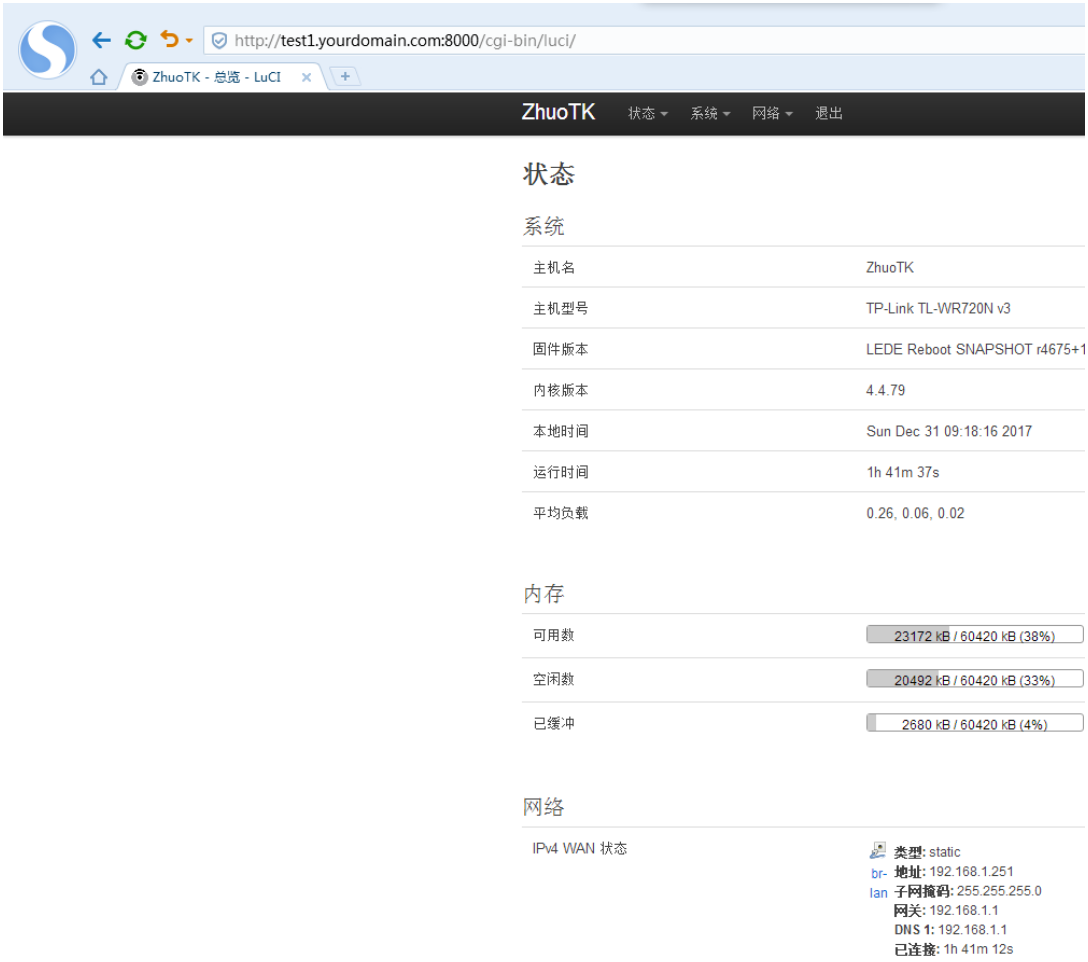
```
root@ZhuoTK:~# ./frpc -c ./frpc.ini
2017/12/31 09:11:29 [I] [control.go:277] [edd0978999e38ef8] login to server success, get run id [edd0978999e38ef8], serv
er udp port [0]
2017/12/31 09:11:29 [I] [control.go:412] [edd0978999e38ef8] [ssh_1] start proxy success
2017/12/31 09:11:29 [I] [control.go:412] [edd0978999e38ef8] [web_1] start proxy success
```

云服务器出现下图提示

```
2017/12/31 17:11:29 [I] [service.go:254] client login info: ip [125.121.209.70:10192] version [0.1
4.1] hostname [] os [linux] arch [mips]
2017/12/31 17:11:29 [I] [proxy.go:175] [edd0978999e38ef8] [ssh_1] tcp proxy listen port [6000]
2017/12/31 17:11:29 [I] [control.go:319] [edd0978999e38ef8] new proxy [ssh_1] success
2017/12/31 17:11:29 [I] [proxy.go:221] [edd0978999e38ef8] [web_1] http proxy listen for host [test
1.yourdomain.com] location []
2017/12/31 17:11:29 [I] [control.go:319] [edd0978999e38ef8] new proxy [web_1] success
```

说明已经开发板客户端连接到服务器，并且服务运行正常。

这时，我们可以用手机或者电脑的浏览器，输入“test1.yourdomain.com:8000”，从外网访问到开发板的配置页面



通过添加“frpc.ini”文件配置，读者还可以添加其他的一个或者多个 web 类型的服务，比如本教程里面“摄像头监控网页”,添加类似如下配置

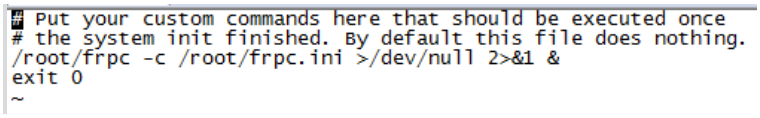
```
[web_2]
type = http
local_port = 8080
custom_domains = test2.yourdomain.com
.....
```

7)　设置开机启动

读者如果需要让程序开机启动，可以在开发板的“/etc/rc.local”添加

```
/root/frpc -c /root/frpc.ini >/dev/null 2>&1 &
```

如下图所示



云服务器添加方式同理。

8)　一些提示

需要深入了解源码和使用的读者可以到“https://github.com/fatedier/frp/”自行研究，“JS9331 开发板配套资料\开发板源码\frp 源码”有下载好的源码。读者还可以到 frp 服务提供商直接购买 frp 服务（比如 https://www.ngrok.cc/），这样就不用自行搭建 frp 服务器了。

9)　“服务器中转访问”优缺点

优点：穿透能力强，对网络环境没有什么限制，设备能上网就能实现远程访问。
缺点：需要额外的服务器，增加了成本，访问带宽受服务器带宽限制。对新手来说，配置也许也有些困难。

3.3.15. 打造本地音乐播放器

JS9331 播放本地音乐的步骤如下

1) 准备一个 USB 声卡（卓钛科技淘宝店铺有售），一个耳机或音箱，将其接到 USB 声卡上。

2) 安装以下软件

“kmod-usb-audio” //USB 声卡驱动

“madplay-alsa” //播放 MP3 的软件

开发板插入 USB 声卡，查看/dev 目录下是否有“audio”、“mixer”、“dsp”，如果有，则说明 USB 声卡驱动安装成功，系统已经正确识别 USB 声卡，如下图所示。

```
root@JoySince:/# ls /dev/
audio                mtd0r
bus                  mtd1
console              mtd1r
cpu_dma_latency      mtd2
dsp                  mtd2r
full                 mtd3
fuse                 mtd3r
kmsg                 mtd4
log                  mtd4r
mem                  mtd5
misc                 mtd5r
mixer                mtdbl
mtd0                 mtdbl
```

3) 将 MP3 文件通过之前介绍的“挂载 U 盘”、“电脑和开发板互传文件”各种方式，将音频文件放到开发板的文件系统中。

4) 开发板执行命令播放本地音乐

假如我们这里将一个 MP3 文件放在开发板的“/tmp/music.mp3”，在开发板的控制终端执行命令

```
madplay /tmp/music.mp3
```

结果类似下图

```
root@ZhuoTK:/# madplay /tmp/Lady\ Gaga\ -\ Summerboy.mp3
MPEG Audio Decoder 0.15.2 (beta) - Copyright (C) 2000-2004 Robert Leslie et al.
           Title: 12.Summerboy
                  Copyright (C) oimp3.com
          Artist: Lady GaGa
           Album: The Fame
           Track: 12
            Year: 2008
           Genre: Pop
         Comment: oimp3.com
[ 8245.100000] playback free_dma_buffer
[ 8245.110000] ptri2s_config->mmap_index:0
```

3.3.16. 打造无线音乐播放器

下面介绍一下将开发板打造成一个无线音乐播放器，并用手机 APP 控制播放。

1）准备硬件

首先，我们必须准备一个 USB 声卡（卓钛科技店铺有售），一个 U 盘，一个耳机或音箱。将一些 MP3 歌曲下载到 U 盘根目录下，以便演示。

2）安装声卡驱动

安装以下安装包

“kmod-usb-audio” //USB 声卡驱动

插入 USB 声卡，查看/dev 目录下是否有“audio”、“mixer”、“dsp”，如果有，则说明 USB 声卡驱动安装成功，系统已经正确识别 USB 声卡，如下图所示。

```
root@JoySince:/# ls /dev/
audio                   mtd0r
bus                     mtd1
console                 mtd1r
cpu_dma_latency         mtd2
dsp                     mtd2r
full                    mtd3
fuse                    mtd3r
kmsg                    mtd4
log                     mtd4r
mem                     mtd5
misc                    mtd5r
mixer                   mtdbl
mtd0                    mtdbl
```

3）安装 MPD

安装以下安装包

“zlib”

“libffi”

“libattr”

“libpthread”

“glib2”

“libflac”

“libmad”

“libcurl”

“libogg”

“libvorbisidec”

“mpd-mini”

4）编辑 MPD 配置文件

```
vi /etc/mpd.conf
```

在末尾加上（这里假设我们自动挂载的 U 盘目录名称是“sda1”，读者请根据实际情况修改，修改错误会导致实验失败）

```
music_directory                 "/mnt/sda1/"
playlist_directory              "/mnt/sda1/.mpd/"
db_file                         "/mnt/sda1/.mpd/mpd.db"
log_file                        "/tmp/mpd.log"
pid_file                        "/tmp/mpd.pid"
state_file                      "/mnt/sda1/.mpd/mpd.state"
user                            "root"
group                           "users"
bind_to_address                 "0.0.0.0"
port                            "6600"

audio_output {
type                    "oss"
name                    "My OSS Device"
device                  "/dev/dsp"          # optional
format                  "44100:16:2"        # optional
        mixer_type      "hardware"          # optional
        mixer_device    "/dev/mixer"        # optional
        mixer_control   "PCM"               # optional
}
```

```
filesystem_charset                    "UTF-8"
id3v1_encoding                        "GBK"
```

保存退出。

5）建立 MPD 所需目录和文件

```
mkdir -p /mnt/sda1/.mpd/playlist
touch /mnt/sda1/.mpd/mpd.db
touch /mnt/sda1/.mpd/mpd.log
touch /mnt/sda1/.mpd/mpd.error
touch /mnt/sda1/.mpd/mpd.pid
touch /mnt/sda1/.mpd/mpdstate
```

6）运行 MPD

```
/etc/init.d/mpd start
```

结果如下图所示。

```
root@JoySince:/# /etc/init.d/mpd start
BusyBox v1.22.1 (2015-07-04 17:37:42 CST) multi-call binary.

Usage: nice [-n ADJUST] [PROG ARGS]

Change scheduling priority, run PROG

        -n ADJUST       Adjust priority by ADJUST
```

7）安装 MPC

安装以下安装包

“libmpdclient”

“mpc”

8）运行 MPC

执行命令

```
mpc listall     #查看 mpd 是否正常识别音乐目录，并列出音乐文件
```

结果类似下图

```
root@JoySince:/# mpc listall
Era - The Mass.mp3
Katy Perry - By The Grace of God.mp3
Katy Perry - Roar(Acoustic).mp3
Katy Perry - Unconditionally(Acoustic).mp3
Lady Gaga - Lovegame.mp3
Lady Gaga - Marry The Night.mp3
Lady Gaga - Money Honey.mp3
Lady Gaga - Monster.mp3
```

接着执行命令

```
mpc listall | mpc add   #把音乐文件添加进音乐数据库
```

9）安装安卓 APP

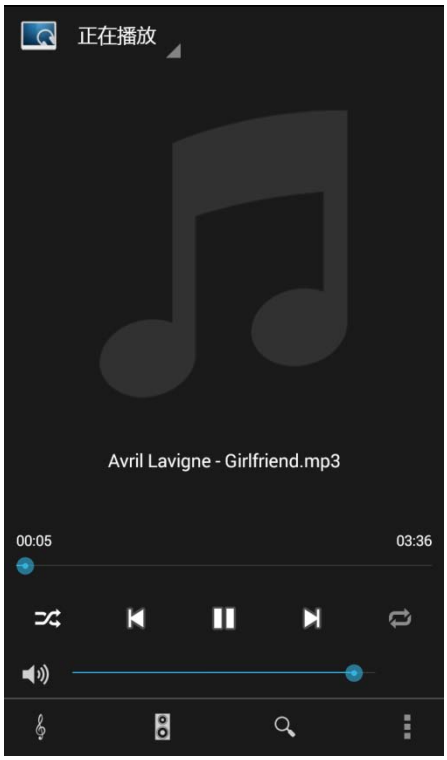
读者可以到“JS9331 开发板配套资料\JS9331 开发板固件镜像安装包\安卓 APP\MPDroid_chs.apk”自行下载安装。

10）设置 APP

手机通过 wifi 连接开发板，并在 APP 中设置开发板的 IP 地址，其他一般无需设置。

11）享受音乐

将耳机插到 USB 声卡，APP 点击“播放”按钮，开始听歌。



3.3.17. 实现迅雷远程下载

什么是"迅雷远程下载"？

下面介绍一下如何用"迅雷远程下载",实现远程控制开发板下载文件,随时随地,想下就下! 具体步骤如下。

1）挂载 U 盘（或移动硬盘）

由于从网上下载的音视频文件或其他文件一般都比较大，我们必须利用 U 盘作为存储介质，来存放迅雷远程下载的文件。挂载 U 盘方法，之前“挂载 U 盘”一节已有介绍，这里不再赘述。

2）确保开发板可以连接到外网。

开发板上执行如下命令

```
ping www.xunlei.com
```

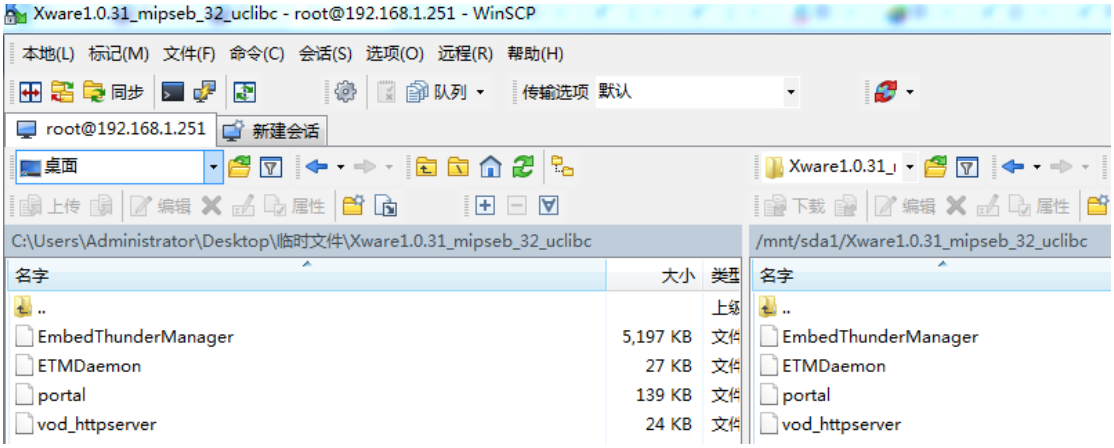
测试开发板是否可以 ping 通外网，如果无法 ping 通，请检查开发板是已经设置好网关、DNS 服务器地址（在 openwrt 网页配置“网络”->“接口”->“LAN”-“修改”中配置，之前“网络菜单项”一节有介绍）。测试结果如下图所示。

```
root@JoySince:/# ping www.xunlei.com
PING www.xunlei.com (119.147.41.155): 56 data bytes
64 bytes from 119.147.41.155: seq=0 ttl=57 time=31.085 ms
64 bytes from 119.147.41.155: seq=1 ttl=57 time=30.807 ms
64 bytes from 119.147.41.155: seq=2 ttl=57 time=30.392 ms
```

3）下载安装“迅雷远程下载”固件

读者可以到“http://luyou.xunlei.com/thread-12545-1-1.html”下载或直接将“JS9331 开发板配套资料\JS9331 开发板固件镜像安装包\迅雷远程下载固件\Xware1.0.31_mipseb_32_uclibc.zip”解压。

用 winscp 将解压后的文件复制到开发板挂载 U 盘的目录下，如下图所示。



4）开发板运行迅雷远程下载

开发板执行如下命令

```
/mnt/sda1/Xware1.0.31_mipseb_32_uclibc/portal
```

执行结果如下图所示

```
root@JoySince:/# /mnt/sda1/Xware1.0.31_mipseb_32_uclibc/portal
initing...
try stopping xunlei service first...
setting xunlei runtime env...
port: 9000 is usable.

YOUR CONTROL PORT IS: 9000

starting xunlei service...
etm path: /mnt/sda1/Xware1.0.31_mipseb_32_uclibc
execv: /mnt/sda1/Xware1.0.31_mipseb_32_uclibc/lib/ETMDaemon.

getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)
the active key is not valid.

try again...(has tried 1 time(s)).
getting xunlei service info...
Connecting to 127.0.0.1:9000 (127.0.0.1:9000)

THE ACTIVE CODE IS:  fhfxmc
```

如上图所示，我们获得了设备绑定激活码“fhfxmc”（读者实际获取到绑定码的应该与之不同）。

执行命令

```
ps
```

结果，如下图所示。

```
5238 root       832 S    /mnt/sda1/Xware1.0.31_mipseb_32_uclibc/lib/ETMDaemon
5240 root      9400 S    {EmbedThunderMan} /mnt/sda1/Xware1.0.31_mipseb_32_uc
5242 root      2872 S    {vod_httpserver} /mnt/sda1/Xware1.0.31_mipseb_32_ucl
```

从上图我们可以看到迅雷远程下载的后台进程。

5）登录迅雷远程下载，绑定开发板

浏览器输入“yuancheng.xunlei.com”，并登录迅雷远程下载页面。如下图所示。

点击上图中的“添加”，输入之前获得到的“设备绑定激活码”，点击“绑定”如下图所示。

如果提示绑定成功，则页面左侧，“我的下载器”会出现“下载器在线”图标，如下图所示

6）测试下载

点击迅雷远程下载页面中的“新建”，建立一个下载任务。

如上图所示，点击“添加到远程下载”，开发板开始下载。我们可以在迅雷下载页面中查看任务下载进度，如下图所示。

下载完成后，如下图所示。

同时，我们可以在开发板的“/mnt/sda1/TDDOWNLOAD”找到下载的文件，如下图所示。

```
root@JoySince:/# ls /mnt/sda1/TDDOWNLOAD/
QQ7.5.exe
```

读者可以用 FTP、winscp 或其他软件、方式，将下载到 U 盘中文件拷贝到需要的地方。

3.3.18. 挂载摄像头实现远程监控

Openwrt 还有一项功能比较吸引人，那就是挂载摄像头，实现远程实时监控。本节介绍通过“mjpeg-streamer”服务，实现输出摄像头的实时图像，然后实现远程监控。具体步骤如下

1)　开发板安装 mjpg-streamer 相关的 IPK 驱动包

“libjpeg”　　//mjpeg-streamer 依赖包

“mjpg-streamer”　// mjpeg-streamer 功能安装包

2)　插入 USB 摄像头

JS9331 开发板默认已经安装了 UVC 驱动，我们只需要将 USB 摄像头（该摄像头要支持 UVC 驱动且具有 MJPEG 输出功能，目前市面上有一些 USB 摄像头不支持以上功能，所以是无法做本实验的，如未购买，请到卓钛科技店铺 www.zhuotk.com 购买）插入开发板，观察调试串口的打印信息，如果有类似下图

```
root@ZhuoTK:/# [  221.940000] usb 1-1.1: new high-speed USB device number 3 using ehci-platform
[  222.430000] uvcvideo: Found UVC 1.00 device HP 720p HD Monitor Webcam (04f2:b2c3)
[  222.470000] input: HP 720p HD Monitor Webcam as /devices/101c0000.ehci/usb1/1-1/1-1.1/1-1.1:1.0/input/input0
```

则说明该 USB 摄像头已被开发板正确识别。

3) 配置摄像头参数

如果“mjpg-streamer”服务安装成功，则会出现“/etc/config/mjpg-streamer”配置文件，编辑该文件，修改成以下配置

```
config mjpg-streamer 'core'
        option enabled '1'                  #1 为开启摄像头功能，0 为不开启
        option input 'uvc'
        option output 'http'                #http 方式输出
        option device '/dev/video0'         #设备名
        option resolution '640x480'         #分辨率，和摄像头硬件支持有关
        option yuv '0'                      #是否是 YUV 输入，YUV 方式输入会非常卡
        option fps '30'                     #帧率，和摄像头硬件支持有关
        option led 'auto'
        option www '/www/webcam'            #访问目录，安装 mjpeg-stramer 包后生成
        option port '8080'                  #访问端口
#       option username 'openwrt'           #是否设置访问用户名
#       option password 'openwrt'           #是否设置访问密码
```

注：开发板选配的摄像头支持 640x480、800x600、1280x720 等分辨率，但是由于带宽、电脑配置等原因，推荐使用 640x480 以得到流畅性和清晰度最佳的平衡点。

4) 启动“mjpg-streamer”服务

输入以下命令

```
/etc/init.d/mjpg-streamer start
```

启动“mjpg-streamer”服务。

提示：

如果用户需要开机自动启动 mjpg-streamer 服务，只需要输入

```
/etc/init.d/mjpg-streamer enable
```

则在/etc/rc.d/目录下会自动出现“mjpg-streamer”服务相关的软连接，从而实现开机启动。如下图所示。

```
root@JoySince:/# /etc/rc.d/
K10mjpg-streamer  K99umount         S19firewall       S50uhttpd
K50dropbear       S00sysfixtime     S20network        S60dnsmasq
K85odhcpd         S10boot           S35odhcpd         S90mjpg-streamer
```

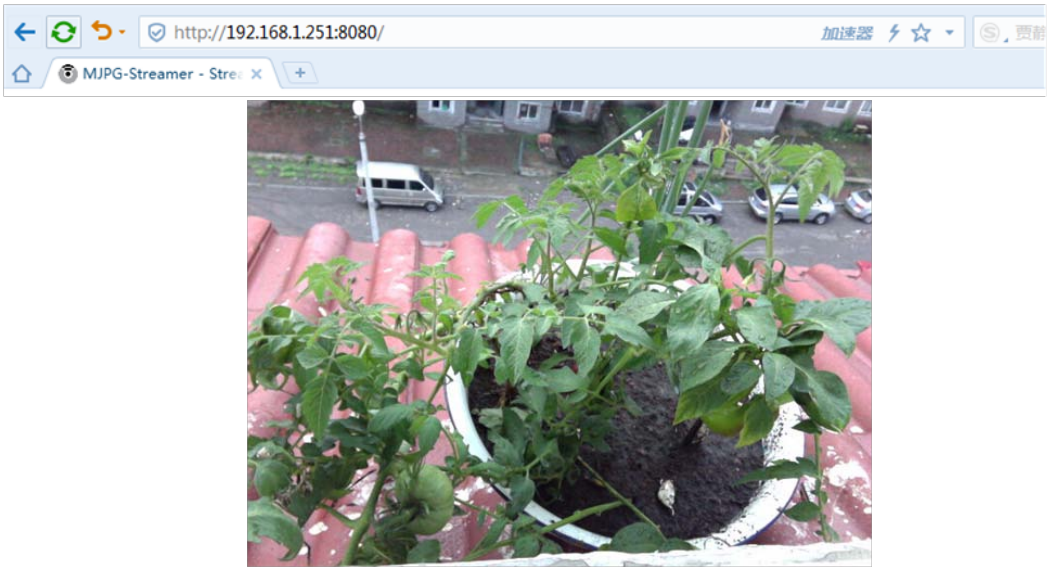
如果用户修改完 mjpg-streamer 的配置，需要让配置生效可以重启开发板或者执行

```
/etc/init.d/mjpg-streamer restart
```

5) 查看摄像头视频

用浏览器(电脑最好是 firefox 或者 chrome 内核的浏览器如搜狗浏览器、360 浏览器或者 QQ 浏览器,并且使用浏览器的急速模式或高速模式，不要用兼容模式，不要用 IE 浏览器。

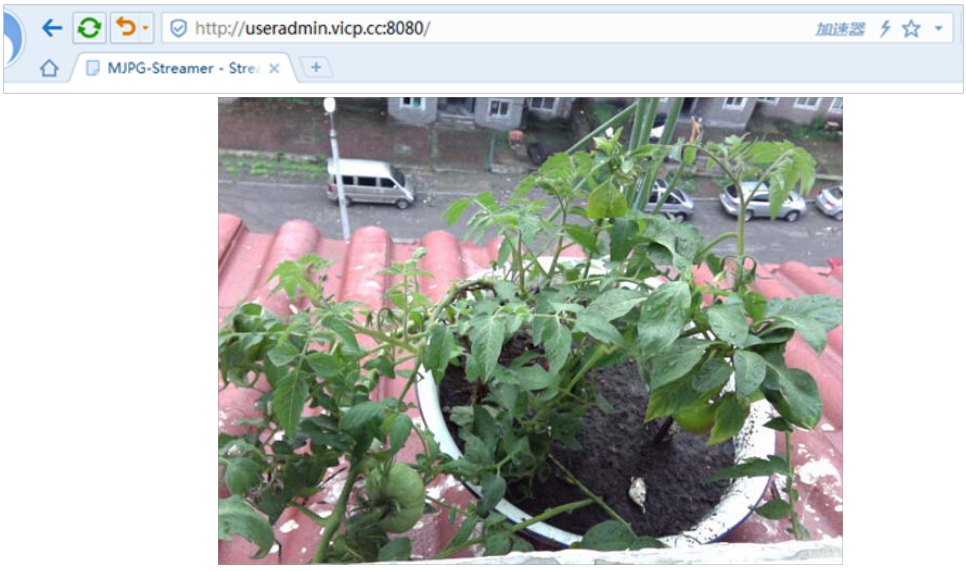
手机可以使用火狐浏览器)访问开发板的 8080 端口查看摄像头视频。效果如下图所示。



采用电脑或手机的 VLC 软件观看的话，在 VLC 地址栏填入
“http://192.168.1.251:8080/?action=stream”

6)　远程查看摄像头视频

这里的步骤和上面的“远程访问开发板”一节类似，我们只需要在路由器中或 frpc 客服端里为 mjpg-streamer 服务专门开放一个端口（比如 8080 端口）。远程访问摄像头的效果如下图所示。



注意：如果未设置开发板的网关、DNS，可能会导致无法从外网访问开发板。

3.3.19. 开发板拨打网络电话

这里简单介绍一下 SIP 会话初始协议（Session Initiation Protocol），它是用于初始、管理和终止网络中的语音和视频会话，具体地说就是用来生成、修改和终结一个或多个参与者

之间的会话。SIP 的业务模式是一个点对点协议，其中有两个要素——SIP 用户代理和 SIP 网络服务器。用户代理是呼叫的终端系统元素，而 SIP 服务器是处理与多个呼叫相关联信令的网络设备。每个 SIP 代理向 SIP 服务器注册之后，就可以通过拨号，向其他 SIP 用户代理进行呼叫、通话。更多的 SIP 资料请查看“JS9331 开发板配套资料\学习资料\SIP 学习资料”或者百度。

下面来介绍一下，如何利用 SIP 服务，实现开发板和手机之间拨打网络电话（VOIP）。具体步骤如下

1）申请 SIP 账号

在本例中，开发板扮演的是“SIP 用户代理”，而“SIP 服务器”则利用网上现有的 SIP 网络电话运营商提供的服务器（读者也可以自己搭建 SIP 服务器）。这里我们以“SIP139”这个运营商提供的 SIP 服务为例，注册两个 SIP 用户代理账户（一个账户给开发板，另外一个账户给手机使用或者也可以给另外一个开发板使用）。注册地址是

http://www.sip139.com/account/register.php?fr=sip139&id=80124787

目前在该网站提供的注册页面中其他方式有一定限制，读者只需要选择“普通注册”即可，如下图所示

注册完成之后，我们可以在该网站的“我的账号”页面下面看到“SIP 终端设置参数”

记下上图中的“SIP 账号”、“服务器地址”、“SIP 密码”（是你的注册密码）。然后再次注册一个账号。

2）开发板安装 baresip

在本例中，我们利用 openwrt 已经带有的 baresip 软件来实现“SIP 用户代理”功能，需要依次安装以下安装包。

“zlib”
“libopenssl”
“libpthread”
“libre”
“librem”
“baresip”
“kmod-sound-core”
“kmod-usb-audio”
“librt”
“alsa-lib”
“baresip-mod-alsa”

“baresip-mod-oss”
“baresip-mod-g711”
“baresip-mod-stdio”
以上安装包安装完成后，重启一次开发板，以便让一些安装包生效。

3）运行 baresip

开发板插上 USB 声卡（如果读者还未购买 USB 声卡，可以到卓钛科技淘宝店铺进行购买），连接好耳机耳麦。运行命令“baresip -f /etc/config/baresip_config”生成 baresip 的配置文件目录。如下图所示

```
root@JoySince:/# baresip -f /etc/config/baresip_config
baresip v0.4.11 Copyright (C) 2010 - 2014 Alfred E. Heggestad et al.
Local network address: IPv4=:192.168.1.252 IPv6=lo:fd66:25b5:781c::1
aucodec: PCMU/8000/1
aucodec: PCMA/8000/1
dl: mod: /usr/lib/baresip/modules/vumeter.so (File not found)
module vumeter.so: No such file or directory
ausrc: oss
auplay: oss
ausrc: alsa
auplay: alsa
medianat: stun
medianat: turn
medianat: ice
Populated 1 account
Populated 2 contacts
dl: mod: /usr/lib/baresip/modules/vidloop.so (File not found)
module vidloop.so: No such file or directory
Populated 2 audio codecs
Populated 0 audio filters
Populated 0 video codecs
Populated 0 video filters
baresip is ready.
reg: sip:user@lan: Register: Destination address required
```

如上图最后一行所示，baresip 无法连接到 SIP 服务器，这是因为我们还没有对账号信息进行配置。编辑/etc/config/baresip_config/accounts 这个文件，在尾行替换原来的配置，类似下图

```
#
<sip:80128888:6666@sip.sip139.com>
```

上图中的“80128888”是我们在上一步骤中获取到的“SIP 账号”，“6666”是密码，“sip.sip139.com”是 SIP 服务器地址，读者请根据自己的实际情况填写。填写完成后，保存退出。

再次运行命令“baresip -f /etc/config/baresip_config”，如果顺利则会出现下图中红圈提示的“连接成功提示”。

```
root@JoySince:/# baresip -f /etc/config/baresip_config
baresip v0.4.11 Copyright (C) 2010 - 2014 Alfred E. Heggestad et al.
Local network address: IPv4=br-lan:192.168.1.252 IPv6=lo:fd66:25b5:781c::1
aucodec: PCMU/8000/1
aucodec: PCMA/8000/1
dl: mod: /usr/lib/baresip/modules/vumeter.so (File not found)
module vumeter.so: No such file or directory
ausrc: oss
auplay: oss
ausrc: alsa
auplay: alsa
medianat: stun
medianat: turn
medianat: ice
Populated 1 account
Populated 2 contacts
dl: mod: /usr/lib/baresip/modules/vidloop.so (File not found)
module vidloop.so: No such file or directory
Populated 2 audio codecs
Populated 0 audio filters
Populated 0 video codecs
Populated 0 video filters
baresip is ready.
8012[redacted]0@sip.sip139.com: {0/UDP/v4} 200 OK (wsvoip2.0.1) [1 binding]
All 1 useragent registered successfully! (4930 ms)
```

出现了上图后，说明开发板上运行的 baresip 已经处于正确运行的状态，此时用户可以输入"h"查看该软件的一些用法，如下图所示。

```
--- Help ---
ENTER   Accept incoming call
ESC     Hangup call
SPACE   Toggle UAs
- ..    Send MESSAGE to peer
/ ..    Dial from contacts
= ..    Select chat peer
A       Stop audio-loop
C       List contacts
M       Main loop debug
a       Start audio-loop
b       Hangup call
c       Call status
d ..    Dial
g       Print configuration
h       Help menu
i       SIP debug
l       List active calls
m       Module debug
n       Network debug
q       Quit
r       Registration info
s       System info
t       Timer debug
u       UA debug
y       Memory status
```

3）手机安装 SIP APP 进行测试

首先必须保证手机的 wifi 或者数据连接已经连接到网络，接着我们用一个叫“CSipSimple”的APP进行软件测试，该软件在“JS9331开发板配套资料\开发工具\csipsimple_1.02.03.apk”。

安装完成后，打开该 APP，如下图所示。

点击上图红圈所示按钮，进入下图所示界面。

点击上图的“增加账号”，进入下图所示界面。

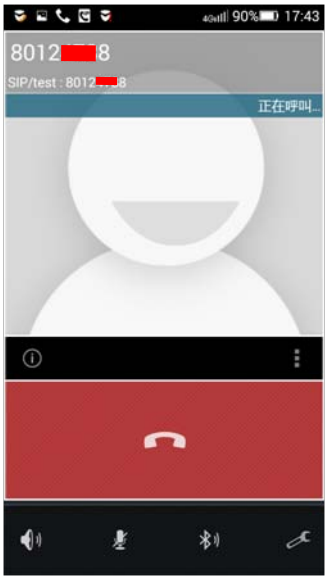
选择上图中的“Basic”一项，进入下图所示界面。

依次填写“账号名称”（随意）、“账号”（即之前注册的另外一个 SIP 账号）、“服务器”（SIP 服务器）、“密码”这几栏，然后点击右下角的“保存”按钮，进入下图所示界面。

如上图所示，图中的“已注册”表示已经和 SIP 服务器建立连接。 我们返回 APP 打开时候

的那个拨号界面，对之前开发板注册的号码进行拨号。如下图所示。

点击上图中的拨号按钮  ，进入正在拨号界面，如下图所示。



此时我们可以看到开发板的终端上出现了“打进电话的提示”，如下图。

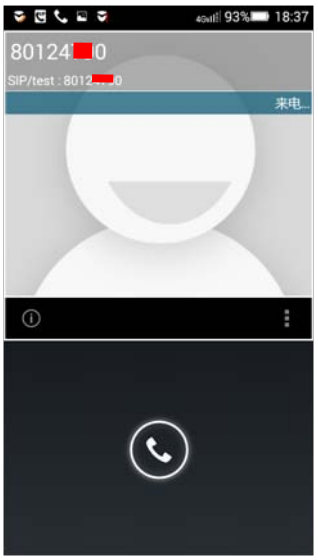
```
sip:80124788@sip.sip139.com: Incoming call from: 80124787 sip:80124787@42.121.195.110:5060 - (press ENTER to accept)
```

同时和 USB 声卡的连接的耳机里面也能听到“来电铃声”。此时我们在开发板终端按下“enter”键即可接通电话，进行通话。

当然，开发板也可以主动拨号。运行 baresip，并成功连接后，在终端输入“d”命令，然后输入需要拨打的号码，输入完成后，按下“enter”键即可，如下图所示

```
> 80124787
call: connecting to 'sip:80124787@sip.sip139.com'..
call: SIP Progress: 100 Trying (/)
call: SIP Progress: 180 Ringing (/)
```

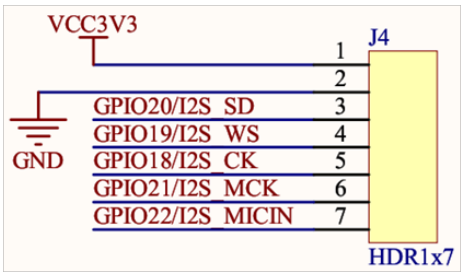
此时如果手机上运行着 SIP 的 APP，则会出现来电提示，如下图所示。

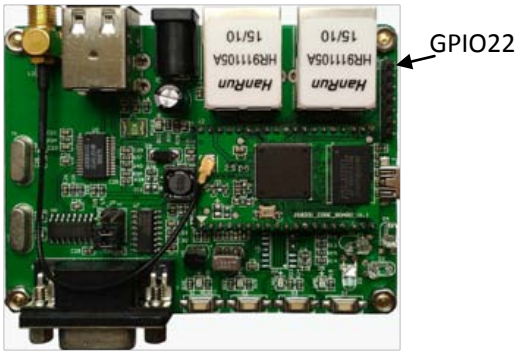


提示：开发板和开发板之间也可以实现打电话，需要两个开发板和两个 USB 声卡、两个耳机，步骤几乎和上面介绍的一致，这里就不再说明，给读者留下一点自由发挥的空间。如果用户的“sip139”有话费的话，甚至还可以直接用开发板或拨打普通电话！

3.3.20. GPIO 简单控制

这里用命令实现简单控制 GPIO。下面我们以 GPIO22 为例子作说明。GPIO22 对应的原理图和实物对应如下（开发板配套资料中有完整底板原理图），读者可以观察 PCB 板子上的丝印得知它们对应关系。





命令控制 GPIO 的具体步骤如下

a) 执行以下命令

```
cd /sys/class/gpio/                    //进入 GPIO 控制目录
echo 22 > export                       //生成 GPIO22 的控制目录
cd gpio22/                             //进入 GPIO22 的控制目录
```

b) 查看当前 GPIO 的输入输出状态，执行命令

```
cat direction
```

结果如下图所示

```
root@JoySince:/sys/devices/virtual/gpio/gpio22# cat direction
in
```

上图“in”表示为输入状态。

c) 查看当前 GPIO 的电平状态，执行命令

```
cat value
```

结果如下图所示

```
root@JoySince:/sys/devices/virtual/gpio/gpio22# cat value
0
```

将 GPIO22 通过电阻或直接上拉到高电平（VCC3V3），再次执行命令结果如下图

```
root@JoySince:/sys/devices/virtual/gpio/gpio22# cat value
1
```

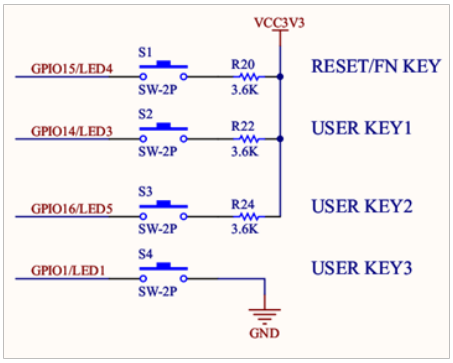
d) 测试 GPIO 输出，执行命令

```
echo out > direction        //将 GPIO 置为输出状态
cat direction               //查看当前 GPIO 输入输出状态
echo 1 > value              //将 GPIO 置为高电平，请读者自行用万用表或其他仪器测量电平
echo 0 > value              //将 GPIO 置为低电平
```

提示：有关 GPIO 控制的官方介绍在“JS9331 开发板配套资料\学习资料\linux 应用开发学习资料\gpio.txt”

3.3.21. 修改按键功能

在开发板上有 4 个按键，它们的原理图如下图所示



在 AR9331 的 openwrt 系统中，增加系统的按键需要做以下几步。

1）修改源文件，添加 GPIO 和按键的对应关系

在 openwrt_barri_test/target/linux/ar71xx/files/arch/mips/ath79/mach-tl-wr720n-v3.c”增加 GPIO 和按键的对应关系，如下图所示（下面只列出关键部分，读者可查看打完补丁后的该文件）。

```
#define TL_WR720N_GPIO_BTN_RESET        15
#define TL_WR720N_GPIO_BTN_SW1          14
#define TL_WR720N_GPIO_BTN_SW2          16
#define TL_WR720N_GPIO_BTN_SW3          1
```

```
static struct gpio_keys_button tl_wr720n_gpio_keys[] __initdata = {
        {
                .desc           = "reset",
                .type           = EV_KEY,
                .code           = BTN_0,
                .debounce_interval = TL_WR720N_KEYS_DEBOUNCE_INTERVAL,
                .gpio           = TL_WR720N_GPIO_BTN_RESET,
                .active_low     = 0,
        },{
                .desc           = "sw1",
                .type           = EV_KEY,
                .code           = BTN_1,
                .debounce_interval = TL_WR720N_KEYS_DEBOUNCE_INTERVAL,
                .gpio           = TL_WR720N_GPIO_BTN_SW1,
                .active_low     = 0,
        },{
                .desc           = "sw2",
                .type           = EV_KEY,
                .code           = BTN_2,
                .debounce_interval = TL_WR720N_KEYS_DEBOUNCE_INTERVAL,
                .gpio           = TL_WR720N_GPIO_BTN_SW2,
                .active_low     = 0,
        },{
                .desc           = "sw3",
                .type           = EV_KEY,
                .code           = BTN_3,
                .debounce_interval = TL_WR720N_KEYS_DEBOUNCE_INTERVAL,
                .gpio           = TL_WR720N_GPIO_BTN_SW3,
                .active_low     = 1,
```

从上图可以看出，程序里面的 GPIO 引脚号是和开发板一一对应的。修改完成后，需要将该文件拷贝至

“openwrt/build_dir/target-mips_34kc_uClibc-0.9.33.2/linux-ar71xx_generic/linux-3.10.49/arch/mips/ath79/”覆盖掉对应的文件，重新编译系统，方可生效。

2）增加按键功能

在开发板的四个按键中，左边第一个默认已经配置为“复位/恢复出厂设置”功能，

在开发板的“/etc/config/system”文件中，读者可以看到这个按键功能是如何配置的，如下图所示

```
config button
        option button 'BTN_0'
        option action 'released'
        option handler 'reboot'
        option min '0'
        option max '3'

config button
        option button 'BTN_0'
        option action 'released'
        option handler 'jffs2reset -y && reboot'
        option min '5'
        option max '30'
```

其他三个按键均未配置功能，下面演示一下如何将左边第二个按键添加“开启/关闭 wifi 功能”。编辑开发板的“/etc/config/system”文件，在其末尾添加如下配置

```
config button
    option button 'BTN_1'#按键名为“BTN_1”
    option action 'pressed' #按下触发
    option handler '/usr/bin/wifionoff' #执行“/usr/bin/wifionoff”这个文件
```

然后再新建文件“/usr/bin/wifionoff”添加如下配置

```
#!/bin/sh
SW=$(uci -q get wireless.@wifi-device[0].disabled)
[    "$SW" == "1"    ] && uci set wireless.@wifi-device[0].disabled=0
[    "$SW" == "1"    ] || uci set wireless.@wifi-device[0].disabled=1
wifi
```

保存退出，执行以下命令

```
chmod +x /usr/bin/wifionoff
```

3）测试按键

完成步骤 2）后，我们可以尝试按下左边第二个按键，看一下 wifi 是否真的可以通过该按键来控制。测试结果，调试串口打印信息如下

第一次按下按键结果

```
root@JoySince:/# [ 2970.150000] device wlan0 left promiscuous mode
[ 2970.150000] br-lan: port 2(wlan0) entered disabled state
```

同时我们用手机已经看不到开发板的 wifi SSID 了。

再次按下按键结果

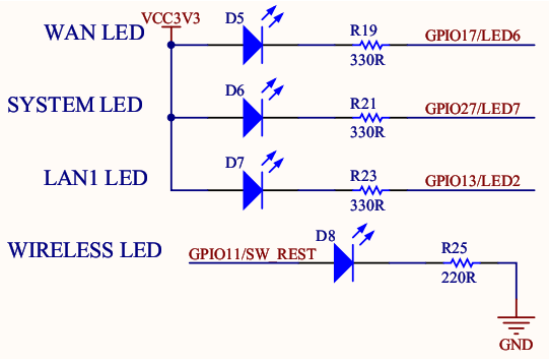
```
[ 3129.760000] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 3129.780000] device wlan0 entered promiscuous mode
[ 3131.020000] br-lan: port 2(wlan0) entered forwarding state
[ 3131.020000] br-lan: port 2(wlan0) entered forwarding state
[ 3131.030000] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 3133.020000] br-lan: port 2(wlan0) entered forwarding state
[ 3141.210000] device wlan0 left promiscuous mode
[ 3141.210000] br-lan: port 2(wlan0) entered disabled state
[ 3141.810000] IPv6: ADDRCONF(NETDEV_UP): wlan0: link is not ready
[ 3141.820000] device wlan0 entered promiscuous mode
[ 3141.820000] br-lan: port 2(wlan0) entered forwarding state
[ 3141.820000] br-lan: port 2(wlan0) entered forwarding state
[ 3142.220000] br-lan: port 2(wlan0) entered disabled state
[ 3143.060000] br-lan: port 2(wlan0) entered forwarding state
[ 3143.060000] br-lan: port 2(wlan0) entered forwarding state
[ 3143.070000] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[ 3145.060000] br-lan: port 2(wlan0) entered forwarding state
```

以上信息表明 wifi 已经开启，同时在手机上又可以搜索到开发板 wifi SSID 了。

提示：在“JS9331 开发板配套资料\学习资料\openwrt 学习资料\Attach functions to a push button [OpenWrt Wiki].pdf”是官方有关按键的说明，请读者自行查看。

3.3.22. 修改 LED 指示功能

在开发板上有 4 个 LED 指示灯，该部分的原理图如下图所示



在 AR9331 的 openwrt 系统中，增加系统的 LED 需要做以下几步。

1）修改源文件，添加 GPIO 和 LED 的对应关系

在“openwrt_barri_test/target/linux/ar71xx/files/arch/mips/ath79/mach-tl-wr720n-v3.c”增加 GPIO 和 LED 的对应关系，如下图所示（下面只列出关键部分，读者可查看打完补丁后的该文件）。

```
25 #define TL_WR720N_GPIO_LED_SYSTEM       27
26 #define TL_WR720N_GPIO_LED_WLAN         11
27 #define TL_WR720N_GPIO_LED_ETH1         13
28 #define TL_WR720N_GPIO_LED_ETH0         17
```

```
48 static struct gpio_led tl_wr720n_leds_gpio[] __initdata = {
49         {
50                 .name           = "tp-link:blue:system",
51                 .gpio           = TL_WR720N_GPIO_LED_SYSTEM,
52                 .active_low     = 1,
53         },{
54                 .name           = "tp-link:blue:wlan",
55                 .gpio           = TL_WR720N_GPIO_LED_WLAN,
56                 .active_low     = 0,
57         },{
58                 .name           = "tp-link:blue:eth1",
59                 .gpio           = TL_WR720N_GPIO_LED_ETH1,
60                 .active_low     = 1,
61         },{
62                 .name           = "tp-link:blue:eth0",
63                 .gpio           = TL_WR720N_GPIO_LED_ETH0,
64                 .active_low     = 1,
65         },
66 };
```

从上图可以看出，程序里面的 GPIO 引脚号是和开发板一一对应的。修改完成后，需要将该文件拷贝至“openwrt/build_dir/target-mips_34kc_uClibc-0.9.33.2/linux-ar71xx_generic/linux-3.10.49/arch/mips/ath79/”覆盖掉对应的文件，重新编译系统，方可生效。

然后编辑文件“openwrt/ target/linux/ar71xx/base-files/etc/uci-defaults/01_leds”，在相应位置修改为如下图所示

```
315 tl-wr720n-v3)
316         ucidef_set_led_netdev "lan" "LAN" "tp-link:blue:eth1" "eth1"
317         ucidef_set_led_netdev "wan" "WAN" "tp-link:blue:eth0" "eth0"
318         ucidef_set_led_netdev "wlan" "WLAN" "tp-link:blue:wlan" "wlan0"
319         ucidef_set_led_blink "system" "SYSTEM" "tp-link:blue:system" "1000" "1000"
```

然后编译固件，烧录固件。

2）添加 LED 对应的指示功能

我们通过以上步骤只是添加了 GPIO 和 LED 对应的关系和名称而已，要真正绑定 LED 的指示功能还需要做下面步骤

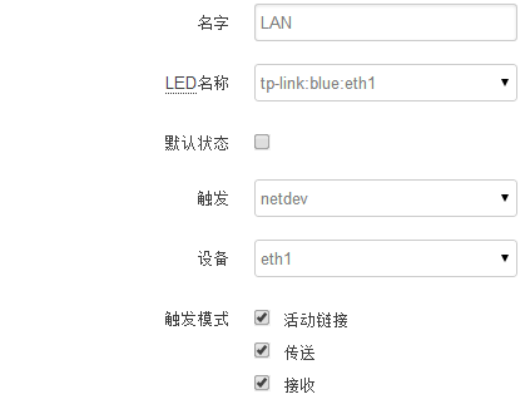
编辑开发板文件“/etc/config/system”配置如下图所示（开发板默认已添加）

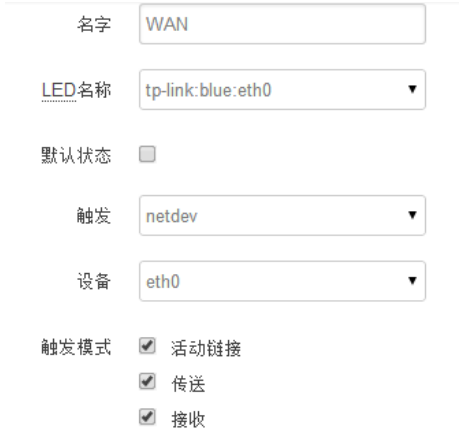
```
config led 'led_lan'
        option name 'LAN'
        option sysfs 'tp-link:blue:eth1'
        option trigger 'netdev'
        option dev 'eth1'
        option mode 'link tx rx'
        option default '0'

config led 'led_wan'
        option name 'WAN'
        option sysfs 'tp-link:blue:eth0'
        option trigger 'netdev'
        option dev 'eth0'
        option mode 'link tx rx'
        option default '0'

config led 'led_wlan'
        option name 'WLAN'
        option sysfs 'tp-link:blue:wlan'
        option trigger 'netdev'
        option dev 'wlan0'
        option mode 'link tx rx'
        option default '0'

config led 'led_system'
        option name 'SYSTEM'
        option sysfs 'tp-link:blue:system'
        option trigger 'timer'
        option delayon '1000'
        option delayoff '1000'
        option default '0'
```

以上选项对应 openwrt 配置页面里面“系统”-》“LED 配置”，如下图所示

LED配置

自定义LED的活动状态。





读者可以通过配置文件或 LED 配置页面进行配置，具体配置效果请读者查看开发板。

注：“LED 名称”并非一定要具有和这个名称一样的功能，同样一个 LED 可以通过“触发”选项选择其他功能，请读者自行测试。

4. 深入开发篇

在这一篇中，我们将深入学习 OpenWrt，包括如何搭建开发环境，编译 OpenWrt 源码，用命令行对 OpenWrt 系统进行各种操作，安装交叉工具链编译一个能在 OpenWrt 上运行的应用程序，并编译出 OpenWrt 现有的安装包做一些高级应用。

4.1. 开发前的硬件准备

JS9331 开发板一块，mini usb 数据线一条（用于 USB 串口调试），网线一条（用于网络测试、传输固件、应用程序等），开发板电源一个，红外遥控器一个（该配件可用于红外遥控实验，以上配件在购买 JS9331 开发板时均已配齐。用户还可以选配“红外遥控 LED 七彩灯”，进行更加直观的红外遥控实验）。

4.2. 搭建软件开发环境

目前大部分的 linux 开发都是在 PC 虚拟机上进行的，所采用的 linux 系统版本有 Ubuntu、Redhat、Debian、Fedora 等，这里我们在 windows 操作系统上利用 VMware + Ubuntu 来搭建虚拟机开发环境。

4.2.1. 安装虚拟机

见“JS9331 开发板配套资料\JS9331 开发板使用手册教程\虚拟机 VMware10 下载和安装详细教程.docx”或者百度。开发板配套资料中已带有虚拟机软件“JS9331 开发板配套资料\开发工具\VMware-workstation-full-10.0.3-1895310.zip”。

4.2.2. 安装 Ubuntu 系统

见“JS9331 开发板配套资料\JS9331 开发板使用手册教程\用 VMware 安装 Ubuntu_12.04 详细过程图解.docx”或者请百度。开发板配套资料中已带有 ubuntu 12.04 安装包“JS9331 开发板配套资料\开发工具 ubuntu-12.04.1-desktop-i386.iso”。

安装完 ubutu 系统后，一定要安装“VM tools”以便开发共享文档，方法见“JS9331 开发板配套资料\JS9331 开发板使用手册教程\Windows_7 与虚拟机下 UBUNTU10.10 共享文件夹.docx”。

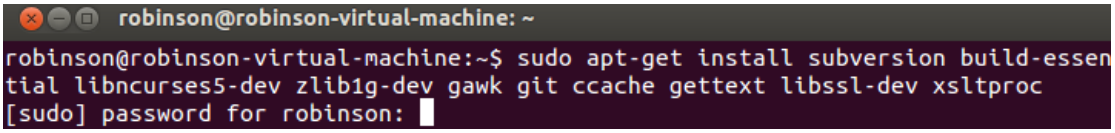
4.3. 搭建 OpenWrt 开发环境

下面我们就来讲解如何进行 OpenWrt 的开发。

4.3.1. 配置编译环境

为了能正常的编译 OpenWrt 源码，我们需要在 Ubuntu 系统中安装一些软件。打开一个终端，在 ubuntu 终端下面输入如下命令。

```
sudo apt-get install subversion build-essential libncurses5-dev zlib1g-dev gawk git ccache gettext libssl-dev xsltproc
```



系统将提示输入系统用户密码。输入密码完成后，回车，系统将开始安装指定软件，稍后安装完成。

4.3.2. 下载 OpenWrt 源码

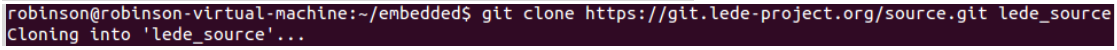
注意：以下操作均是在普通用户环境下操作，如果是在 root 用户环境下操作，可能会出现很多错误。建议读者将源码下载到当前用户的工作目录中（即~/目录下）进行编译，以防止编译源码时出现权限不够的情况。

4.3.2.1. 下载 OpenWrt 官方的 openwrt 源码（选做）

下载 openwrt 官方的源码的步骤是

1) 这里我们采用的是新的 openwrt 源码 LEDE，ubuntu 终端输入如下命令，下载源码。

```
git clone https://git.lede-project.org/source.git lede_source
```



等待代码下载完成。。。下载完成后，将在当前目录下生成一个名为"lede_source"的文件夹，里面存放的就是 OpenWrt(LEDE)的源码了。

2) 在 ubuntu 终端中输入以下命令：

```
cd lede_source                    //进入 openwrt 主目录
./scripts/feeds update -a         //更新安装包
./scripts/feeds install -a        //安装更新
```

提示：用官方原版 openwrt 源码编译出的固件也是可以在 JS9331 上运行的，但是有一些功能会不正常，比如按键，LED 指示灯等。

4.3.2.2. 用 JS9331 配套的 openwrt 源码

在“JS9331 开发板配套资料\开发板源码\openwrt 源码\lede_AR9331_zhuotk_source_32bit_xx.tar.bz2”(xx 是日期，采用 64 位 ubuntu 的用户请用 lede_AR9331_zhuotk_source_64bit_xx.tar.bz2)里面已经提供了下载好的 openwrt 源码，此源码已经更新了安装包、根据 JS9331 开发板功能进行了修改，读者可以将该安装包拷贝到当前登录 ubuntu 的用户的工作目录中（即~/目录下），执行

```
tar xjvf ./lede_AR9331_zhuotk_source_32bit_xx.tar.bz2 -C ./      //解压源码包
cd ./lede_AR9331_zhuotk_source_32bit      //进入源码目录
./scripts/feeds install -a      //安装更新的软件包
```

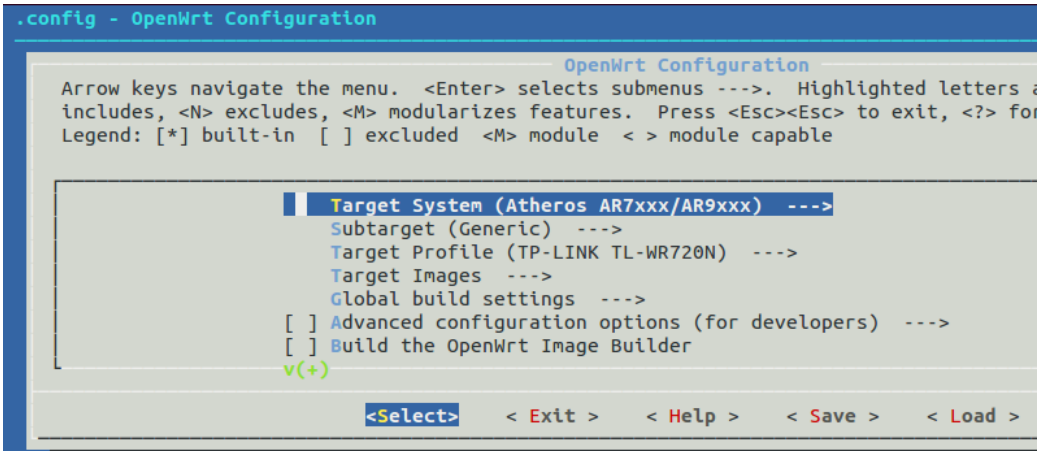
提示：这个源码编译出来的 openwrt 固件可以在 JS9331 开发板上正确运行。

4.3.3. make menuconfig 配置系统功能

OpenWrt 也具有像经典的 linux 的 make menuconfig 功能，用户用一种熟悉的方式对 OpenWrt 系统功能进行配置。在 OpenWrt 源码根目录下输入以下命令：

```
make menuconfig        //进入 OpenWrt 系统配置界面，更新.config
```

效果如下图所示



上图的这个界面，对于之前做 linux 开发的人员来说，应该是感觉很熟悉了。在这里，我们可以对 OpenWrt 的各种功能进行配置，包括选择芯片平台、选择各种各样功能的安装包、生成交叉工具链等等，非常多。

如果读者采用的是 JS9331 配套的源码，那么在这个页面中，用户不用做任何更改，直接选择上图“Exit”保存退出即可。采用官方源码的读者，因为配置的选项非常多，可以参考我们提供源码自行研究配置。

这里暂时不做系统配置介绍，以后会根据读者的反映，考虑是否增加介绍。

4.3.4. 编译 OpenWrt 源码

在 OpenWrt 源码根目录下输入以下命令：

```
make V=s        //编译源码，V=s 是用来生成编译信息的，方便用户查找出错原因。不建议加“-j x”（x 为数字）选项，那样可能会导致有时很难查找错误原因。
```

接下来就是“漫长的等待”。。。少则 2~3 个小时，多则 1~2 天。耗时的原因是因为 OpenWrt 系统要下载各种各样的源文件安装包，编译 linux 内核，生成交叉工具链，生成各种工具，生成文件系统等，工作量非常大（不用担心，这一切都是它自动完成的）。其中最耗时也最容易出错的就是下载各种源码包了，因为这些源码包绝大部分都是从国外下载的，所以速度很慢，而且非常容易出现无法下载的情况，最终导致 OpenWrt 编译报错。

提示：在“JS9331 开发板配套资料\开发板源码\openwrt 源码\dl”下，我们已经将这些需要下载的源码包下载好了，用户只需要将里面的文件拷贝到“openwrt 源码根目录\dl”目录下即可。用户如果想自己下载源码包，推荐用 VPN 软件（俗称翻墙软件）。具体 ubuntu 设置 VPN 方法请参考“JS9331 开发板配套资料\开发板使用手册教程\ubuntu 挂 VPN（green vpn）.docx”。

最后，如果编译成功会出现类似下图的提示。

```
Generating index for package ./luci-proto-ipv6_git-17.219.28590-1fdad26-1_all.ipk
Generating index for package ./luci-proto-ppp_git-17.219.28590-1fdad26-1_all.ipk
Generating index for package ./luci-theme-bootstrap_git-17.219.28590-1fdad26-1_all.ipk
Generating index for package ./luci_git-17.219.28590-1fdad26-1_all.ipk
Generating index for package ./rpcd-mod-rrdns_20170710_mips_24kc.ipk
Generating index for package ./libsqlite3_3190300-1_mips_24kc.ipk
Generating index for package ./minicom_2.7-3_mips_24kc.ipk
Signing package index...
make[2]: Leaving directory `/home/robinson/embedded/lede_AR9331_zhuotk_source_32bit_test'
export MAKEFLAGS= ;make -w -r checksum
make[2]: Entering directory `/home/robinson/embedded/lede_AR9331_zhuotk_source_32bit_test
make[2]: Leaving directory `/home/robinson/embedded/lede_AR9331_zhuotk_source_32bit_test'
make[1]: Leaving directory `/home/robinson/embedded/lede_AR9331_zhuotk_source_32bit_test'
```

编译完成后的 OpenWrt 固件可以在“openwrt 源码根目录/bin/targets/ar71xx/generic/”下找到。

```
robinson@robinson-virtual-machine:~/embedded/lede_AR9331_zhuotk_source_32bit_test$ ls bin/targets/ar71xx/generic/
config.seed                                        lede-ar71xx-generic-vmlinux.bin
lede-ar71xx-generic-device-tl-wr720n-v3.manifest   lede-ar71xx-generic-vmlinux.elf
lede-ar71xx-generic-root.squashfs                  lede-ar71xx-generic-vmlinux.lzma
lede-ar71xx-generic-tl-wr720n-v3-squashfs-factory.bin  lede-ar71xx-generic-vmlinux-lzma.elf
lede-ar71xx-generic-tl-wr720n-v3-squashfs-sysupgrade.bin  packages
lede-ar71xx-generic-uImage-lzma.bin                sha256sums
```

上图中的“lede-ar71xx-generic-tl-wr720n-v3-squashfs-sysupgrade.bin”即为 LEDE 固件。

LEDE 编译后的安装包（IPK）在“openwrt 源码根目录/bin/targets/ar71xx/generic/packages/”和“openwrt 源码根目录/bin/packages/”中，请用户自行查看。

4.3.5. 刷新 OpenWrt 固件

有关如何刷新 OpenWrt 固件，在《JS9331 开发板使用手册》中的“openwrt 固件烧写说明”一节中已有叙述，这里不再赘述。

4.4. 生成交叉工具链

用户如果想自己编译获得 openwrt 的交叉工具链，只需要在 openwrt 的 menuconfig 顶层配置界面中，勾选上“Package the LEDE-based Toolchain”，如下图所示。

```
[ ] Build the LEDE Image Builder
[ ] Build the LEDE SDK
[*] Package the LEDE-based Toolchain
[ ] Image configuration  --->
```

然后保存退出，再 make V=s，生成的交叉工具链将在“openwrt 源码根目录/bin/targets/ar71xx/generic/ lede-toolchain-ar71xx-generic_gcc-5.4.0_musl.Linux-i686.tar.bz2”（32 位交叉编译工具）下找到。

提示：不方便编译的用户，可以在“JS9331 开发板配套资料\JS9331 开发板固件镜像安装包\交叉工具链”中找到编译好的有 32 位也有 64 位的 openwrt 交叉工具链，可以直接拿来使用。

4.5. 安装交叉工具链

openwrt 交叉工具链和一般的 linux 软件包一样，需要将其解压到需要安装的目录中，然后设置环境变量即可。下面是安装步骤。

1）解压交叉工具链压缩包

这里我们演示将交叉工具链安装到 ubuntu 的“/opt”目录下。首先切换到 openwrt 源码的根目录下，输入如下命令：

```
sudo tar xjvf ./bin/targets/ar71xx/generic/lede-toolchain-ar71xx-generic_gcc-5.4.0_musl.Linux-i686.tar.bz2 -C /opt
```

提示输入超级用户密码后，系统将交叉工具链压缩包解压到 ubuntu 系统的/opt/目录下。

提示：有关 tar 解压命令详细介绍，在“JS9331 开发板配套资料\学习资料\linux 应用开发学习资料\嵌入式 Linux 应用程序开发详解\嵌入式 Linux 应用程序开发详解-第 2 章 Linux 基础命令.pdf”有介绍。

这里告诉大家一个非常实用的小技巧，如果用手工敲上面的命令，非常的费时费力，还很容易出错，你可以敲完“sudo tar jxvf ./b”后按“Tab”键，系统将自动补全为“sudo tar jxvf ./bin”，后面的文件夹和文件也是同理，非常的省时省力。

2）设置环境变量

```
sudo vi /etc/bash.bashrc
```

在最后一行添加

```
export PATH=/opt/lede-toolchain-ar71xx-generic_gcc-5.4.0_musl.Linux-i686/toolchain-mips_24kc_gcc-5.4.0_musl/bin:$PATH
export STAGING_DIR=/your_openwrt_path/staging_dir
```

效果如下

```
export PATH=/opt/lede-toolchain-ar71xx-generic_gcc-5.4.0_musl.Linux-i686/toolchain-mips_24kc_gcc-5.4.0_musl/bin/:$PATH
export STAGING_DIR=/your_openwrt_path/staging_dir
```

注意上面这个“STAGING_DIR”变量中的“your_openwrt_path”是读者实际放 openwrt 源码的根目录，如果这个“STAGING_DIR”变量不设置的话，会在用交叉工具链编译文件时有警告，但是不影响编译结果。

最后保存退出。

接着在终端执行以下命令：

```
source /etc/bash.bashrc
```

3）检查是否安装成功

```
mips-openwrt-linux-gcc -v
```

此时应打印出交叉编译工具的一些信息，表示安装成功，如下图所示。

```
robinson@robinson-virtual-machine:~/embedded/openwrt$ mips-openwrt-linux-gcc -v
Reading specs from /opt/OpenWrt-Toolchain-ar71xx-for-mips_34kc-gcc-4.8-linaro_uClibc-0.9.33.2/toolchain-mips_34kc
_gcc-4.8-linaro_uClibc-0.9.33.2/bin/../lib/gcc/mips-openwrt-linux-uclibc/4.8.3/specs
COLLECT_GCC=mips-openwrt-linux-uclibc-gcc.bin
COLLECT_LTO_WRAPPER=/opt/OpenWrt-Toolchain-ar71xx-for-mips_34kc-gcc-4.8-linaro_uClibc-0.9.33.2/toolchain-mips_34k
c_gcc-4.8-linaro_uClibc-0.9.33.2/bin/../libexec/gcc/mips-openwrt-linux-uclibc/4.8.3/lto-wrapper
Target: mips-openwrt-linux-uclibc
Configured with: /home/robinson/embedded/openwrt_barri/build_dir/toolchain-mips_34kc_gcc-4.8-linaro_uClibc-0.9.33
.2/gcc-linaro-4.8-2014.04/configure --with-bugurl=https://dev.openwrt.org/ --with-pkgversion='OpenWrt/Linaro GCC
4.8-2014.04 r43380' --prefix=/home/robinson/embedded/openwrt_barri/staging_dir/toolchain-mips_34kc_gcc-4.8-linaro
```

.....

```
/opt/lede-toolchain-ar71xx-generic_gcc-5.4.0_musl.Linux-i686/toolchain-mips_24kc_gcc-5.4.0_musl/bin/../lib/gcc/mips-openwrt-linux-musl/5.4.0/../
../../../mips-openwrt-linux-musl/lib/crt1.o: In function `_start_c':
/home/robinson/embedded/lede_CC_AR9331/build_dir/toolchain-mips_24kc_gcc-5.4.0_musl/musl-1.1.16/crt/crt1.c:17: undefined reference to `main'
/home/robinson/embedded/lede_CC_AR9331/build_dir/toolchain-mips_24kc_gcc-5.4.0_musl/musl-1.1.16/crt/crt1.c:17: undefined reference to `main'
collect2: error: ld returned 1 exit status
```

上图提示的错误，不用管。

4.6. 编译第一个“Hello World”程序

在 Ubuntu 下，切换到任意目录，输入以下命令：

```
vi hello_world.c          //用 vi 新建一个名为 hello_world.c 文件
```

并输入以下内容。

```
#include <stdio.h>
int main(char argc, char *argv[])
{
   int i = 1;
```

```
    while(1){
        //1~10 循环
        printf("Hello world!!!%d\n",i);//打印内容
        if (i < 10){
            i++;
        }else{
            i = 1;
        }
        sleep(1);// 一秒钟打印一次
    }
    return    0;
}
```

然后

mips-openwrt-linux-gcc hello_world.c -o hello_world　　//用 mips-openwrt-linux-gcc 编译“hello_world.c”文件，并生成“hello_world”可以执行文件。

最后我们用 winscp 将文件传输到 JS9331 开发板的“/tmp”目录下，在开发板的串口终端下，执行如下命令。

chmod +x /tmp/hello_world　　//“hello_world”加上执行权限

/tmp/hello_world　　//执行“hello_world”可执行文件

效果如下图所示。

```
root@JoySince:/# /tmp/hello_wold
Hello world!!!1
Hello world!!!2
Hello world!!!3
Hello world!!!4
Hello world!!!5
Hello world!!!6
Hello world!!!7
Hello world!!!8
Hello world!!!9
Hello world!!!10
Hello world!!!1
Hello world!!!2
```

用户可以用“Ctrl+c”组合键，停止程序的运行。

4.7. 编译第一个“Hello World”应用程序 IPK 安装包

openwrt 一个比较重要的特点就是它采用 ipk 包的形式安装软件。有点像是 windows 下面的安装包一样，用户只需用简单的命令就可以将 ipk 安装包安装到 openwrt 系统中，非常方便。在“安装 IPK 包”一节中，我们已经介绍过如何通过网络下载安装 openwrt 的 ipk 安装包，但那是 openwrt 官方已经为我们编译好的，下面来介绍一下如何制作编译一个简单的安装包。

切换到 openwrt 根目录，然后执行下列命令：

cd ./package/utils　　//进入 package/utils 目录

mkdir hello_world　　//创建一个名为“hello_world”的文件夹，用于放置安装包源码。

cd hello_world　　//进入“hello_world”目录

mkdir src　　//新建一个名为“src”的目录用于放置源码

vi src/hello_world.c　　//在 src 目录下新建一个名为 hello_world.c 文件

输入 hello_world.c 中的内容同 4.6 节中一致，这里不再赘述，编辑完成后，保存退出。

vi src/Makefile　　//在 src 目录下新建一个 Makefile

输入以下内容：

```
all: hello_world
hello_world: hello_world.o
	$(CC) $(LDFLAGS) hello_world.o -o hello_world
helloworld.o: hello_world.c
	$(CC) $(CFLAGS) -c hello_world.c
clean:
	rm *.o hello_world
```

保存退出，然后再

```
vi Makefile                    //当前目录下新建一个 Makefile 文件
```

输入以下内容：

```
include $(TOPDIR)/rules.mk

PKG_NAME:=hello_world
PKG_VERSION:=1.0
PKG_BUILD_DIR:= $(BUILD_DIR)/$(PKG_NAME)

include $(INCLUDE_DIR)/package.mk

define Package/hello_world
	SECTION:=base
	CATEGORY:=Utilities
	TITLE:=Hello world -prints a hello world message
endef

define Package/hello_world/description
	If you can't figure out what this program does, you're probably
	brain-dead and need immediate medical attention.
endef

define Build/Prepare
	mkdir -p $(PKG_BUILD_DIR)
	$(CP) ./src/* $(PKG_BUILD_DIR)/
endef

define Package/hello_world/install
	$(INSTALL_DIR) $(1)/bin
	$(INSTALL_BIN) $(PKG_BUILD_DIR)/hello_world $(1)/bin/
endef

$(eval $(call BuildPackage,hello_world))
```

接下来回到 openwrt 根目录,执行命令

```
make menuconfig
```

并选择我们已经加进去的安装包。

```
Utilities    --->
    <M> hello_world..... Hello world -prints a hello world message
```

保存退出。

```
make V=s
```

等待编译完成后，我们就可以在以下路径找到生成的安装包

“openwrt/bin/ar71xx/packages/base/hello_world_1.0_ar71xx.ipk”。

安装包安装完成后，直接在串口终端输入：

```
hello_world
```

结果和之前一样。

```
root@ZhuoTK:/# hello_world
Hello world!!!1
Hello world!!!2
Hello world!!!3
Hello world!!!4
Hello world!!!5
Hello world!!!6
```

本实验的源码在“JS9331 开发板配套资料\开发板源码\IPK 实验源码\hello_world”目录中。

4.8. 编译一个“gpio 控制”驱动程序 IPK 安装包

之前我们介绍过“GPIO 简单控制”，不过那种方法是通过命令控制或脚本控制的，在运行过程中效率比较低，本节介绍用驱动程序控制，效率会高一些，是一种控制 GPIO 比较直接的方式。

复制“JS9331 开发板配套资料\开发板源码\gpio_control_driver”文件夹到“openwrt 源码根目录/package/kernel/”目录下，然后执行

```
make menuconfig
```

进入配置页面，选上如下配置

```
Kernel modules    --->
    Other modules    --->
        <M> kmod-gpio_control_driver..... Driver for JS9331/JS7628 gpios control
```

保存退出，执行

```
make V=s
```

重新编译源码，如果编译没有问题的话，找到编译生成的 IPK

“kmod-gpio_control_driver_4.4.79-1_mips_24kc.ipk”，将此 IPK 传输到开发板并安装。安装完成后，执行 lsmod，如下图所示

```
gpio_control_driver     1262  0
```

可以看到“gpio_control_driver”内核模块已经正确的插入内核中运行，并且出现了“/dev/gpio_control”这个设备。

```
root@ZhuoTK:/# ls /dev
audio                   mtd0
bus                     mtd0ro
console                 mtd1
cpu_dma_latency         mtd1ro
dsp                     mtd2
full                    mtd2ro
fuse                    mtd3
gpio_control            mtd3ro
i2c-0                   mtd4
```

参考文档：

“JS9331 开发板配套资料\学习资料\linux 驱动开发学习资料\Linux 设备驱动开发详解-宋宝华.pdf”

“JS9331 开发板配套资料\学习资料\openwrt 学习资料\Creating packages [OpenWrt Wiki].pdf”

“JS9331 开发板配套资料\学习资料\linux 应用开发学习资料\gpio.txt”

4.9. 编译一个“gpio 控制”应用程序 IPK 安装包

上一节我们完成了“gpio 控制”驱动的安装，这一节我们来完成“gpio 控制”应用程序的安装，以便我们在应用层控制 GPIO。

复制“JS9331 开发板配套资料\开发板源码\gpio_control_app”到“openwrt 源码根目录/package/utils/”目录下，并执行

```
make menuconfig
```

进入配置页面，选上如下配置

```
Utilities  --->
    <M> gpio_control_app..... Control gpios
```

保存退出，执行

```
make V=s
```

重新编译源码，如果编译没有问题的话，找到编译生成的 IPK

“gpio_control_app_1.0_mips_24kc.ipk”，将此 IPK 传输到开发板并安装。安装完成后，执行

```
gpio_control_app -h
```

结果如下图所示

```
root@ZhuoTK:/# gpio_control_app -h
gpio_contrl_app usage: demo1 <GPIO number>
```

出现上图，说明已经正常安装。这里我们还是采用之前“GPIO 简单控制”一节中用到的 GPIO22 进行测试。测试 GPIO22 命令如下

```
gpio_control_app  demo1  22
```

执行以上命令后，GPIO22 将会出现“1 秒高电平 1 秒低电平 1 秒高电平...”的循环（ctr+c 退出），读者可以用万用表或连接到 LED 进行测试（参考“GPIO 简单控制”一节）。

参考文档：

“JS931 开发板配套资料\学习资料\openwrt 学习资料\Creating packages [OpenWrt Wiki].pdf”

4.10. 编译一个“网络通讯服务器”应用程序 IPK 安装包

这里介绍一个比较简单的“网络通讯服务器”，它通过 TCP socket 连接，接收来自 TCP socket 客户端的连接，最终显示接收到的信息。

复制“JS9331 开发板配套资料\开发板源码\net_control_server”到“openwrt 源码根目录/package/utils/”目录下，并执行

```
make menuconfig
```

进入配置页面，选上如下配置

```
Utilities  --->
    <M> net_control_server..... net control server
```

保存退出，执行

```
make V=s
```

重新编译源码，如果编译没有问题的话，找到编译生成的 IPK

“net_control_server_1.0_mips_24kc.ipk”，将此 IPK 传输到开发板并安装。安装完成后，执行

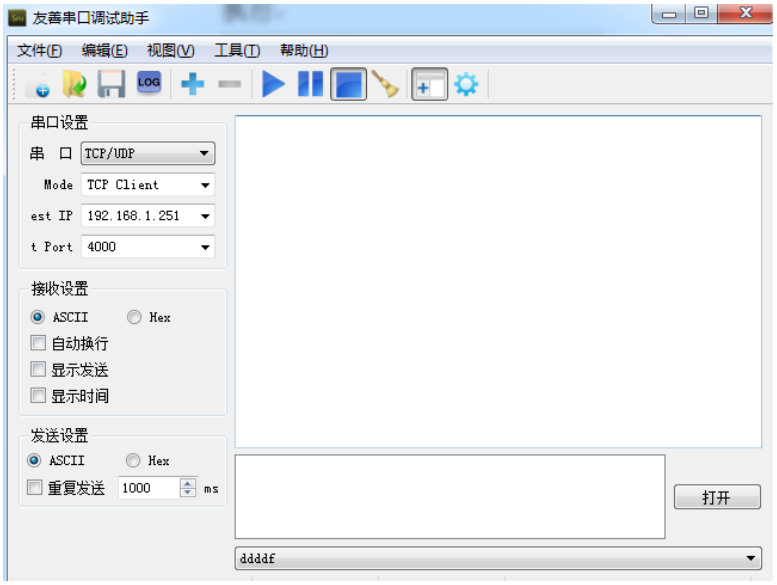
```
net_control_server  tcp_server  4000
```

如下图所示

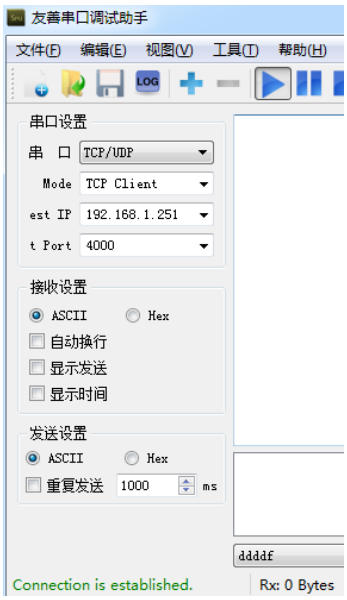
```
root@ZhuoTK:/# net_control_server tcp_server 4000
***net_control_server***
listening...
```

此时进入“网络监听状态”等待客户端的连接。。。

我们可以用之前介绍的“友善串口调试助手”做为“客户端”连接这个服务器。配置如下



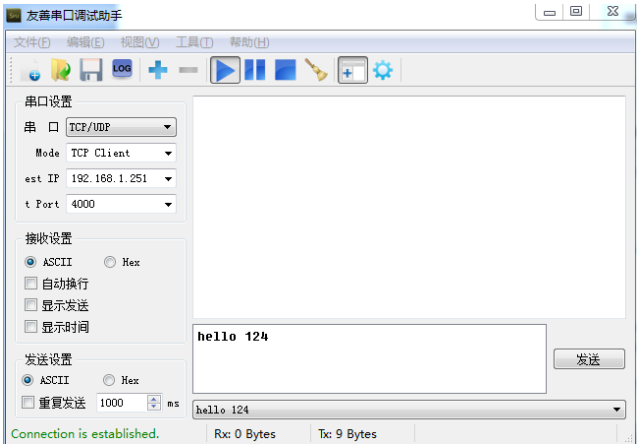
然后点击  连接服务器，结果如下图



接着开发板会出现类似如下提示

```
new client_fd = 4
New thread 2009199920
```

说明已经连接上了。我们可以用这个“友善串口调试助手”尝试发送一些信息给开发板，填入“hello 124”然后点“发送”，结果开发板上出现了相应的信息，如下图所示。



以上介绍了简单的通讯，读者可以阅读修改源码，实现更复杂的功能。

参考文档：

“JS9331 开发板配套资料\学习资料\linux 应用开发学习资料\嵌入式 Linux 应用程序开发详解”

“JS9331 开发板配套资料\学习资料\openwrt 学习资料\ Creating packages [OpenWrt Wiki].pdf”

4.11. 编译一个“网络通讯客户端”应用程序 IPK 安装包

这里介绍一个“网络通讯客户端”，它通过 TCP socket 连接到服务端，连接上服务端后发送一个消息给服务器，并等待服务器给它发送数据。

复制“JS9331 开发板配套资料\开发板源码\net_control_client”到“openwrt 源码根目录/package/utils/”目录下，并执行

```
make menuconfig
```

进入配置页面，选上如下配置

```
Utilities  --->
    <M> net_control_client..... net control client
```

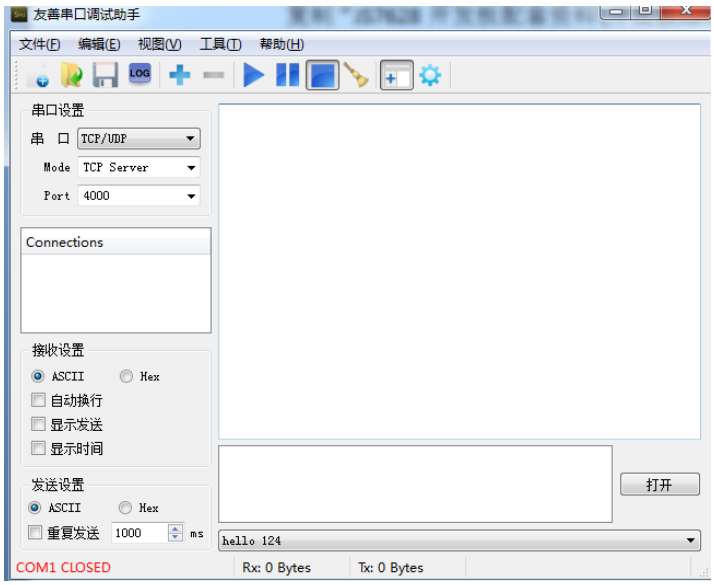
保存退出，执行

```
make V=s
```

重新编译源码，如果编译没有问题的话，找到编译生成的 IPK

“net_control_client_1.0_mips_24kc.ipk”，将此 IPK 传输到开发板并安装。

我们可以用之前介绍的“友善串口调试助手”做为“服务端”让开发板连接。配置如下



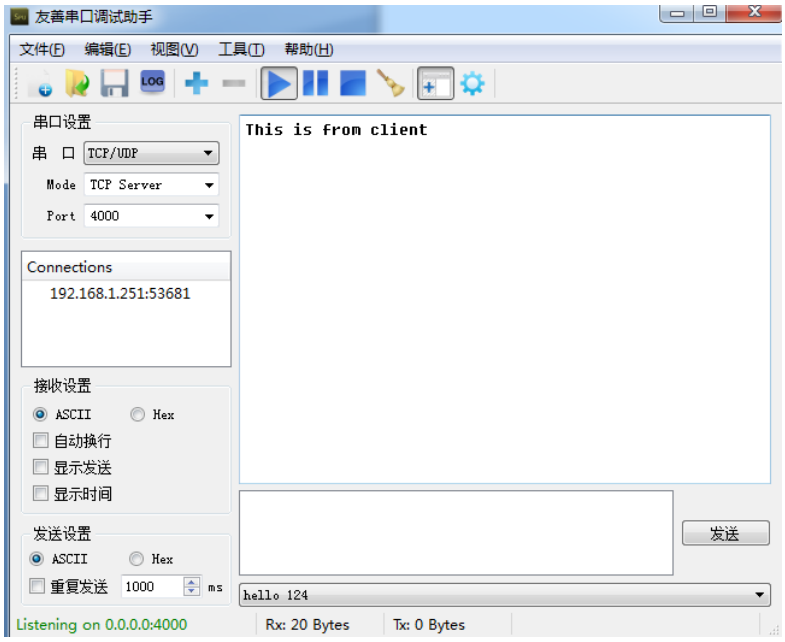
开发板执行

net_control_client tcp_connect　192.168.1.xxx　4000　　#192.168.1.xxx 为读者自己电脑的 IP

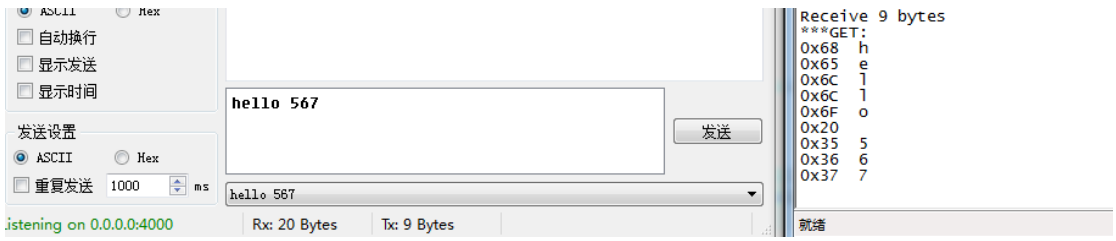
结果如下图所示（这里作者电脑 IP 是“192.168.1.110”）

```
root@ZhuoTK:/# net_control_client tcp_connect 192.168.1.110 4000
socket id = 3
***connect to 192.168.1.110.....***
***connected***
```

出现上图提示表示开发板的“客户端”已经连接到“友善串口助手”的服务端了，并且该串口助手也显示了开发板发送的消息“This is from client”，如下图所示



串口助手发送“hello 567”开发板将会有相应的显示，如下图所示



如果读者有两块开发板的话还可以实现，开发板“客户端”连接另外一块开发板“服务端”，这个留给读者自行测试。

参考文档：

“JS9331 开发板配套资料\学习资料\linux 应用开发学习资料\嵌入式 Linux 应用程序开发详解”

“JS9331 开发板配套资料\学习资料\openwrt 学习资料\ Creating packages [OpenWrt Wiki].pdf”

4.12. LUCI 界面修改

请读者参考“JS9331 开发板配套资料\学习资料\openwrt 学习资料\ openwrt 学习资料\luci 学习笔记.pdf”和“JS9331 开发板配套资料\学习资料\openwrt 学习资料\【OpenWRT 之旅】LuCI 探究.pdf”。读者也可以参考学习“JS9331 开发板配套资料\开发板源码\温度传感器源码\ temp_sensor.tar.bz2”

LUCI 有关介绍 https://github.com/openwrt/luci/wiki/Documentation

5. Openwrt 学习网站

1）恩山论坛（推荐）：http://www.right.com.cn/forum/forum.php，这个国内在 openwrt 讨论方面算是比较多的，很多有关 openwrt 的问题、教程在这里都能找到。

2）openwrt 官方网站：www.openwrt.org，wiki.openwrt.org，里面有最全的 openwrt 资料。

3）openwrt 中文网站（非 openwrt 官方）：http://www.openwrt.org.cn/，这个网站是国人建的，但在里面讨论的人现在好像已经不是很多了，可以找到许多的 openwrt 资料。

4）百度。

6. 常见问题及解答

7. 修改说明

版本	时间	修改说明
V1.0	2015.06.10	JS9331 开发板入门教程初始版本。
V1.1	2015.07.07	增加了一部分 openwrt 的高级应用实验。
V1.2	2015.07.19	增加了一部分 openwrt 的高级应用实验。
V1.3	2015.07.30	修正了一些遗漏的说明，修正了复制乱码的问题。
V1.4	2015.09.10	增加了“实现迅雷远程下载”实验
V1.5	2015.09.20	增加“挂载摄像头实现远程监控”、“GPIO 控制”、“按键功能”、“LED 指示”等实验.增加一些安装 ubuntu 的提示。
V1.6	2015.11.09	增加了一些常见问题的解决方法。
V1.7	2016.01.11	增加了编译 openwrt 源码的说明。 增加了“无线二级路由器”的说明。 修改了有关“客户端模式”、“中继模式”、“桥接模式”的说明。
V1.8	2016.03.09	修改了安装交叉编译工具的说明。 添加了“二级无线路由器模式设置”中有关上级无线路由器访问开发板的说明。

		补充了有关挂载 U 盘的一些说明。 修正了有关源码编译问题的说明。 添加了“挂载 4G 网卡上网”一节。
V1.9	2016.04.23	添加了有关 LUCI 官方地址的说明。 添加了“开发板挂 VPN”的实验。 添加了“开发板拨打网络电话”的实验。 添加了有关“ser2net”开机启动的说明。
V1.10	2016.06.20	修正了"开发板拨打网络电话"的一些说明。 修正了交叉编译工具的说明。
V1.11	2016.07.31	修改了有关 U 盘挂载 VFAT 时需要的安装的安装包说明
V1.12	2017.06.01	修改了有关“打造无线音乐播放器”的说明。 添加了 ubuntu 挂 VPN 文档的说明。
V1.13	2017.07.01	修改了挂载 4G 网卡的说明
V1.14	2017.08.01	修改了摄像头监控说明 修改了“挂载 U 盘”的说明 修改了“挂载 4G 网卡”的说明 添加了“安装 IPK 包”的说明
V1.15	2017.08.06	将文档一些说明修改为 LEDE 系统说明
V1.16	2017.12.31	在“远程访问开发板”一节中添加了“服务器中转”（内网穿透）的例子。 修正了英文字体。
V1.17	2018.03.28	添加了：编译一个“gpio 控制”驱动程序 IPK 安装包、编译一个“gpio 控制”应用程序 IPK 安装包、编译一个“网络通讯服务器”应用程序 IPK 安装包、编译一个“网络通讯客户端”应用程序 IPK 安装包、打造本地音乐播放器，这些章节。
V1.18	2018.07.30	修正了“二级无线路由器”一节的说明
V1.19	2018.08.08	修改了“安装 IPK 包”一节的说明。修正了“服务器中转”的说明
V1.20	2018.11.18	修正了 NFS 挂载的说明。 添加了“挂 VPN”里面的一些说明。 添加了有关修改 ser2net 配置文件编辑器的说明。