

JS7628 开发板 openwrt 入门教程

v1.4.4(2018.09.04)

杭州卓钛科技有限公司

网站: www.zhuotk.com

前言

这份《JS7628 开发板 OpenWrt 入门教程》是卓钛科技编写的，主要介绍基于 JS7628 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	5
1.1. 什么是 OpenWrt	5
1.2. 为什么学 OpenWrt	5
1.3. OpenWrt 版本发展史	5
2. 开发学习概述	6
3. 入门应用篇	7
3.1. 进入 OpenWrt 系统	7
3.2. 系统菜单	7
3.2.1. “状态”菜单项	8
3.2.2. “系统”菜单项	9
3.2.3. “网络”菜单项	14
3.2.4. “退出”菜单项	16
3.3. 基本硬件功能测试	17
3.3.1. 添加修改按键功能	17
3.3.1.1. 添加按键	17
3.3.1.2. 添加按键功能	18
3.3.2. 添加修改 LED 指示功能	19
3.3.2.1. 添加 LED 指示	19
3.3.2.2. 修改 LED 指示	21
3.3.3. TTL/RS232 串口测试说明	21
3.3.3.1. 串口介绍	21
3.3.3.2. 串口测试	22
3.3.4. TTL/RS485 串口测试说明	23
3.3.4.1. 串口介绍	23
3.3.4.2. 串口测试	24
3.3.5. GPIO 简单控制	24
3.4. 玩转 OpenWrt 高级功能	25
3.4.1. 电脑和开发板互传文件	25
3.4.1.1. 用 scp 协议传输文件	25
3.4.1.2. 用 NFS 服务传输文件	26
3.4.1.3. 用 FTP 服务传输文件（搭建 FTP 服务器）	28
3.4.2. 安装 IPK 包	29
3.4.3. 路由模式设置	29
3.4.4. 拨号上网（路由器模式）	30
3.4.5. 多拨功能	31
3.4.6. AP 模式设置	32
3.4.7. sta 模式设置	33
3.4.8. 挂载 4G 网卡上网	34
3.4.9. 开发板挂 VPN	37
3.4.10. 网络串口透传	40
3.4.11. 挂载 U 盘、micro SD（TF）卡	41
3.4.12. 远程访问开发板	42

3.4.12.1.	路由器端口映射访问.....	42
3.4.12.2.	服务器中转访问（实现内网穿透）.....	44
3.4.13.	打造本地音乐播放器.....	51
3.4.14.	挂载摄像头实现远程监控.....	51
4.	深入开发篇.....	53
4.1.	开发前的硬件准备.....	53
4.2.	搭建软件开发环境.....	54
4.2.1.	安装 VMware 虚拟机软件.....	54
4.2.2.	安装 Ubuntu 系统.....	54
4.3.	搭建 OpenWrt 开发环境.....	54
4.3.1.	配置编译环境.....	54
4.3.2.	下载 OpenWrt 源码.....	54
4.3.2.1.	下载 OpenWrt 官方的 openwrt 源码（选做）.....	55
4.3.2.2.	用 JS7628 配套的 openwrt 源码.....	55
4.3.3.	make menuconfig 配置系统功能.....	55
4.3.3.1.	配置界面简介.....	55
4.3.3.2.	有关“IOT-device”、“IOT-gateway”模式的配置.....	56
4.3.4.	编译 OpenWrt 源码.....	56
4.3.5.	刷新 OpenWrt 固件.....	57
4.4.	生成交叉工具链.....	57
4.5.	安装交叉工具链.....	58
4.6.	openwrt 源码简介.....	59
4.7.	配置 JS7628 开发板硬件默认功能.....	61
4.8.	编译第一个“Hello World”应用程序.....	64
4.9.	编译第一个“Hello World”应用程序 IPK 安装包.....	65
4.10.	编译一个“gpio 控制”驱动程序 IPK 安装包.....	67
4.11.	编译一个“gpio 控制”应用程序 IPK 安装包.....	67
4.12.	编译一个“网络通讯服务器”应用程序 IPK 安装包.....	68
4.13.	编译一个“网络通讯客户端”应用程序 IPK 安装包.....	70
4.14.	LUCI 界面修改.....	72
5.	Openwrt 学习网站.....	72
6.	常见问题及解答.....	72
7.	历史版本说明.....	72

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,一些新的开发人员加入了这项目 ,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。

版本时间轴

版本号	发布日期	代号
测试版本（不稳定版本）		
	持续更新	trunk
稳定版本		
15.05.1	2016 年 3 月	Chaos Calmer
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。

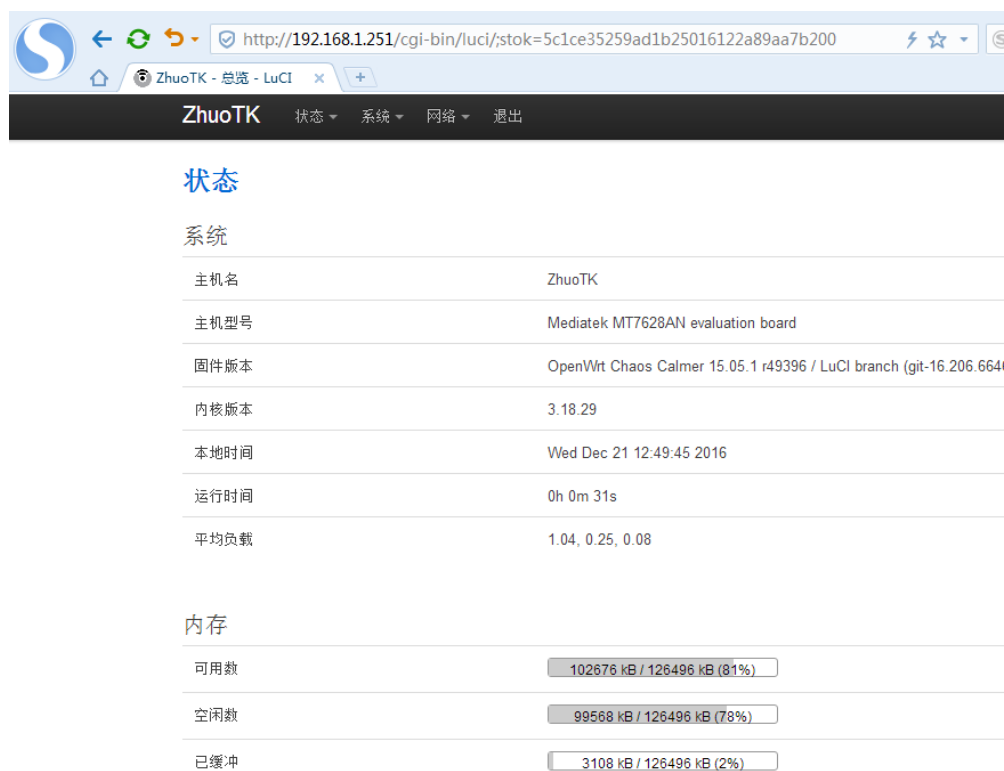
3. 入门应用篇

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，这些知识不是必须的。下面我们就以 JS7628 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 JS7628 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《JS7628 开发使用手册》中的“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（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。
- 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。
- 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。

系统进程

系统中正在运行的进程和其状态信息。

PID	用户名	进程命令	CPU 使用 率(%)	内存 使用 率(%)	挂起	关闭	强制关闭
1	root	/sbin/procd	0%	1%	挂起	关闭	强制关闭
2	root	[kthreadd]	0%	0%	挂起	关闭	强制关闭

- 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。

- a) 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

ZhutoTK 状态 系统 网络 退出

软件包

动作 配置

```
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/base/Packages.gz.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/base/Packages.sig.  
Signature check failed.  
Remove wrong Signature file.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/luci/Packages.gz.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/luci/Packages.sig.  
Signature check failed.  
Remove wrong Signature file.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/management/Packages.gz.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/management/Packages.sig.  
Signature check failed.  
Remove wrong Signature file.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/packages/Packages.gz.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/packages/Packages.sig.  
Signature check failed.  
Remove wrong Signature file.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/routing/Packages.gz.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/routing/Packages.sig.  
Signature check failed.  
Remove wrong Signature file.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/telephony/Packages.gz.  
Downloading http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/telephony/Packages.sig.  
Signature check failed.  
Remove wrong Signature file.
```

wget: bad address 'downloads.openwrt.org'

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。

接口 - LAN

配置网络接口信息。

一般设置

基本设置 高级设置 物理设置 防火墙设置

状态


br-lan

运行时间: 0h 2m 59s

MAC 地址: 00:CA:11:22:33:44

接收: 60.30 KB (653 数据包)

发送: 675.92 KB (964 数据包)

IPv4: 192.168.1.251/24

IPv6: fde5:8108:25ac::1/60

协议

静态地址

IPv4 地址

192.168.1.251

IPv4 子网掩码

255.255.255.0

IPv4 网关

192.168.1.1

IPv4 广播

使用自定义的 DNS 服务器

192.168.1.1

最后别忘了点击当前页面下方的“保存&应用”按钮。



这样，开发板就可以和外网进行通信了。

- b) 然后点击“刷新列表”按钮，系统会从“配置”中设置好的地址下载软件包列表。
- c) 更新完成后，读者可以直接在“下载并安装软件包”一栏填写需要安装的软件包，如下图

下载并安装软件包:

也可以点击“查找软件包”按钮，系统会列出可用的软件包。

提示：为了保证软件兼容性，“kmod*”的软件包尽量用自己编译的或者是资料里面提供的，否则有可能造成安装的软件功能不正常。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“http://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages”进行安装包的下載,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）。
- c. 换软件包源，读者可以把“软件包”->“配置”里面的“Distribution feeds”栏中所有的“downloads.openwrt.org”改为“openwrt.proxy.ustclug.org”（国内教育网的 openwrt 软件源），效果如下，

Distribution feeds

Build/distribution specific feed definitions. This file will NOT be preserved in any sysupgrade.

```
src/gz chaos_calmer_base http://openwrt.proxy.ustclug.org/chaos_calmer/15.05.1/ramips/mt7628/packages/base
src/gz chaos_calmer_luci http://openwrt.proxy.ustclug.org/chaos_calmer/15.05.1/ramips/mt7628/packages/luci
src/gz chaos_calmer_management http://openwrt.proxy.ustclug.org/chaos_calmer/15.05.1/ramips/mt7628/packages/management
src/gz chaos_calmer_packages http://openwrt.proxy.ustclug.org/chaos_calmer/15.05.1/ramips/mt7628/packages/packages
src/gz chaos_calmer_routing http://openwrt.proxy.ustclug.org/chaos_calmer/15.05.1/ramips/mt7628/packages/routing
src/gz chaos_calmer_telephony http://openwrt.proxy.ustclug.org/chaos_calmer/15.05.1/ramips/mt7628/packages/telephony
```

修改完成后，点击“提交”，这样更新软件包速度和成功率大大提高。

- 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。

LED配置

自定义LED的活动状态。

名字	<input type="text" value="system"/>
LED名称	<input type="text" value="mediatek:green:system"/>
默认状态	<input type="checkbox"/>
触发	<input type="text" value="timer"/>
通电时间	<input type="text" value="1000"/>
关闭时间	<input type="text" value="1000"/>

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 openwrt 源码学习如何添加修改），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。

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。

ZhuoTK

状态 ▾ 系统 ▾ 网络 ▾ 退出

刷新操作

动作 配置

备份/恢复

备份/恢复当前系统配置文件或重置OpenWrt(仅squashfs固件有效)。

下载备份: 生成备份

恢复到出厂设置: 执行复位

上传备份存档以恢复配置。

恢复配置: 选择文件 未选择任何文件 上传备份...

刷写新的固件

上传兼容的sysupgrade固件以刷新当前系统。

保留配置: ☒

固件文件: 选择文件 未选择任何文件 刷写固件...

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。

文件备份列表

动作 配置

系统升级时要保存的配置文件和目录的清单。目录/etc/cr

显示当前文件备份列表 打开列表...

```
## This file contains files and directories that should
## be preserved during an upgrade.
/overlay
# /etc/example.conf
# /etc/openvpn/
```

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，在开发板调试终端输入“firstboot”也能恢复出厂设置。

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。

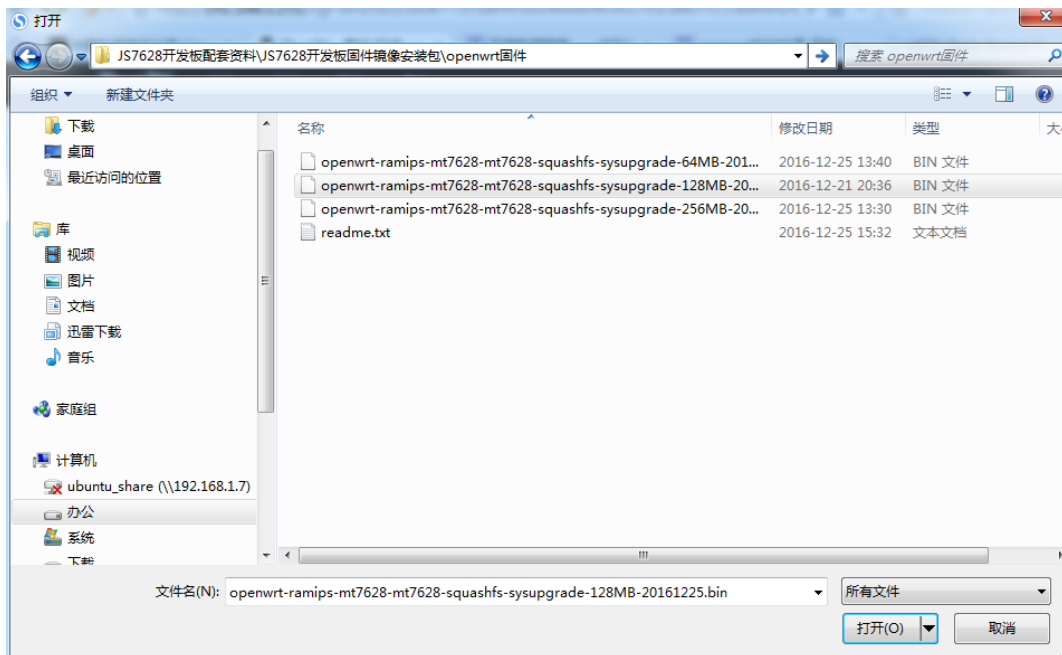
d) “刷写新固件栏”

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



点击“刷写固件”，如果固件正确，则会出现下图提示

刷新固件 - 验证

固件已上传，请注意核对文件大小和校验值！
刷新过程切勿断电！

- 校验值: 76a47d22f0409a8681d412120e5d8e40
- 大小: 4.75 MB(15.69 MB 可用)
- 注意: 配置文件将被删除。

取消 执行

点击上图右下角“执行”按钮，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：**JS7628 开发板恢复出厂设置后的 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。（**提示：**读者也可以在调试终端输入“reboot”执行重启）

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) “交换机”子菜单（切换 LAN/WAN 口功能）

在这个页面中，可以设置各个网口之间的“VLAN”，决定它们是否在一个网络里面，可以实现配置以太网口是“LAN”还是“WAN”功能，这里做一些介绍。该功能页面如下图所示

交换机

本设备可以划分为多个VLAN，并支持电脑间的直接通讯。VLAN也常用于分割不同网段。默认通常是一条上行端口连接ISP，其余端口为本地子网。

交换机"switch0" (rt305x-esw)

启用VLAN ☒

"switch0" (rt305x-esw)上的VLAN

VLAN ID	端口 0	端口 1	端口 2	端口 3	端口 4	端口 5	CPU
端口状态:	未连接	100baseT 全双工	未连接	未连接	未连接	未连接	1000baseT 全双工
1	不关联 ▼	不关联 ▼	不关联 ▼	不关联 ▼	关 ▼	关 ▼	关联 ▼
2	关 ▼	关 ▼	关 ▼	关 ▼	不关联 ▼	关 ▼	关联 ▼

从上图我们可以看到 JS7628 开发板默认有 2 个“VLAN ID”，一个是“VLAN ID 1”，连接到“LAN”网络，从下图可以看出

接口 - LAN

配置网络接口信息。

一般设置

基本设置 高级设置 物理设置 防火墙设置

桥接接口 ☒ 为指定接口创建桥接

开启STP ☐ 在此桥接上启用生成树协议

接口 ☐ 以太网适配器: "apcli0"
☐ 以太网交换机: "eth0"
☒ VLAN接口: "eth0.1" (lan)
☐ VLAN接口: "eth0.2" (wan, wan6)
☒ 以太网适配器: "ra0"
☒ 无线网络: Master "ZhuoTK_3344" (lan)

另外一个“VLAN ID 2”，连接到“WAN”的网络，从下图可以看出

接口 - WAN

配置网络接口信息。

一般设置

基本设置

高级设置

物理设置

防火墙设置

桥接接口 ☐ 为指定接口创建桥接

- 接口
- ☐ 以太网适配器: "apcli0"
 - ☐ 以太网交换机: "eth0"
 - ☐ VLAN接口: "eth0.1" (lan)
 - ☒ VLAN接口: "eth0.2" (wan, wan6)

切换这些接口和“VLAN”的“关联性”可以实现这些接口“WAN”-“LAN”口切换。JS7628 开发板上的 3 个网口分别对应上面的“端口 0”、“端口 1”、“端口 2”，它们均连接到“VLAN ID 1”，即 3 个网口默认都是 LAN 口，这些网口都可以切换成 WAN 口。

这里我们假设需要将“端口 0”（即 port0）切换为 WAN 口，我们只需要将“端口 0”和“VLAN ID 1”的关联性设置为“关”且和“VLAN ID 2”关联性设置为“不关联”，这样端口 0 就进入了“VLAN ID 2”中，最后点击右下角“保存&应用”按钮，“端口 0”即变为“WAN”口，配置如下图所示

"switch0" (rt305x-esw)上的VLAN

VLAN ID	端口 0	端口 1	端口 2	端口 3	端口 4	端口 5	CPU	
端口状态:								
1	关	不关联	不关联	不关联	关	关	关联	
2	不关联	关	关	关	不关联	关	关联	
<div>保存&应用 保存 复位</div>								

注意：开发板出厂时烧录的是“物联网网关”的固件，3 个网口均可用，如果读者烧录的是“物联网设备”固件，只有一个“端口 0”是可用的，如果配置了该端口为“WAN”口，可能会导致无法通过有线网口登录开发板配置页面。

4) “DHCP/DNS”子菜单

可以设置 DHCP 和 DNS 一些配置，比如 DNS 转发等。

5) “主机名”子菜单

可以设置主机名和 IP 的对应关系。

6) “静态路由”子菜单

可以设置开发板的路由表。

7) “防火墙”子菜单

可以设置接口的过滤规则、端口转发等。

8) “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。

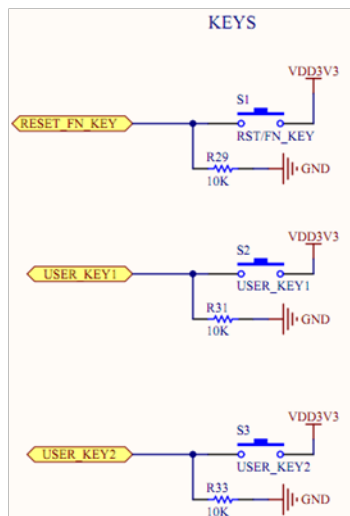
3.3. 基本硬件功能测试

这里介绍的硬件包括串口、LED、按键、GPIO，其他硬件如 micro sd 接口、音频接口、USB 接口等在下面的“玩转 openwrt 高级功能”一节中介绍。

3.3.1. 添加修改按键功能

3.3.1.1. 添加按键

JS7628 开发板上有 3 个按键，它们的原理图如下图所示



在 openwrt 的 MTK 系统中，添加按键是通过修改 openwrt 源码相应的 dts 文件实现的，文件在“openwrt/target/linux/ramips/dts/MT7628.dts”，打开这个文件我们可以看到下图

```

190         button_1 {
191             label = "button_1";
192             gpios = <&gpio1 6 0>;
193             linux,code = <0x101>;
194         };
195
196         button_2 {
197             label = "button_2";
198             gpios = <&gpio1 9 0>;
199             linux,code = <0x102>;
200         };
201
202         button_3 {
203             label = "button_3";
204             gpios = <&gpio1 12 0>;
205             linux,code = <0x103>;
206         };

```

这个 dts 文件中已经添加了对 JS7628 开发板 3 个按键的定义，我们以下图做一个介绍

```

button_1 {
    label = "button_1";
    gpios = <&gpio1 6 0>;
    linux,code = <0x101>;
};

```

其中

- a) “label” 表示按键名称
- b) “gpios” 中 “&gpio1” 表示处于 gpio 组 1，MT7628 有 3 组 gpio 分别是 gpio0、gpio1、gpio2 读者可以查看 MT7628 datasheet（在“JS7628 开发板配套资料\芯片元器件手册\MT7628\MT7628 Datasheet.pdf”），“6” 代表是当前组中的第 6 个 gpio，gpio1 的基础号是 32，所以本例中的 gpio 实际号是 $32+6=38$ （gpio38），“0” 代表低电平有效
- c) “linux,code” 表示的是该按键按下后给系统发送的标识号，这里是“0x101”，它是和 openwrt 源码
“openwrt/build_dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/linux-ramips_mt7628/linux-3.18.29/include/uapi/linux/input.h” 里面是相对应的，如下图所示

```

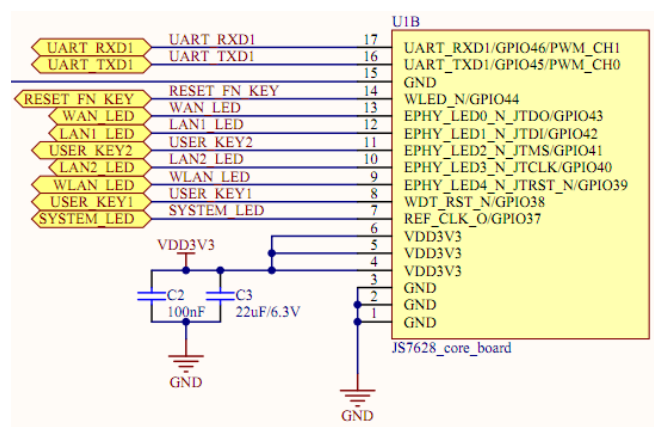
480 #define BTN_MISC          0x100
481 #define BTN_0             0x100
482 #define BTN_1             0x101
483 #define BTN_2             0x102
484 #define BTN_3             0x103
485 #define BTN_4             0x104
486 #define BTN_5             0x105
487 #define BTN_6             0x106
488 #define BTN_7             0x107
489 #define BTN_8             0x108
490 #define BTN_9             0x109

```

当用户按下这个按键后，系统就会到获取这个按键的事件。
按照上面的分析可以知道，其他按键和 GPIO 的对应关系分别是

“button_2” 处于 gpio1 组， $32+9$ ，对应 gpio41

“button_3” 处于 gpio1 组， $32+12$ ，对应 gpio44
对应的 GPIO 原理图是



3.3.1.2. 添加按键功能

做完上一节，我们知道开发板上的 3 个按键在按下后会给系统发送相应的按键事件，这一节我们来捕捉这些按键事件，并实现一些功能。

在开发板的 3 个按键中，“button_3” 在 JS7628 开发板出厂时，已经默认做为“重启/恢复出厂”功能了（开发板上最左边的按键，PCB 上标识为“s1”，“RST/FN_KEY”）。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_3
    option action      released
    option handler      reboot
    option min          0
    option max          3

config button
    option button      BTN_3
    option action      released
    option handler      'jffs2reset -y && reboot'
    option min          5
    option max          30

```

其他 2 个按钮出厂时默认未配置功能，下面演示一下如何将“button_2”（PCB 上丝印为“USER_KEY2”）添加“开启/关闭 wifi 功能”，步骤如下

a) 添加功能源码

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_2'          #按钮名为“BTN_2”
    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
```

b) 测试按钮

添加完上面的功能代码后，我们可以按下“button_2”，看一下实际效果。

第一次按下，调试信息如下图显示

```
root@zhuotk:/# [15811.710000] br-lan: port 2(ra0) entered disabled state
```

同时可以看到“WLAN”LED 指示灯熄灭，并且名为“ZhuoTK_xxxx”的 wifi 也没有了，说明 wifi 已被成功关闭。

再次按下，调试信息如下图显示

```

[15906.580000] EEPROM:Read from [factory] offset 0x0,length 0x400.
[15906.770000] br-lan: port 2(ra0) entered forwarding state
[15906.770000] br-lan: port 2(ra0) entered forwarding state
[15908.770000] br-lan: port 2(ra0) entered forwarding state

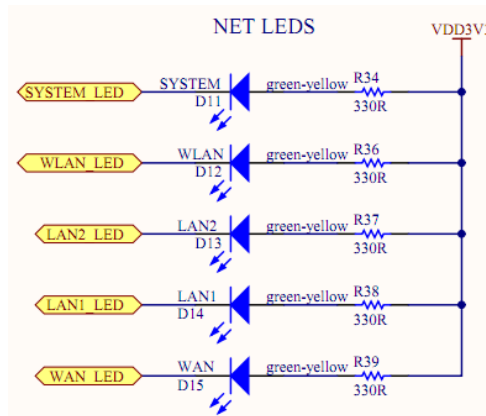
```

同时“WLAN”LED 指示灯亮起，又能搜索到名为“ZhuoTK_xxxx”的 wifi 了，说明 wifi 已被成功开启。

3.3.2. 添加修改 LED 指示功能

3.3.2.1. 添加 LED 指示

在开发板上有 5 个 LED 指示灯，该部分的原理图如下图所示



在 openwrt 中 MTK 系统添加 LED 是通过修改 openwrt 源码相应的 dts 文件实现的，文件在“openwrt/target/linux/ramips/dts/MT7628.dts”，打开这个文件我们可以看到下图

```

148     gpio-leds {
149         compatible = "gpio-leds";
150
151         system {
152             label = "mediatek:green:system";
153             gpios = <&gpio1 5 1>;
154             default-state = "off";
155         };
156
157         wifi {
158             label = "mediatek:green:wifi";
159             gpios = <&gpio1 7 1>;
160             default-state = "off";
161         };
162
163         wan {
164             label = "mediatek:green:wan";
165             gpios = <&gpio1 11 1>;
166             default-state = "off";
167         };
168
169         lan1 {
170             label = "mediatek:green:lan_1";
171             gpios = <&gpio1 10 1>;
172             default-state = "off";
173         };
174
175         lan2 {
176             label = "mediatek:green:lan_2";
177             gpios = <&gpio1 8 1>;
178             default-state = "off";
179         };

```

这个 dts 文件中已经添加了对 JS7628 开发板 5 个按键的定义，我们以下图做一个介绍

```

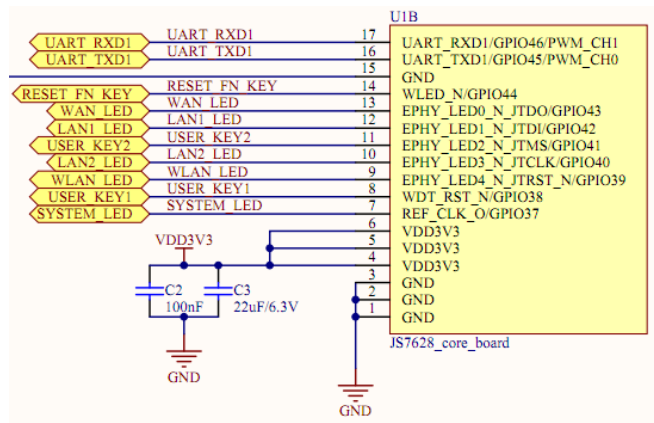
system {
    label = "mediatek:green:system";
    gpios = <&gpio1 5 1>;
    default-state = "off";
};

```

其中

- “label”表示 LED 名称，最终可以在 openwrt 配置页面->“LED 配置”->“LED 名称”中看到。
- “gpios”表示该 LED 用到的 GPIO，其中“&gpio1”表示处于 gpio 组 1，MT7628 有 3 组 gpio 分别是 gpio0、gpio1、gpio2 读者可以查看 MT7628 datasheet (在“JS7628 开发板配套资料\芯片元器件手册\MT7628\MT7628 Datasheet.pdf”)，“5”代表是当前组中的第 5 个 gpio，gpio1 的基础号是 32，所以本例中的 gpio 实际号是 32+5=37 (gpio37)，“1”代表高电平有效。
- “default-state”是默认状态

对应的 GPIO 原理图是



3.3.2.2. 修改 LED 指示

开发板的 5 个按键已经配置了功能。这里我们修改“SYSTEM”LED 指示时间做为例子，演示如何修改 LED 功能。比如现在我们需要将“SYSTEM”LED 亮灯时间改为 3 秒，灭灯时间为 1 秒（板子出厂时默认亮灭都是 1 秒），修改方式有如下 2 种

- a) 方法 1。编辑开发板的“/etc/config/system”文件，找到“system”LED 的配置行，修改为下图所示

```
config led
option default '0'
option name 'system'
option sysfs 'mediatek:green:system'
option trigger 'timer'
option delayoff '1000'
option delayon '3000'
```

修改完成后，保存退出，重启开发板生效。

- b) 方法 2。打开 openwrt 配置网页，到“系统”->“LED 配置”，将“system”一栏修改为下图

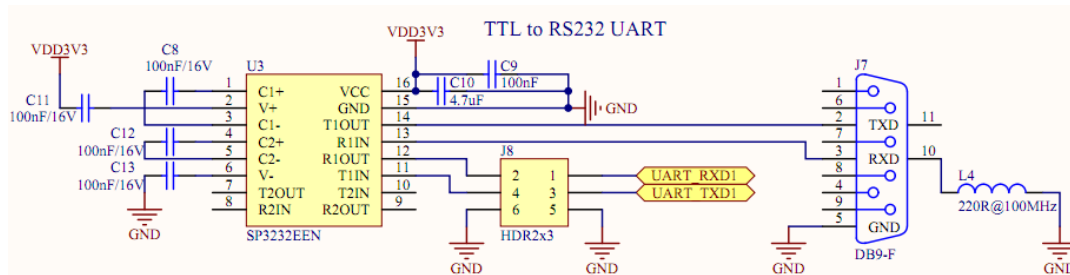
名字	system
LED名称	mediatek:green:system
默认状态	<input type="checkbox"/>
触发	timer
通电时间	3000
关闭时间	1000

点击当前页面右下角的“保存&应用”按钮，修改立即生效。具体配置效果请读者查看开发板。

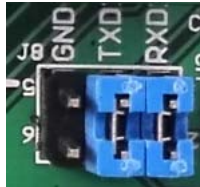
3.3.3. TTL/RS232 串口测试说明

3.3.3.1. 串口介绍

开发板上板载有一个 TT/RS232 串口，该串口的原理图如下



这个串口可以通过 J8 排针插上跳线帽，连接 1<==>2, 3<==>4，实物图如下图所示



实现 RS232 通讯功能。去掉跳线帽，则可以连接 1 号脚(RX), 3 号脚(TX), 5 或 6 号脚(GND) 实现 3.3V 串口通讯功能。

3.3.3.2. 串口测试

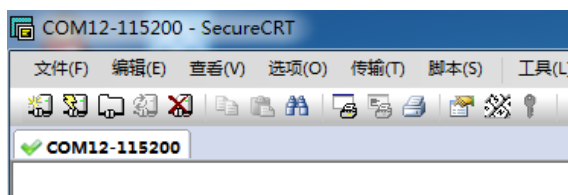
下面介绍一下这个 TTL/RS232 串口的测试。

1) 连接串口线到 TTL/RS232 串口

我们需要一根单独 RS232 串口线或 TTL 串口线（开发板配件中未有该配件，请读者自备）将其连接到开发板上的这个 TTL/RS232 串口接口（用 RS232 功能则接上跳线帽，用 TTL 串口则去掉跳线帽进行连接）。

2) 打开外部串口

新建另外一个 SecureCRT 窗口，并连接该串口线的串口号，设置波特率 115200，数据位 8，停止位 1，奇偶校验无，无流控，打开这个串口，如下图所示（作者电脑上这个串口号是 12）

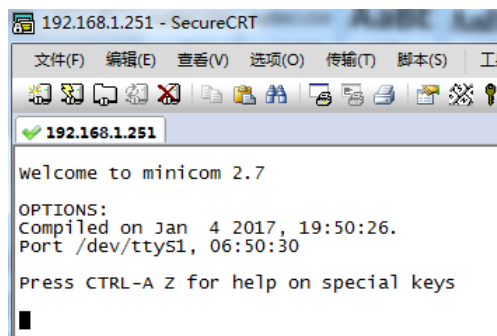


3) 开发板安装“minicom”安装包（安装方法参考本教程中“安装 IPK 包”一节）。

4) 打开开发板 TTL/RS232 串口

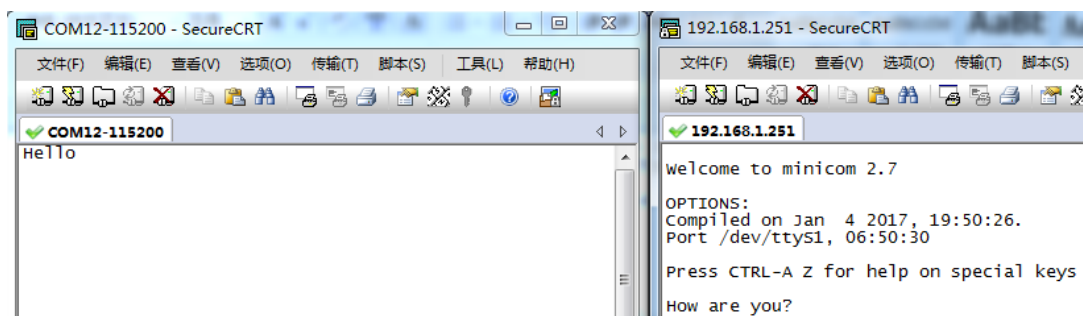
本来我们可以直接通过调试串口终端执行 minicom 命令，但是 minicom 在串口终端下，由于界面显示问题，会导致调试界面“乱码”，影响我们的控制，所以我们这里采用“ssh 登录命令行终端”的方法（《JS7628 开发板使用手册》中有介绍这个方法的具体步骤），来控制 minicom。

在 ssh 终端下我们输入“minicom -D /dev/ttyS1 -b 115200”（“/dev/ttyS1”是该 TTL/RS232 串口的设备号），然后我们将进入 minicom 控制台，如下图所示



5) 测试串口通讯

现在我们将 COM12 的控制终端和 ssh 控制终端同时显示。在两边测试输入一些东西。



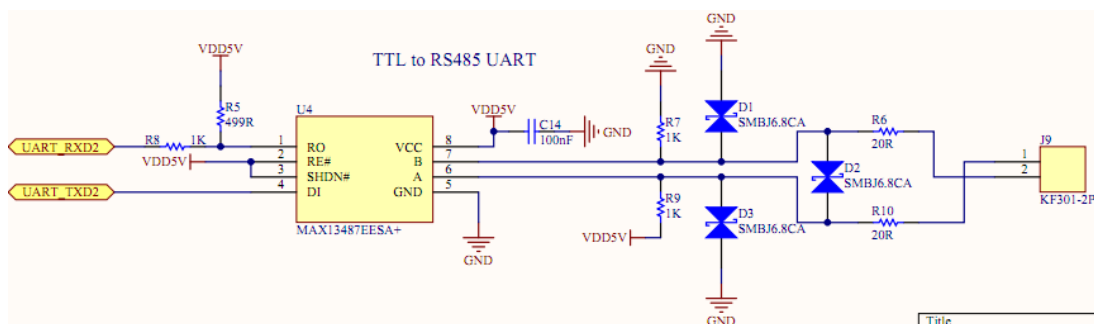
我们输入在右侧窗口“hello”，结果在左侧窗口显示了。在左侧窗口输入“How are you?”结果在右侧窗口显示了。这表示开发板 TTL/RS232 串口和外部串口通信成功！

提示：在《JS7628 开发板 openwrt 入门教程》中将介绍“wifi 串口透传”高级应用。

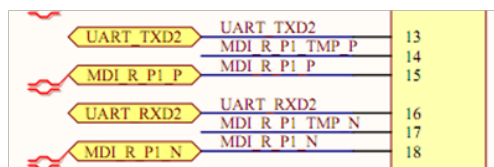
3.3.4. TTL/RS485 串口测试说明

3.3.4.1. 串口介绍

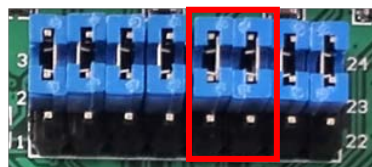
开发板上板载有一个 TT/RS485 串口，该串口的原理图如下



这个串口可以通过 J3 排针接口（接口如下图实物图红框所示）



原理图



实物图

开发板出厂时默认用跳线帽连接 14<==>15, 17<==>18 网口连接模式。将跳线帽连接 13<==>14, 16<==>17 实现 RS485 串口硬件连接，如果外部串口线直接连接 14、17、GND 脚

则可以实现 TTL 串口连接。

由于本实验用到了 MT7628 的串口 2，串口设备号是/dev/ttyS2，要使用该串口，必须让芯片处于“物联网设备”模式（烧录

“openwrt-ramips-mt7628-mt7628-squashfs-sysupgrade-xxxMB-IOT-device-xxxx.bin”固件才能使用这个串口 2。

3.3.4.2. 串口测试

下面介绍一下这个 TTL/RS485 串口的测试。

1) 连接串口线到 TTL/RS485 串口

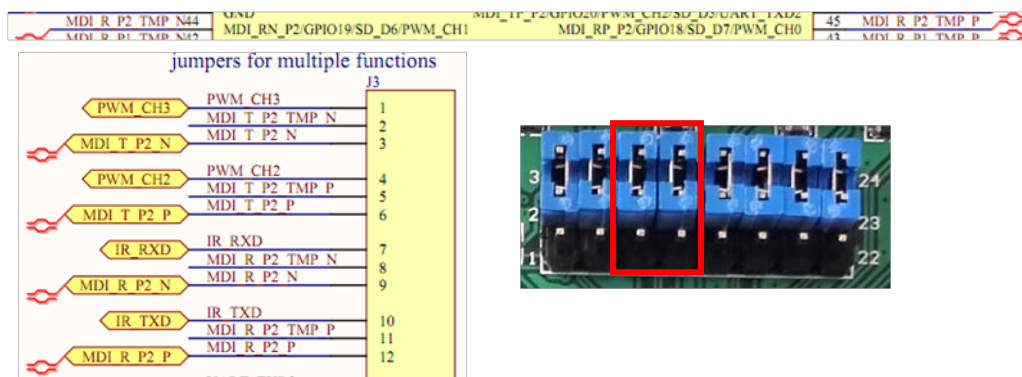
我们需要一根单独 RS485 串口线或 TTL 串口线（开发板配件中未有该配件，请读者自备）将其连接到开发板上的这个 TTL/RS485 串口接口，并注意 RS485 的正负端连接，A（D+）接 A（D+），B（D-）接 B（D-）。

2) 接下来的步骤和上一节“TTL/RS232 串口测试说明”完全一致，这里不再赘述。只是需要注意的是，打开开发板这个串口 2 的命令是“minicom -D /dev/ttyS2 -b 115200”（“/dev/ttyS2”是该 TTL/RS485 串口的设备号）

3.3.5. GPIO 简单控制

JS7628 核心板上有开发板上有 30 个与其他功能复用的 GPIO 引脚，JS7628 开发板上有最多 11 个可以单独引出的 GPIO 引脚。下面就以开发板上的 GPIO18、GPIO19 为例，介绍一下如何控制 JS7628 开发板上的 GPIO。

这两个管脚对应的原理图是



由原理图可知，这两个引脚对应关系是

GPIO18——J3 的 11 号脚

GPIO19——J3 的 8 号脚

它们在开发板上设计用于控制红外发射和接收功能或者是网口功能，由于本实验用到的 GPIO 引脚与网口复用，必须让芯片处于“物联网设备”模式（烧录

“openwrt-ramips-mt7628-mt7628-squashfs-sysupgrade-xxxMB-IOT-device-xxxx.bin”固件）才能使用这个 GPIO 引脚。开发板默认配置这两个引脚是 GPIO 功能。

简单测试 GPIO 功能的具体步骤如下

- 1) 拔掉对应的跳线帽
- 2) 执行以下命令


```
cd /sys/class/gpio/           //进入 GPIO 控制目录
echo 18 > export              //生成 GPIO 的控制目录
cd gpio18/                    //进入 GPIO 的控制目录
```

3) 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@zhuotk:/sys/devices/10000000.palmbus/10000600.gpio/gpio/gpio18# cat direction
in
```

上图“in”表示为输入状态。

4) 查看当前 GPIO 的电平状态，执行命令

```
cat value
```

结果如下图所示

```
root@zhuotk:/sys/devices/10000000.palmbus/10000600.gpio/gpio/gpio18# cat value
0
```

“0”表示低电平，如果是“1”则表示高电平。

将 GPIO18 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@zhuotk:/sys/devices/10000000.palmbus/10000600.gpio/gpio/gpio18# cat value
1
```

5) 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
```

GPIO19 的测试步骤同上。有兴趣的读者，也可以把这两个 GPIO 引脚，用杜邦线跳接到 J3 的 LED 指示灯接口上实现 GPIO 控制 LED 亮灭，这个留给读者自己实现。

提示：开发板资料里面提供的

“openwrt-ramips-mt7628-mt7628-squashfs-sysupgrade-xxxMB-IOT-device-xxxx.bin” 固件默认配置其他引脚不是 GPIO 功能，读者如果需要用到更多的 GPIO 功能的引脚，需要自行修改 dts 文件，编译源码，这部分的说明将在“深入开发篇”==》“配置 JS7628 开发板硬件默认功能”一节中给出。

3.4. 玩转 OpenWrt 高级功能

3.4.1. 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。

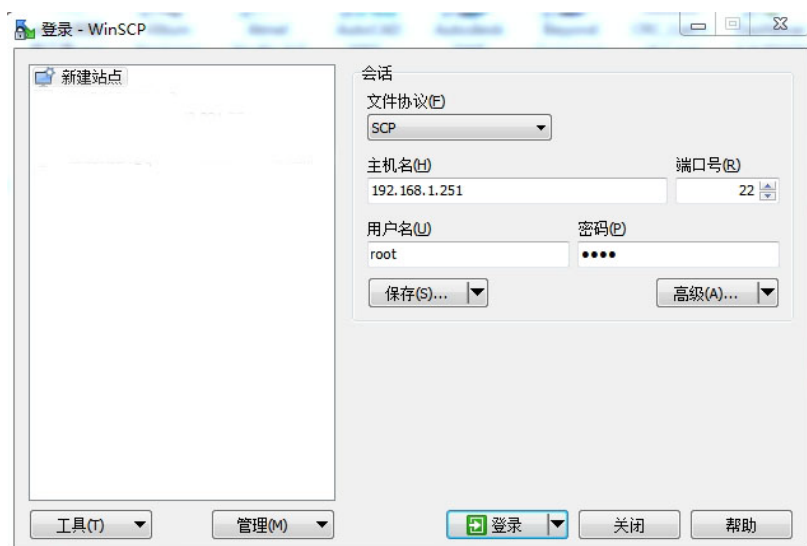
3.4.1.1. 用 scp 协议传输文件

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。

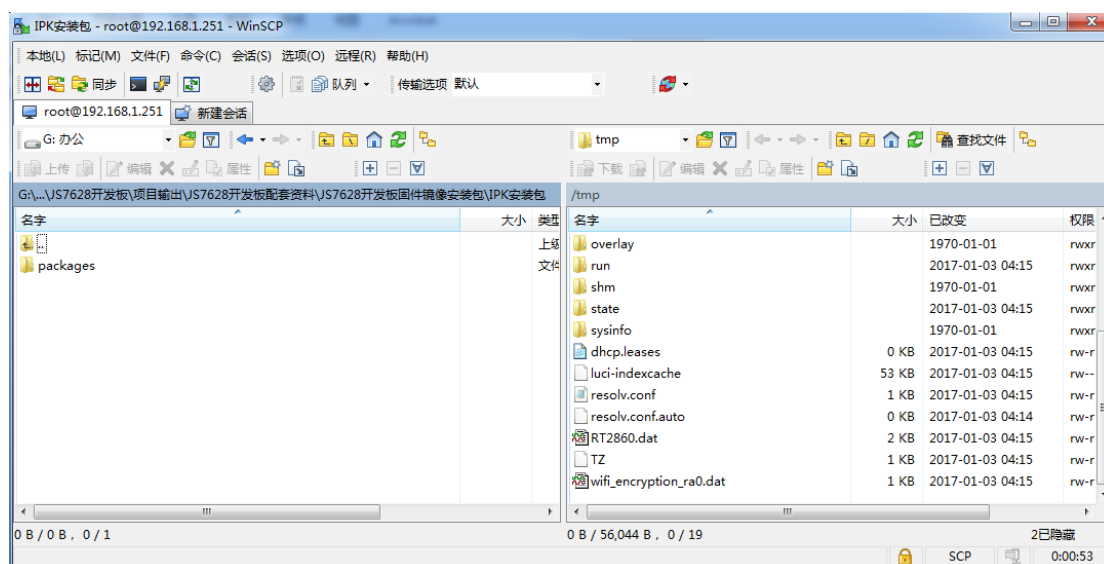
在 windows 下可以用“winscp”软件进行 scp 协议文件的传输，该软件在“JS7628 开发板配套资料\开发工具软件\winscp_V5.7.1.5235_setup.exe”请读者自行安装，安装完成后，



点击图标 WinSCP 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@192.168.0.1251:/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

3.4.1.2. 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”：资料同步写入内存和硬盘

“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

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 #开发板默认已经安装了 NFS 支持的安装包，可以直接挂载 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

```
root@zhuotk:/# ls /mnt/nfs/
core      涓涓涓涓涓涓  涓涓涓涓涓涓  涓涓涓涓涓涓
embedded  涓涓涓涓涓涓  涓涓涓涓涓涓  涓涓涓涓涓涓
examples.desktop 涓涓涓涓涓涓  涓涓涓涓涓涓  涓涓涓涓涓涓
```

ubuntu 上执行

```
ls /home/robinson
```

结果如下图所示

```
robinson@robinson-virtual-machine:~$ ls
core embedded examples.desktop 公共的 模板 视频 图片 文档 下载 音乐 桌面
```

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。

3.4.1.3. 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

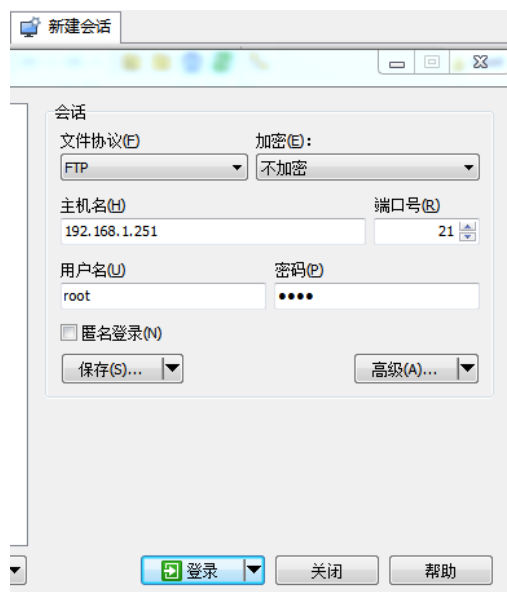
安装完成“vsftpd”后，系统自动启动 FTP 服务并设置了开机启动。

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

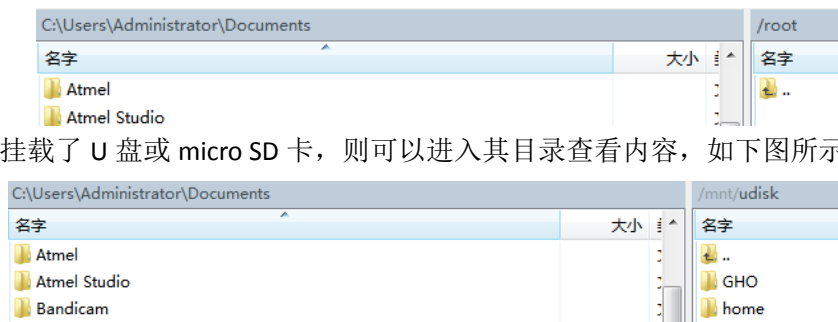
```
/etc/init.d/vsftpd restart #重启 FTP 服务，使修改后的配置生效
```

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 盘或 micro SD 卡，则可以进入其目录查看内容，如下图所示。

3.4.2. 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“https://downloads.openwrt.org/chaos_calmer/15.05.1/ramips/mt7628/packages/”或者到“JS7628 开发板配套资料\JS7628 开发板固件镜像安装包\IPK 安装包\packages”目录下，搜索“usbutils”，找到相应的安装包“usbutils_007-1_ramips_24kec.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_007-1_ramips_24kec.ipk
```

结果如下所示

```
root@zhuoTK:/# opkg install /tmp/usbutils_007-1_ramips_24kec.ipk
Installing usbutils (007-1) to root...
Collected errors:
* satisfy_dependencies_for: Cannot satisfy the following dependencies for usbutils:
  * libusb-1.0 * zlib *
* opkg_install_cmd: Cannot install package usbutils.
```

以上结果提示我们还需要先安装“usbutils”依赖的“libusb-1.0”和“zlib”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/zlib_1.2.8-1_ramips_24kec.ipk
```

```
opkg install /tmp/libusb-1.0_1.0.19-1_ramips_24kec.ipk
```

```
opkg install /tmp/usbutils_007-1_ramips_24kec.ipk
```

结果如下图所示

```
root@zhuoTK:/# opkg install /tmp/zlib_1.2.8-1_ramips_24kec.ipk
Installing zlib (1.2.8-1) to root...
Configuring zlib.
root@zhuoTK:/# opkg install /tmp/libusb-1.0_1.0.19-1_ramips_24kec.ipk
Installing libusb-1.0 (1.0.19-1) to root...
Configuring libusb-1.0.
root@zhuoTK:/# opkg install /tmp/usbutils_007-1_ramips_24kec.ipk
Installing usbutils (007-1) 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 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
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。

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。
```

3.4.3. 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 JS7628 开发板进行路由模式的配置。

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。

一般设置

基本设置

高级设置

物理设置

防火墙设置

状态

br-lan

运行时间: 11h 4m 17s
MAC-地址: AA:02:35:8E:89:FC
接收: 2.39 MB (28441 数据包)
发送: 3.38 MB (27337 数据包)
IPv4: 192.168.1.251/24
IPv6: FD4D:BE54:883B:0:0:0:0:1/64

协议

静态地址

IPv4地址

192.168.1.251

IPv4子网掩码

255.255.255.0

IPv4网关

IPv4广播

使用自定义的DNS服务器

DHCP服务器

基本设置

高级设置

IPv6 Settings

关闭DHCP

☒ 禁用本接口的DHCP。

开始

100

网络地址的起始分配基址。

客户数

150

最大地址分配数量。

租用时间

12h

地址租期，最小2分钟(2m)。

保存&应用

保存

复位

上图中，已经配置好了路由模式。

3.4.4. 拨号上网（路由器模式）

开发板实现拨号上网功能步骤如下

- 1) 在 JS7628 开发板上的 3 个网口默认都是“LAN 口”功能，但拨号上网一般需要用“WAN 口”的功能，所以我们需要将其中一个切换为“WAN 口”，这里我们用“端口 0”做为例子，按照“网络”菜单项一节中介绍的方法将其切换 WAN 口。

- 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。

一般设置

基本设置 高级设置 物理设置 防火墙设置

状态

pppoe-wan

接收: 0.00 B (0 数据包)
发送: 0.00 B (0 数据包)

协议

PPPoE

PAP/CHAP用户名

PAP/CHAP密码

接入集中器

自动

留空则自动探测

服务名

自动

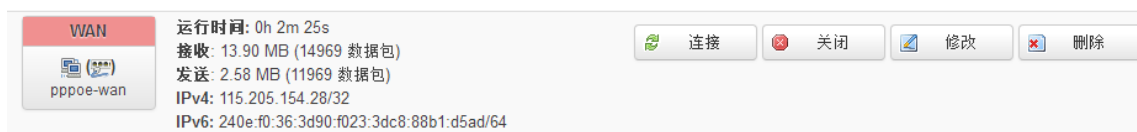
留空则自动探测

保存&应用

保存

复位

- 5) 将端口 0(port0)接上运营商的网络，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。



3.4.5. 多拨功能

3.4.6. 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 JS7628 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。

接口 - LAN

配置网络接口信息。

一般设置

基本设置 高级设置 物理设置 防火墙设置

状态	 br-lan	运行时间: 0h 5m 51s MAC-地址: AA-02:35:8E:89:FC 接收: 68.63 KB (569 数据包) 发送: 176.15 KB (624 数据包) IPv4: 192.168.1.251/24 IPv6: FD4D:BE54:883B:0:0:0:0:1/60
协议	静态地址	
IPv4地址	192.168.1.251	
IPv4子网掩码	255.255.255.0	
IPv4网关	192.168.1.1	
IPv4广播		
使用自定义的DNS服务器	192.168.1.1	
IPv6 assignment length	60	
Assign a part of given length of every public IPv6-prefix to this interface		
IPv6 assignment hint		
Assign prefix parts using this hexadecimal subprefix ID for this interface.		

DHCP服务器

基本设置 IPv6 Settings

关闭DHCP ☒ 禁用本接口的DHCP。

保存&应用 保存 复位

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。

无线网络开关 ☒ 禁用

信道 8 (2.447 GHz)

无线电功率 30 dBm (1000 mW)

☒ dBm

接口配置

基本设置 无线安全 MAC过滤

ESSID JoySince

模式 接入点AP (WDS)

网络 ☒ lan: ☒ wan: ☐ wan6: ☐ 创建:

注意，开发板出厂时以上参数已设置好，一般无需更改。

3.4.7. sta 模式设置

目前 JS7628 开发板采用的是 MTK 的 wifi 驱动，可以用修改配置文件的方式实现 STA 模式步骤如下

- 1) 修改/etc/config/network

wan 接口的“ifname”改为“apcli0”下图所示

```
config interface 'wan'
    option ifname 'apcli0'
    option force_link '1'
    option macaddr '00:ca:02:01:00:02'
    option proto 'dhcp'
```

- 2) 执行 iwpriv apcli0 get_site_survey 查看 wifi 列表，并记录下 wifi 名，加密方式

```
root@zhuotk:~# iwpriv apcli0 get_site_survey
apcli0 get_site_survey:
Ch SSID BSSID Security Signal(%)w-Mode ExtCH NT
1 TP-LINK_603 fc:d7:33:00:00:00 WPA1PSK/WPA2PSK/AES 37 11b/g/n ABOVE Ir
2 FLY f4:6a:92:2d:00:00 WPA1PSK/WPA2PSK/AES 57 11b/g/n ABOVE Ir
2 00:15:6d:00:00:00 NONE 24 11b/g/n ABOVE Ir
4 hcedu d4:83:04:00:00:00 NONE 57 11b/g/n ABOVE Ir
6 FAST_94D6 d4:83:04:00:00:00 WPA1PSK/WPA2PSK/AES 10 11b/g/n BELOW Ir
11 0x3132e2809431e2809431383033 bc:46:99:00:00:00 WPA1PSK/WPA2PSK/AES 73 11b/g/n BELOW Ir
11 16wifi 02:01:7a:00:00:00 NONE 13 11b/g/n NONE Ir
11 qjc38000 30:fc:68:00:00:00 WPA1PSK/WPA2PSK/AES 7 11b/g/n BELOW Ir
11 robinson 0c:84:dc:00:00:00 WPA2PSK/AES 100 11b/g/n NONE Ir
11 TP-LINK_2C6F bc:46:99:00:00:00 WPA1PSK/WPA2PSK/AES 0 11b/g/n BELOW Ir
```

如上图所示，作者建立了一个 SSID 是“robinson”的热点，这个热点加密方式是“WPA2PSK/AES”。

- 3) 修改配置文件/etc/config/wireless，根据上一步的 SSID，加密方式填写如下

```

config wifi-device ra0
    option type ralink
    option mode 9
    option channel 8
    option txpower 17
    option ht 20
    option country US

config wifi-iface
    option device ra0
    option network 'lan'
    option mode 'ap'
    option ssid ZhuoTK_0001
    option encryption 'none'
    option ApCliEnable '1'
    option ApCliSsid 'robinson'
    option ApCliAuthMode 'WPA2PSK'
    option ApCliEncryptType 'AES'
    option ApCliPassword '123456abcd'

```

保存退出。

4) 然后执行命令

`/etc/init.d/network restart`

重启网络或者直接重启板子，如果看到如下提示

```
[ 3724.560000] !!! APCLI LINK UP - IF(apcli0) AuthMode(7)=WPA2PSK, wpaStatus(6)=AES!
```

则说明开发板成功连接上了热点，可以上网了

```

root@zhuotk:/# ping www.baidu.com
PING www.baidu.com (117.185.16.31): 56 data bytes
64 bytes from 117.185.16.31: seq=0 ttl=50 time=30.649 ms
64 bytes from 117.185.16.31: seq=1 ttl=50 time=248.807 ms
64 bytes from 117.185.16.31: seq=2 ttl=50 time=69.000 ms

```

注意：上级路由器的 IP 网段不能和开发板的 IP 网段（192.168.1.*）一样，否则可能会导致开发板网络不正常，无法上网。

3.4.8. 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）

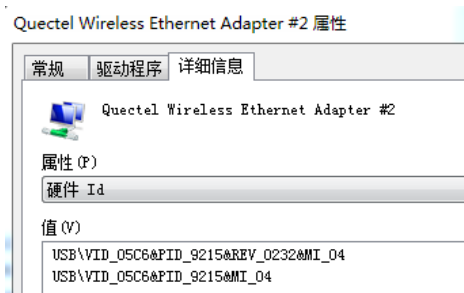


为例，介绍一下，如何增加开发板对 4G 网卡的支持。

具体步骤如下：

1) 获取网卡 PID、VID、接口号（本步骤可选）

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。



或者将网卡插到 Ubuntu 里面执行 “lsusb” 命令也可以查看设备号。

本例中网卡的硬件是 VID:05C6,PID:9215

2) 添加内核本 4G 网卡的驱动支持
编辑

“openwrt/build_dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/linux-ramips_mt7628/linux-3.18.29/drivers/net/usb/qmi_wwan.c” 文件，
在大概 674 行后面添加

```
“{QMI_FIXED_INTF(0x05c6, 0x9215, 4)}, /*Quectel EC20 V1*/”
```

如下图所示。

```
673 {QMI_FIXED_INTF(0x05c6, 0x9209, 3)},
674 {QMI_FIXED_INTF(0x05c6, 0x9215, 4)}, /*Quectel EC20*/
675 {QMI_FIXED_INTF(0x05c6, 0x9215, 4)}
```

增加内核驱动对本网卡 PID、VID、接口号的识别。

注释掉大概在 822 行的

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

防止设备号冲突

```
821 {QMI_Gobi_DEVICE(0x05c6, 0x9215)}, /* HP Gobi 2000 Modem device (VP412) */
822 // {QMI_Gobi_DEVICE(0x05c6, 0x9215)}, /* Acer Gobi 2000 Modem device (VP413) */
823 {QMI_Gobi_DEVICE(0x05c6, 0x9215)}, /* Asus Gobi 2000 Modem device (VP305) */
```

保存退出。

3) 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 converters

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

Network --->

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

保存退出。

4) 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

“wwan”

“kmod-mii”

“kmod-usb-net”

“kmod-usb-wdm”

“kmod-usb-net-qmi-wwan”

“uqmi”

提示：相关安装包在“JS7628 开发板配套资料\JS7628 开发板固件镜像安装包\IPK 安装包”里面，读者可以

根据之前的“安装包 IPK 包”一节里面介绍的方法，进行本地安装，这里不再赘述。

5) 测试 4G 网卡上网

安装了上述的安装包后，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@zhuotk:/# [ 3152.250000] usb 1-1.3: new high-speed USB device number 5 using ehci-platform
[ 3152.370000] qmi_wwan 1-1.3:1.4: cdc-wdm0: USB WDM device
[ 3152.380000] qmi_wwan 1-1.3:1.4 wwan0: register 'qmi_wwan' at usb-101c0000.ehci-1.3, WWAN/QMI device, 82:ba:ea:ed:6b:42
```

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

```
root@zhuotk:/# ls /dev
audio          mtd0           mtblock0       random
bus            mtd0ro         mtblock1       shm
cdc-wdm0       mtd1           mtblock2       snd
console        mtd1ro         mtblock3       tty
cpu_dma_latency mtd2           mtblock4       ttys0
```

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。

6) 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

创建新接口

新接口的名称

合法字符: A-Z, a-z, 0-9 和 _

新接口的协议

在多个接口上创建桥接 ☐

包括以下接口

- ☐ 以太网适配器: "@wan" (wan6)
- ☐ 以太网适配器: "eth0" (wan, wan6)
- ☐ 以太网适配器: "eth1" (lan)
- ☒ 以太网适配器: "wwan0"
- ☐ 无线网络: Master "JoySince" (lan)
- ☐ 自定义接口:

[返回至概况](#)

[提交](#)

然后点击该页面中的“提交”按钮。接着会出现下图

接口 - WWAN

配置网络接口信息。

一般设置

基本设置	高级设置	物理设置	防火墙设置
------	------	------	-------

状态

wwan0

MAC-地址: 00:00:00:00:00:00
接收: 0.00 B (0 数据包)
发送: 0.00 B (0 数据包)

协议

DHCP客户端

请求DHCP时发送的主机名

JoySince

保存&应用

保存

复位

选择“防火墙设置”一栏，并配置成“wan”模式，如下图所示。

接口 - WWAN

配置网络接口信息。

一般设置

基本设置	高级设置	物理设置	防火墙设置
------	------	------	-------

创建/分配 防火墙区域

lan: lan:

wan: wan: wan6:

未指定 // 创建:

此接口的防火墙区域。填写 创建 可新建防火墙区域。

保存&应用

保存

复位

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。

WWAN	运行时间: 0h 2m 38s MAC-地址: 0E:3B:B3:5A:28:B4 接收: 972.00 B (8 数据包) 发送: 1.82 KB (16 数据包) IPv4: 10.45.180.174/30	连接	关闭	修改	删除
------	--	----	----	----	----

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“JS7628 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“JS7628 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”

3.4.9. 开发板挂 VPN

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。

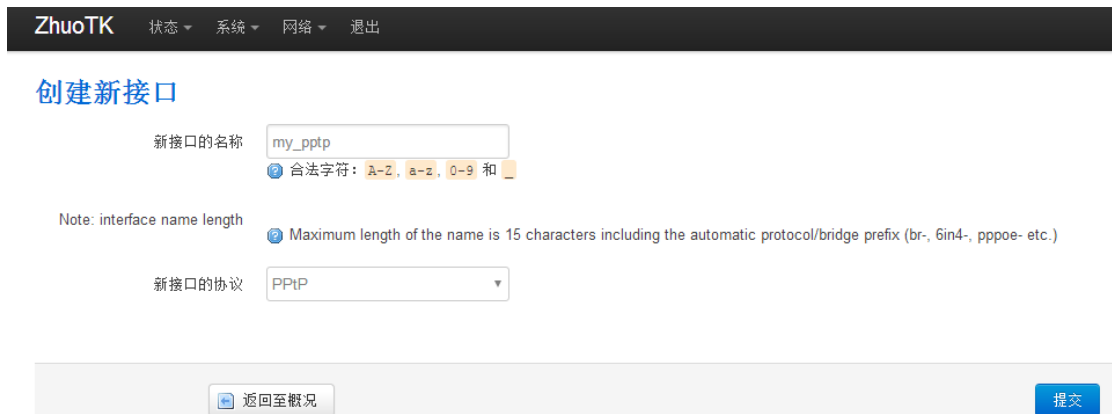
下面介绍如何让开发板实现 VPN 客户端来连接远程 VPN 服务器，最终实现开发板通过 VPN 间接访问因特网。步骤如下。

- 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
“kmod-iptunnel”
“kmod-gre”
“kmod-pptp”
“kmod-crypto-aead”
“kmod-crypto-pcompress”
“kmod-crypto-sha1”
“kmod-crypto-pcompress”
“kmod-crypto-manager”
“kmod-crypto-ecb”
“kmod-mppe”
“resolveip”
“ppp-mod-pptp”

(注意：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



ZhuoTK 状态 ▾ 系统 ▾ 网络 ▾ 退出

创建新接口

新接口的名称

合法字符: A-Z, a-z, 0-9 和 _

Note: interface name length Maximum length of the name is 15 characters including the automatic protocol/bridge prefix (br-, 6in4-, ppoe- etc.)

新接口的协议

[返回至概况](#) [提交](#)

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。

接口 - MY_PPTP

配置网络接口信息。

一般设置

基本设置

高级设置

防火墙设置

状态

pptp-my_pptp

接收: 0.00 B (0 数据包)
发送: 0.00 B (0 数据包)

协议 PPTP

VPN服务器 us1.freejsq.pw

PAP/CHAP用户名 username

PAP/CHAP密码

保存&应用 保存 复位

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。

（提示：上图中作者用的是免费 VPN，注册地址 <http://gjsq.me/412332>，登录后可以在 <https://www.getgreenjsq.com/xianlu.html> 查看 VPN 服务器地址。）

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

网络	状态	动作
MY_PPTP	运行时间: 0h 0m 3s 接收: 160.00 B (7 数据包) 发送: 192.00 B (8 数据包) IPv4: 10.0.0.85/32	连接 关闭 修改 删除

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.youtube.com，结果如下图所示

```
root@zhuotk:/# ping www.youtube.com
PING www.youtube.com (199.16.156.41): 56 data bytes
64 bytes from 199.16.156.41: seq=0 ttl=244 time=300.871 ms
64 bytes from 199.16.156.41: seq=1 ttl=244 time=294.923 ms
64 bytes from 199.16.156.41: seq=2 ttl=244 time=295.927 ms
64 bytes from 199.16.156.41: seq=3 ttl=244 time=291.033 ms
```

而未连接该 VPN 前，无法访问 www.youtube.com，如下图所示

```
root@zhuotk:/# ping www.youtube.com
PING www.youtube.com (199.16.156.41): 56 data bytes
```

提示：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。

接口 - MY_PPTP

配置网络接口信息。

一般设置

基本设置

高级设置

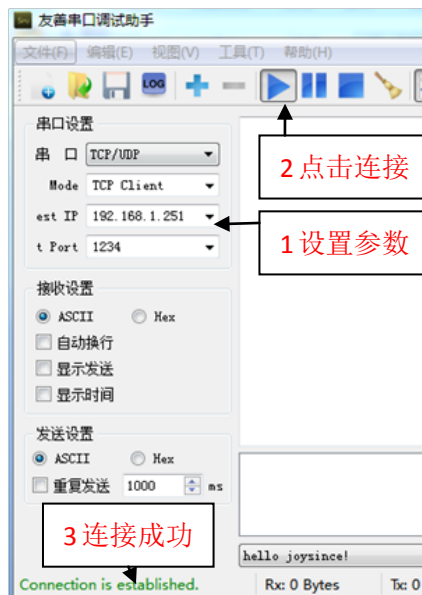
防火墙设置

开机自动运行 ☐Use builtin IPv6-management ☒使用默认网关 ☒ 留空则不配置默认路由通过网关访问

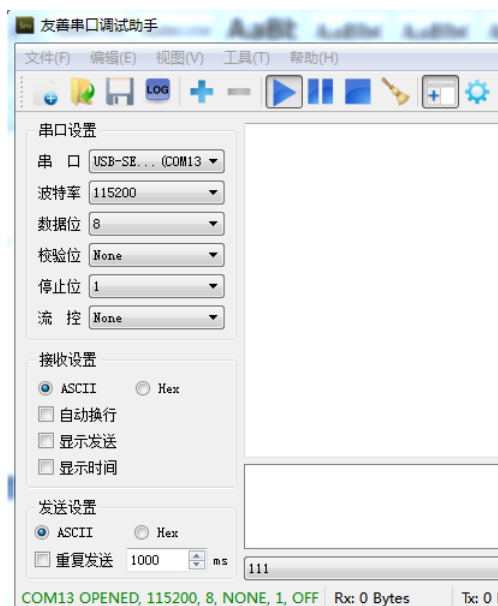
3.4.10. 网络串口透传

在本教程之前的“TTL/* 串口测试说明”章节中,已经介绍过 JS7628 的应用串口的简单使用方法了,这里以开发板上的“TT/RS232 串口”(设备号是/dev/ttyS1)介绍一下如何实现“网络串口透传功能”。步骤如下

- 1) 安装“ser2net”软件包(软件包安装方法请参考“安装 IPK 包”一节)
- 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。在其最后一行添加如下所示配置。
`1234:raw:0:/dev/ttyS1:115200 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”(“JS7628 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。

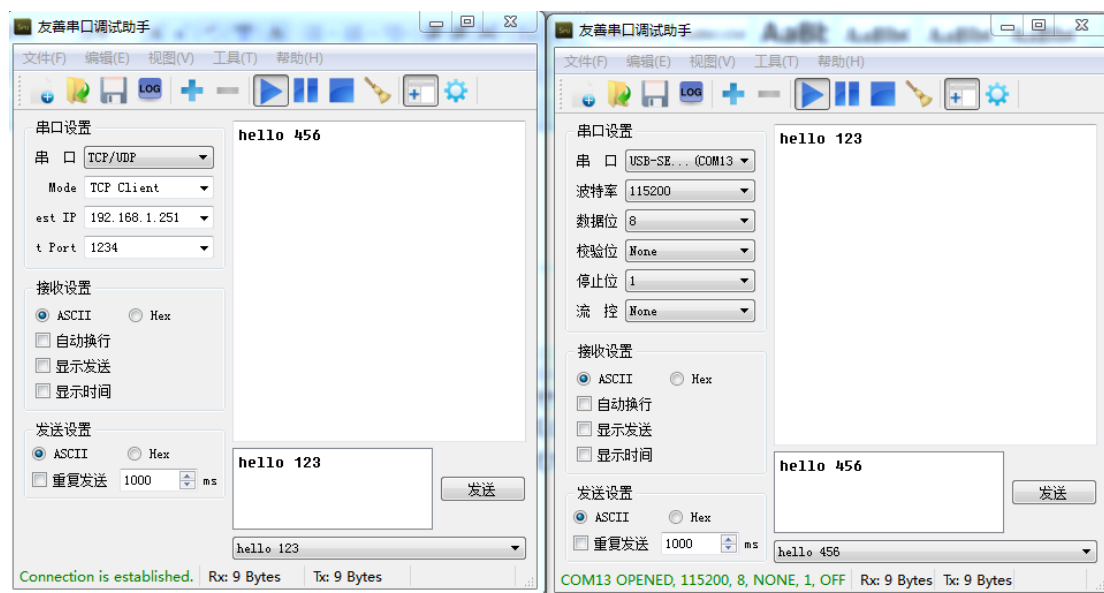


- 5) 用一条 USB 转 RS232 串口线连接电脑和开发板 RS232 接口(或者连接该串口开发板上对应的 RX、TX 引脚),这里我们也用“友善串口调试助手”来连接串口,如下图所示。



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“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.
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。

3.4.11. 挂载 U 盘、micro SD（TF）卡

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

JS7628 开发板默认已经安装了 U 盘驱动、SD 卡驱动、ext4、vfat 文件系统驱动, 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”。插入 micro SD 卡, 调试串口会出现类似下图提示

```
root@zhuotk:/# [ 310.930000] mmc0: new SD card at address 8001
[ 310.940000] mmcblk0: mmc0:8001 SU256 241 MiB
[ 310.940000] mmcblk0: p1
```

上图说明开发板已经正确识别 micro SD 卡, 并提示设备号是“mmcblk0p1”, 设备路径在“/dev/mmcblk0p1”。

执行命令

```
mkdir /mnt/udisk #建立一个用于挂载 U 盘的目录
```

```
mkdir /mnt/sd #建立一个用于挂载 micro SD 卡的目录
```

```
mount /dev/sda1 /mnt/udisk #将 U 盘设备“/dev/sda1”挂载到“/mnt/udisk”目录下
```

```
mount /dev/mmcblk0p1 /mnt/sd #将 micro SD 卡设备“/dev/mmcblk0p1”挂载到“/mnt/sd”目录下
```

```
ls /mnt/udisk #查看 U 盘内容
```

```
ls /mnt/sd #查看 micro SD 卡内容
```

2) 自动挂载

有时手动挂载 U 盘、micro SD 卡不方便, 这时我们需要自动挂载它们。这里我们利用 openwrt 现有的“mountd”这个软件实现自动挂载。参考“安装 IPK 安装包”一节, 安装好“mountd”软件后, 用户直接将 U 盘 (micro SD 卡) 插入开发板 USB 接口 (micro SD 卡座), 并出现正确识别的调试信息后, mountd 程序会把 U 盘 (micro SD 卡) 自动挂载到开发板的“/tmp/run/mountd/sda1/”目录下 (micro SD 卡则会挂载到“/tmp/run/mountd/mmcblk0p1/”目录下)。执行

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

```
ls /tmp/run/mountd/mmcblk0p1 #查看 micro SD 卡文件
```

3.4.12. 远程访问开发板

3.4.12.1. 路由器端口映射访问

下面我们使用“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 tp-link 路由器管理界面->“动态 DNS”，配置如下图所示



动态DNS设置

本页设置“Oray.net花生壳DDNS”的参数。

服务商链接: [花生壳动态域名解析服务申请](#) [花生壳动态域名解析服务帮助](#)

服务提供者: 花生壳 (www.oray.net) ▼

用户名: useradmin

密码: ●●●

启用DDNS: ☒

连接状态: 连接成功

服务类型: 标准服务

域名信息: 1: user.vicp.cc

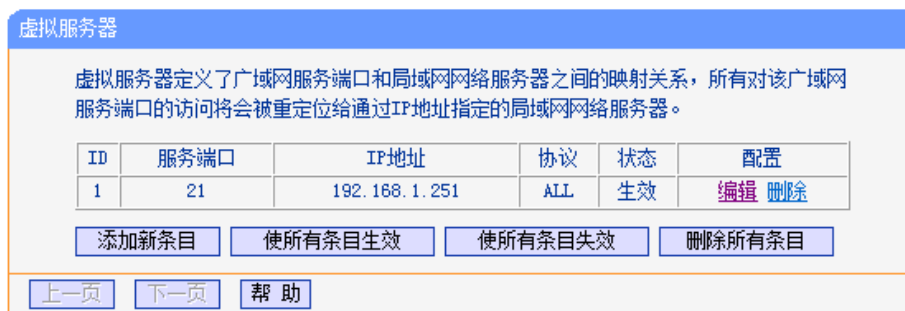
登录 退出

保存 帮助

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。



虚拟服务器

虚拟服务器定义了广域网服务端口和局域网网络服务器之间的映射关系，所有对该广域网服务端口的访问将会被重定位给通过IP地址指定的局域网网络服务器。

ID	服务端口	IP地址	协议	状态	配置
1	21	192.168.1.251	ALL	生效	编辑 删除

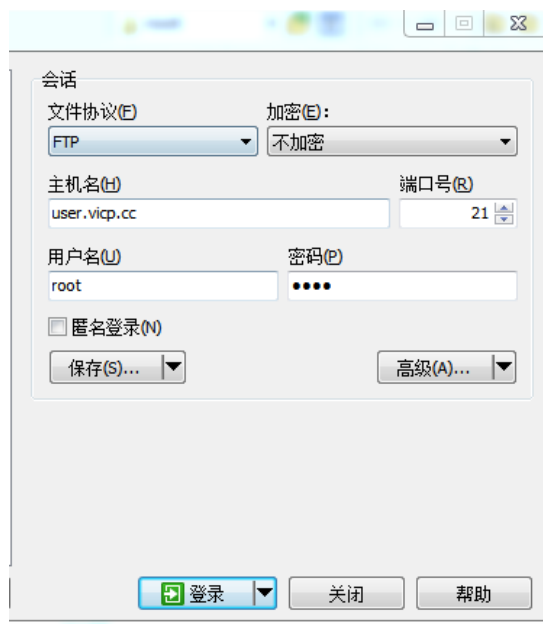
添加新条目 使所有条目生效 使所有条目失效 删除所有条目

上一页 下一页 帮助

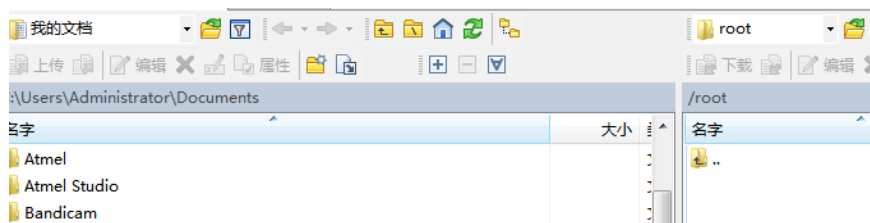
如上图所示，添加了 FTP 服务（21 端口）映射。

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，读者可以购买“花生棒”来解决这个问题。

3.4.12.2. 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 frp 客户端无法访问云服务器），设置如下图所示

添加安全组规则

网卡类型: 内网

规则方向: 入方向

授权策略: 允许

协议类型: 自定义 TCP

* 端口范围: 6000/8000

优先级: 1

授权类型: 地址段访问

* 授权对象: 0.0.0.0/0

描述:

长度为2-256个字符, 不能以http://或https://开头。

确定 取消

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

实例名称	监控	所在可用区	IP地址	状态(全部)	网络类型(全部)	配置	付费方式(全部)
实例名称							
实例名称							

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

```

aliyun-hd - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
aliyun-hd
welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-62-generic x86_64)
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
welcome to Alibaba Cloud Elastic Compute Service !
Last login: Sun Dec 31 12:41:46 2017 from 125.121.209.70
root@izuf69b96ia36ahgldsh3kz:~#

```

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

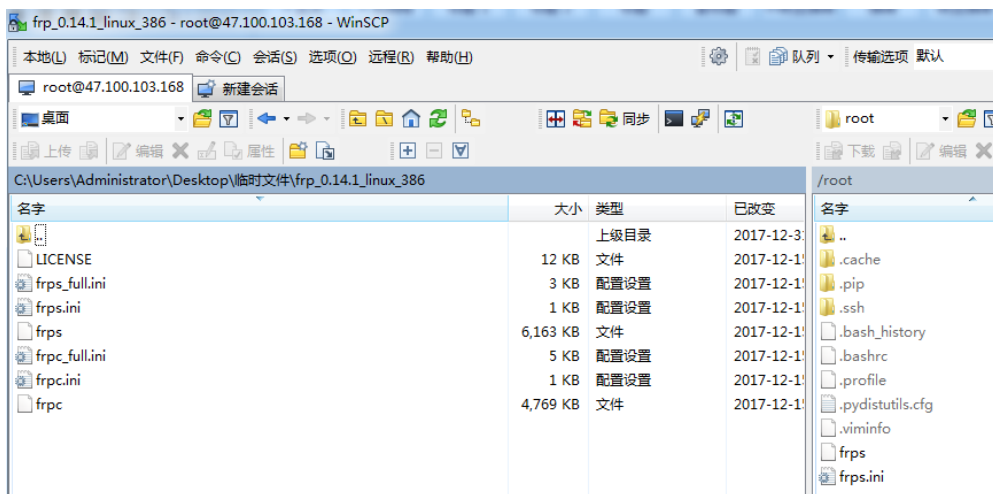
```

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“JS7628 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 security issues
```

4) 开发板下载并运行“frp”内网穿透客户端

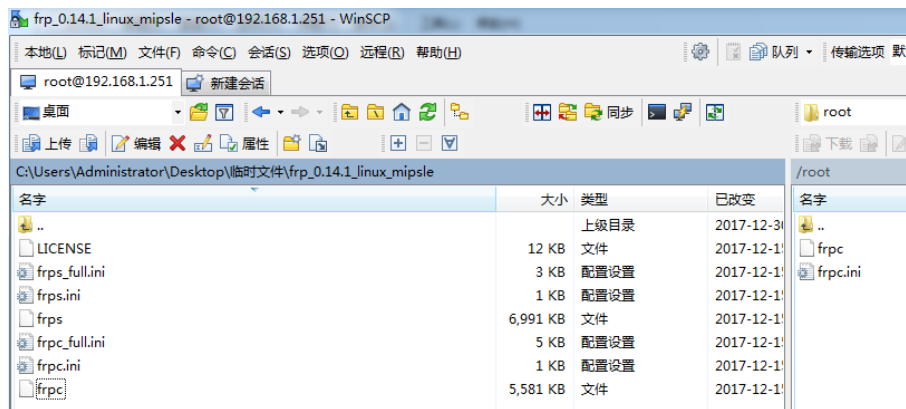
开发板上执行命令

`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，MT7628 和 MT7688 是 mipsel 构架，因此我们下载“frp_0.14.1_linux_mipsel.tar.gz”（提示:在“JS7628 开发板配套资料\JS7628 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], server udp port [0]
```

说明该客户端已经连接到云服务器端了。

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_port = 22
remote_port = 6000

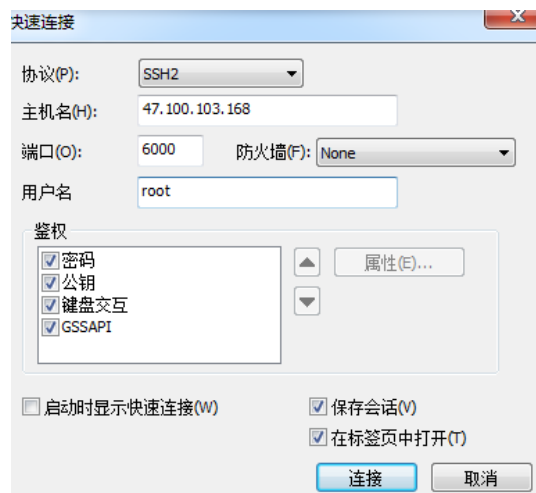
[common]
server_addr = xxx.xxx.xxx.xxx
server_port = 7000

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

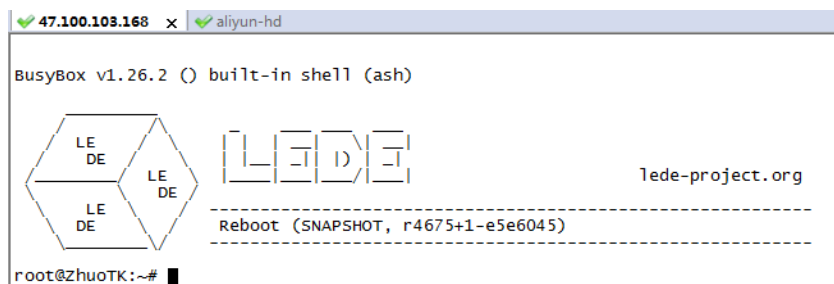
（上图“xxx.xxx.xxx.xxx”为云服务器 IP）开发板重新执行

`./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

修改解析 ×

记录类型：

主机记录：

解析线路：

记录值：

TTL值：

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
```

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

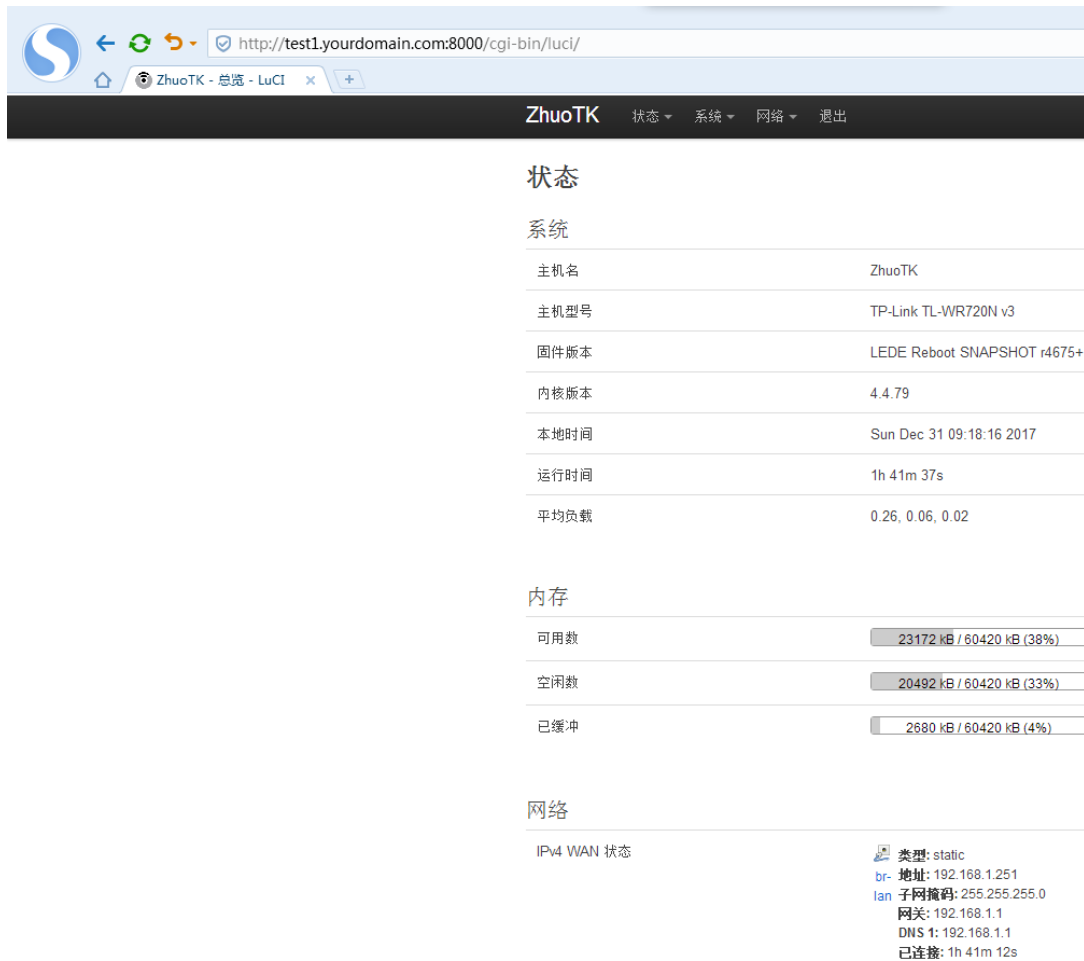
```
root@zhuotk:~# ./frpc -c ./frpc.ini
2017/12/31 09:11:29 [I] [control.go:277] [edd0978999e38ef8] login to server success, get run id [edd0978999e38ef8], server udp port [0]
2017/12/31 09:11:29 [I] [control.go:412] [edd0978999e38ef8] [ssh_1] start proxy success
2017/12/31 09:11:29 [I] [control.go:412] [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 [0.14.1] 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 [test1.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



The screenshot shows the ZhuoTK web interface. The top navigation bar includes links for '状态' (Status), '系统' (System), '网络' (Network), and '退出' (Logout). The main content area is divided into three sections:

- 状态 (Status):** A table displaying system information:

主机名	ZhuoTK
主机型号	TP-Link TL-WR720N v3
固件版本	LEDE Reboot SNAPSHOT r4675+1
内核版本	4.4.79
本地时间	Sun Dec 31 09:18:16 2017
运行时间	1h 41m 37s
平均负载	0.26, 0.06, 0.02
- 内存 (Memory):** A table showing memory usage:

可用数	23172 kB / 60420 kB (38%)
空闲数	20492 kB / 60420 kB (33%)
已缓冲	2680 kB / 60420 kB (4%)
- 网络 (Network):** A table showing network configuration:

IPv4 WAN 状态	类型: static 地址: 192.168.1.251 子网掩码: 255.255.255.0 网关: 192.168.1.1 DNS 1: 192.168.1.1 已连接: 1h 41m 12s
-------------	--

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

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.
/root/frpc -c /root/frpc.ini >/dev/null 2>&1 &
exit 0
~
```

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，“JS7628 开发板配套资料\JS7628 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。

3.4.13. 打造本地音乐播放器

JS7628 开发板上板载了 WM8960 声卡，下面我们测试一下如何让开发板驱动这个声卡实现本地音频播放，步骤如下

- 1) 将普通的带 3.5mm 接头耳机和开发板的 3.5mm 音频输出接口座连接
- 2) 传输音频文件到开发板

读者可以通过之前介绍的“挂载 U 盘”、“挂载 micro SD 卡”、“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。

- 3) 开发板执行命令本地播放音乐

由于开发板集成了 WM8960 音频驱动和 madplay 播放器，假如我们这里将一个 MP3 文件放在开发板的 “/tmp/music.mp3”，只需要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] ptr12s_config->mmap_index:0
```

美妙动听的音乐即可从开发板音频输出接口放出。

3.4.14. 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 mjpeg-streamer 相关的 IPK 驱动包

```
“libjpeg” //mjpeg-streamer 依赖包
```

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

- 2) 插入 USB 摄像头

JS7628 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 “mjpeg-streamer” 服务安装成功，则会出现 “/etc/config/mjpeg-streamer” 配置文件，编辑该文件，修改成以下配置

```
config mjpeg-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-streamer 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。

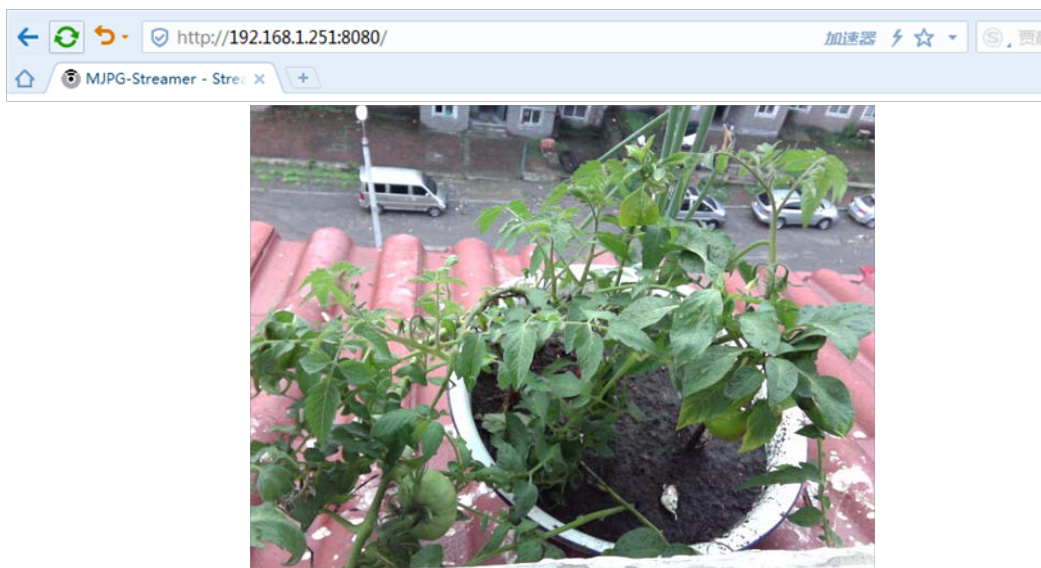
```
root@JoySince: /# /etc/rc.d/
K10mjpg-streamer K99umount S19firewall S50uhttpd
K50dropbear S00sysfixtime S20network S60dnsmasq
K85odhcpd S10boot S35odhcpd S90mjpg-streamer
```

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
```

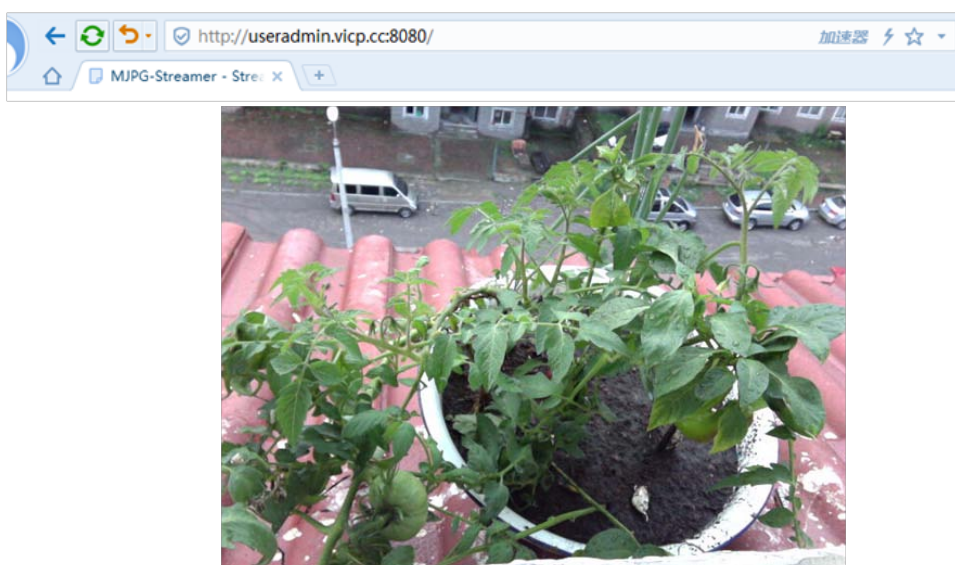
5) 查看摄像头视频

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。



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为 `mjpg-streamer` 服务专门开放一个端口（比如 8080 端口）即可。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。

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

JS7628 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，以上配件在购买 JS7628 开发板时均已配齐。

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 VMware 虚拟机软件

虚拟机安装方法见“JS7628 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

安装方法见“JS7628 开发板配套资料\JS7628 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 ubuntu 系统后，一定要安装“VM tools”以便开发共享文档，方法见“JS7628 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”。

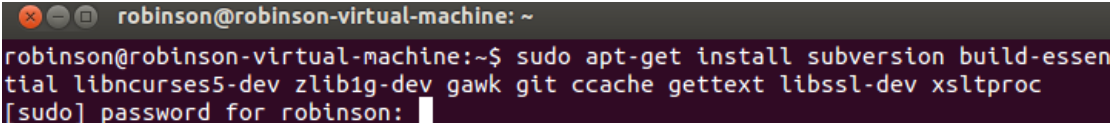
4.3. 搭建 OpenWrt 开发环境

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
```



```
robinson@robinson-virtual-machine: ~  
robinson@robinson-virtual-machine:~$ sudo apt-get install subversion build-essential libncurses5-dev zlib1g-dev gawk git ccache gettext libssl-dev xsltproc  
[sudo] password for robinson: 
```

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。**注意：如果未完全安装以上软件，可能会造成编译 openwrt 源码时出错，请务必安装完全。**

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。

下面介绍两种下载 openwrt 源码的方式。一种是从 openwrt 官方下载 openwrt 源码，这

种方法是为了让读者以后可以自己基于 openwrt 原版的源码基础上进行修改。另外一种是直接卓钛科技提供的 openwrt 源码（也是基于 openwrt 官方的源码修改而来）。这两种方法读者二选一，下面介绍这两种方法。

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) ubuntu 终端输入如下命令，下载源码。

```
git clone -b chaos_calmer git://github.com/openwrt/openwrt.git
```

```
robinson@robinson-virtual-machine:~/embedded/new_disk$ git clone -b chaos_calmer git://github.com/openwrt/openwrt.git
Cloning into 'openwrt'...
```

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"openwrt"的文件夹，里面存放的就是 OpenWrt 的源码了。

- 2) 在 ubuntu 终端中输入以下命令：

```
cd openwrt //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 JS7628 上运行的，但是有一些功能会不正常，比如按键，LED 指示灯，串口波特率等。

4.3.2.2. 用 JS7628 配套的 openwrt 源码

在“JS7628 开发板配套资料\JS7628 开发板源码\openwrt 源码\openwrt_CC_mt76xx_zhuotk_source_xxxx.tar.bz2”（xxxx 是日期）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 JS7628 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 ./openwrt_CC_mt76xx_zhuotk_source_xxxx.tar.bz2 -C ./ //解压源码包
```

```
cd ./openwrt_CC_mt76xx_zhuotk_source //进入源码目录
```

```
./scripts/feeds install -a //安装更新的软件包，必须执行，否则开发板功能会不正常
```

提示：这个源码编译出来的 openwrt 固件可以在 JS7628 开发板上正确运行。

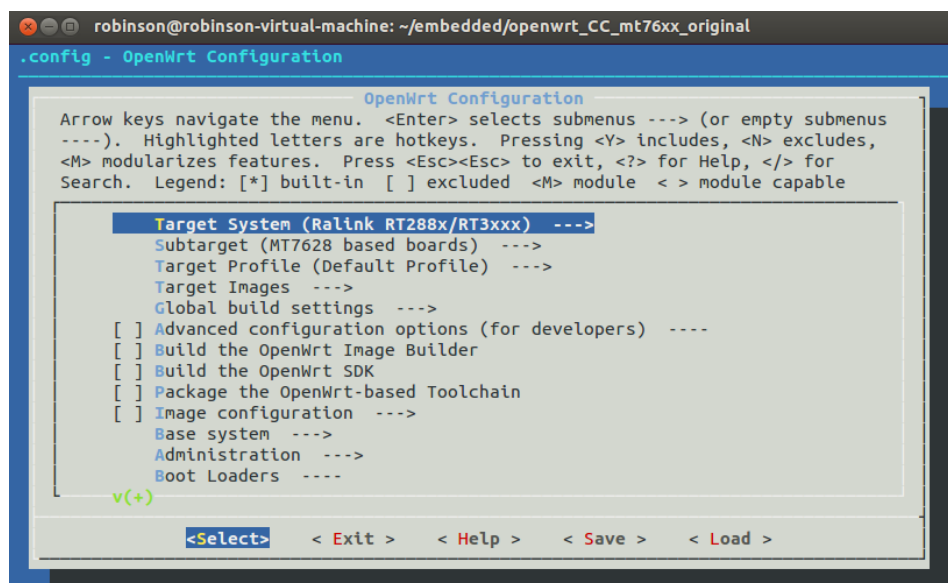
4.3.3. make menuconfig 配置系统功能

4.3.3.1. 配置界面简介

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。获取到 openwrt 源码后，我们在 OpenWrt 源码根目录下输入以下命令：

```
make menuconfig //进入 OpenWrt 系统配置界面
```

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。

这里暂时不做过多的系统配置介绍，以后会根据读者的反映，考虑是否增加介绍。

4.3.3.2. 有关 “IOT-device”、“IOT-gateway” 模式的配置

进入到菜单

Kernel modules --->

Other modules --->

<*> kmod-sdhci-mt7620

如果选上 “kmod-sdhci-mt7620” 则开发板支持 micro SD 卡接口功能，同时启用 “IOT-device” 功能，如果选择不启用则开发板不支持 micro SD 卡接口功能，并启用 “IOT-gateway” 功能，

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 “-jx” (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 “JS7628 开发板配套资料\JS7628 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 ubuntu 里面的 “openwrt 源码根目录\dl” 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）。这里推荐 “Green 加速器”，网站是 “<http://gisq.me/412332>” 或 “<http://www.jsqgreen.com>” 或者 **百度贴吧搜索 “greenvpn”** 看置顶帖子的网址（注意：百度搜索很多是假的 “Green VPN” 网站），作者长期用这个 Green VPN 下载国外资源，速度很快，并且可以注册免费试用账号。

最后，如果编译成功会出现类似下图的提示


```

Generating index for package ./ntfs-3g_2014.2.15-1-fuseint_ramips_24kec.ipk
Generating index for package ./wifidog_1.2.1-1_ramips_24kec.ipk
Generating index for package ./luci-app-firewall_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci-base_git-17.002.25203-6cf07ec-1_ramips_24kec.ipk
Generating index for package ./luci-i18n-base-en_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci-i18n-base-zh-cn_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci-i18n-firewall-en_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci-i18n-firewall-zh-cn_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci-lib-ip_git-17.002.25203-6cf07ec-1_ramips_24kec.ipk
Generating index for package ./luci-lib-nixio_git-17.002.25203-6cf07ec-1_ramips_24kec.ipk
Generating index for package ./luci-mod-admin-full_git-17.002.25203-6cf07ec-1_ramips_24kec.ipk
Generating index for package ./luci-proto-ipv6_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci-proto-ppp_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci-theme-bootstrap_git-17.002.25203-6cf07ec-1_all.ipk
Generating index for package ./luci_git-17.002.25203-6cf07ec-1_all.ipk
Signing package index...
make[2]: Leaving directory `/home/robinson/embedded/new_disk/openwrt_CC_mt76xx'
make[1]: Leaving directory `/home/robinson/embedded/new_disk/openwrt_CC_mt76xx'
robinson@robinson-virtual-machine:~/embedded/new_disk/openwrt_CC_mt76xx$

```

编译完成后的 JS7628 的 OpenWrt 固件

“openwrt-ramips-mt7628-mt7628-squashfs-sysupgrade.bin” 可以在 “openwrt / bin/ramips/” 下找到

```

robinson@robinson-virtual-machine:~/embedded/new_disk/openwrt_CC_mt76xx$ ls bin/ramips/
md5sums                                openwrt-ramips-mt7628-vmlinux.bin
openwrt-ramips-mt7628-LinkIt7688-squashfs-sysupgrade.bin  openwrt-ramips-mt7628-vmlinux.elf
openwrt-ramips-mt7628-mt7628-squashfs-sysupgrade.bin    packages
openwrt-ramips-mt7628-root.squashfs                     sha256sums
openwrt-ramips-mt7628-uImage.bin

```

IPK 安装包在 “openwrt/ bin/ramips/packages/” 目录下，如下图所示

```

robinson@robinson-virtual-machine:~/embedded/new_disk/openwrt_CC_mt76xx$ ls bin/ramips/packages/
base luci management packages routing telephony

```

图中的三个文件夹 “base”、“luci”、“management”、“routing”、“telephony”、“packages” 均放有 IPK 安装包，其中

“base” —— 里面存放的是系统基本的安装包

“luci” —— 存放的是 LUCI 网页相关的安装包

“packages” —— 存放其他很多的功能安装包

其他安装包存放目录请读者自行查看。

4.3.5. 刷新 OpenWrt 固件

有关如何刷新 OpenWrt 固件，在《JS7628 开发板使用手册》中的“开发板固件镜像烧录说明”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 “Build the OpenWrt based Toolchain”，如下图所示

```

Subtarget (MT7628 based boards) --->
Target Profile (Default Profile) --->
Target Images --->
Global build settings --->
[ ] Advanced configuration options (for developers)
[ ] Build the OpenWrt Image Builder
[ ] Build the OpenWrt SDK
[*] Package the OpenWrt-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 源码目录/ bin/ramips/

OpenWrt-Toolchain-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-i686.tar.bz2” 下找到。

（提示：不方便编译的用户，可以在 “JS7628 开发板配套资料\JS7628 开发板固件镜像安装

包\交叉工具链”中找到编译好的 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 的根目录下，输入如下命令：

```
sudo tar
jxvf ./bin/ramips/OpenWrt-Toolchain-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-i686
.tar.bz2 -C /opt/
```

在提示输入超级用户密码后，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“JS7628 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”。

这里告诉大家一个非常实用的小技巧，如果用手敲上面的命令，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

执行命令

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

在文件最后添加以下两行配置

```
export
PATH=/opt/OpenWrt-Toolchain-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-i686/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin:$PATH
```

```
export STAGING_DIR=/your_openwrt_path/staging_dir
```

效果如下

```
export PATH=/opt/OpenWrt-Toolchain-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-i686/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin:$PATH
export STAGING_DIR=/your_openwrt_path/staging_dir
```

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

```
mipsel-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。

```

Reading specs from /opt/OpenWrt-Toolchain-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-i
c-4.8-linaro_uClibc-0.9.33.2/bin/./lib/gcc/mipsel-openwrt-linux-uclibc/4.8.3/specs
COLLECT_GCC=mipsel-openwrt-linux-uclibc-gcc.bin
COLLECT_LTO_WRAPPER=/opt/OpenWrt-Toolchain-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-
cc-4.8-linaro_uClibc-0.9.33.2/bin/./libexec/gcc/mipsel-openwrt-linux-uclibc/4.8.3/lto-wrapper
Target: mipsel-openwrt-linux-uclibc
Configured with: /home/robinson/embedded/openwrt_CC_mt7628/build_dir/toolchain-mipsel_24kec+ds
gcc-linaro-4.8-2014.04/configure --with-bugurl=https://dev.openwrt.org/ --with-pkgversion='Ope
8' --prefix=/home/robinson/embedded/openwrt_CC_mt7628/staging_dir/toolchain-mipsel_24kec+dsp_g
uild-i686-linux-gnu --host=i686-linux-gnu --target=mipsel-openwrt-linux-uclibc --with-gnu-ld -
-libgomp --disable-libmudflap --disable-multilib --disable-nls --with-host-libstdcxx=-lstdc++
.....
nux-i686/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin/././usr/lib -lgc
n-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-i686/toolchain-mipsel_24kec+dsp_gcc-4
gcc/mipsel-openwrt-linux-uclibc/4.8.3/crtend.o /opt/OpenWrt-Toolchain-ramips-mt7628_gcc-4.
oolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33.2/bin/./lib/gcc/mipsel-openwrt-lin
enwrt-linux-uclibc/lib/crti.o
/opt/OpenWrt-Toolchain-ramips-mt7628_gcc-4.8-linaro_uClibc-0.9.33.2.Linux-i686/toolchain-m
-0.9.33.2/bin/./lib/gcc/mipsel-openwrt-linux-uclibc/4.8.3/./././././mipsel-openwrt-lin
bedded/openwrt_CC_mt7628/build_dir/toolchain-mipsel_24kec+dsp_gcc-4.8-linaro_uClibc-0.9.33
/mips/crt1.S:95: undefined reference to `main'
collect2: error: ld returned 1 exit status

```

上图的错误提示不用管。

4.6. openwrt 源码简介

下载 openwrt 源码后,源文件如下图所示,下面我们来一一解释

BSDmakefile	Config.in	feeds	include	Makefile	README	scripts	toolchain
config	docs	feeds.conf.default	LICENSE	package	rules.mk	target	tools

1) scripts

存放了一些脚本,使用了 bash,Python,perl 等多种脚本语言.编译过程中,用于第三方软件包管理的 feeds 文件也是在这个目录当中.在编译过程中,使用到的脚本也统一放在这个目录中。

2) tools

编译时,主机需要使用一些工具软件,tools 里包含了获取和编译这些工具的命令.软件包里面有 Makefile 文件,有的还包含了 patch.每个 Makefile 当中都有一句\$(eval \$(call HostBuild)),这表明编译这个工具是为了在主机上使用的。

3) config

存放着整个系统的配置文件

4) docs

包含了整个宿主机的文件源码的介绍, 里面还有 Makefile 为目标系统生成 docs.使用 make -C docs/可以为目标系统生成文档。

5) toolchain

做过嵌入式的童鞋应该都知道交叉编译链,这个文件中存放的就是编译交叉编译链的软件包.包括:binutils,gcc,libc 等等。

6) target

openwrt 的源码可以编译出各个平台适用的二进制文件,各平台在这个目录里定义了 firmware 和 kernel 的编译过程。

7) package

存放了 openwrt 系统中适用的软件包,包含针对各个软件包的 Makefile。openwrt 定义了一套 Makefile 模板.各软件参照这个模板定义了自己的信息,如软件包的版本、下载地址、编译方式、安装地址等。在二次开发过程中,这个文件夹我们会经常打交道。事实上,通过 ./scripts/feed update -a 和 ./scripts/feed install -a 的软件包也会存放在这个目录之中。

8) include

openwrt 的很多 Makefile 都存放在这里。文件名为 *.mk 。这里的文件上是在 Makefile 里被 include 的,类似于库文件.这些文件定义了编译过程。

9) 其他

主要目录就是前面提及的 8 个,剩下的是单个文件。

Makefile

在顶层目录执行 make 命令的入口文件。

rules.mk

定义了 Makefile 中使用的一些通用变量和函数

Config.in

在 include/toplevel.mk 中我们可以看到,这是和 make menuconfig 相关联的文件。

feeds.conf.default

这个文件中可以配置下载第三方一些软件包时所使用的地址

LICENSE & README

即软件许可证和软件基本说明.其中 README 描述了编译软件的基本过程和依赖文件。

至此我们把原始目录大致浏览了一遍,再读者编译一次 openwrt 源码后,会出现一些新的目录,下面我们看看这些生成的目录,如下图所示

bin	Config.in	feeds.conf.default	LICENSE	rules.mk	tmp
BSDmakefile	dl	include	Makefile	scripts	toolchain
build_dir	docs	key-build	package	staging_dir	tools
config	feeds	key-build.pub	README	target	

10) feeds

openwrt 的附加软件包管理器的扩展包索引目录.有点绕,简单来说就是下载管理软件包的.默认的 feeds 下载有 packages、management、luci、routing、telephony。如要下载其他的软件包,需打开源码根目录下面的 feeds.conf.default 文件,去掉相应软件包前面的#号,然后更新源:

```
./scripts/feeds update -a
```

安装下载好的包:

```
./scripts/feeds install -a
```

11) build_dir

在前面的原始目录中,我们提到了 host 工具,toolchain 工具还有目标文件.openwrt 将在这个目录中展开各个软件包,进行编译.所以这个文件夹中包含 3 个子文件夹:

--host

在该文件夹中编译主机使用的工具软件

--toolchain-XXX

在该文件夹中编译交叉工具链

--target-XXX

在此编译目标平台的目标文件,包括各个软件包和内核文件.openwrt 系统的 linux 解压后的源码在

“openwrt/build_dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/linux-ramips_mt7628/linux-3.18.29/” 目录下。

12) bin

保存编译完成后的二进制文件,包括:完整的 bin 文件,所有的 ipk 文件.

13) dl

在编译过程中使用的很多软件,刚开始下载源码并没有包含,而是在编译过程中从其他服务器下载的,这里是统一的保存目录

14) staging_dir

用于保存在 build_dir 目录中编译完成的软件.所以这里也和 build_dir 有同样的子目录结构.比如,在 target-XXX 文件夹中保存了目标平台编译好的头文件,库文件.在我们开发自己的 ipk 文件时,编译过程中,预处理头文件,链接动态库,静态库都是到这个子文件夹中.

15) tmp

从名字来看,是临时文件夹.在编译过程中,有大量中间临时文件需要保存,都是在这里.

16) logs

这个文件夹,有时可以看到,有时没有.这是因为这个文件夹保存的是,编译过程中出错的信息,只有当编译出错了才会出现.我们可以从这里获取信息,从而分析我们的软件编译为什么没有完成.

至此我们把 openwrt 的目录结构大体浏览了一遍。

4.7. 配置 JS7628 开发板硬件默认功能

在 openwrt 系统源码里面,采用 MTK (联发科) 芯片的板子硬件配置文件 DTS (Device tree source) 一般都放在 “openwrt 源码/target/linux/ramips/dts/” 目录下, 如下图所示

```
robinson@robinson-virtual-machine:~/embedded/new_disk/openwrt_CC_mt76xx$ ls target/linux/ramips/dts/
3G150B.dts      DAP-1350.dts    M4-4M.dts      PX4885-4M.dts   WIZARD8800.dts
3G300M.dts      DCS-930.dts     M4-8M.dts      PX4885-8M.dts   WIZFI630A.dts
3G-6200N.dts    DCS-930L-B1.dts MicroWRT.dts    PX4885.dtsi     WL_330N3G.dts
3G-6200NL.dts   DIR-300-B1.dts  MLW221.dts     RE6500.dts      WL_330N.dts
A5-V11.dts      DIR-300-B7.dts  MLW2.dts       RP-N53.dts      WL341V3.dts
AIBR100.dts     DIR-320-B1.dts  MOFI3500-3GN.dts rt2880.dtsi     WL-351.dts
AIR3GII.dts     DIR-600-B1.dts  MPRA1.dts      rt3050.dtsi     WLI-TX4-AG300N.dts
ALL0239-3G.dts  DIR-600-B2.dts  MPRA2.dts      rt3352.dtsi     WMR300.dts
ALL0256N-4M.dts DIR-610-A1.dts  MR-102N.dts    rt3883.dtsi     WNCE2001.dts
ALL0256N-8M.dts DIR-615-D.dts   MT7620a.dts     rt5350.dtsi     WR512-3GN-4M.dts
ALL5002.dts     DIR-615-H1.dts  mt7620a.dtsi    RT-G32-B1.dts   WR512-3GN-8M.dts
ALL5003.dts     DIR-620-A1.dts  MT7620a_MT7530.dts RT-N10-PLUS.dts WR6202.dts
AR670W.dts      DIR-620-D1.dts  MT7620a_MT7610e.dts RT-N13U.dts     WR8305RT.dts
AR725W.dts      DIR-645.dts     MT7620a_V22SG.dts RT-N14U.dts     WRTNODE.dts
ArcherC20.dts   DIR-810L.dts    mt7620a.dtsi    RT-N15.dts      WSP-1166.dts
```

JS7628 开发板的 DTS 配置文件也在这个目录下, 文件名是 “MT7628.dts”。打开这个 “MT7628.dts” 文件, 我们可以看到 JS7628 开发板相关的硬件配置信息, 如下所示


```

1 /dts-v1/;
2
3 /include/ "mt7628an.dtsi"
4
5 / {
6     compatible = "mediatek,mt7628an-eval-board", "mediatek,mt7628an-soc";
7     model = "Mediatek MT7628AN evaluation board";
8
9     chosen {
10         bootargs = "console=ttyS0,115200";
11     };
12
13     memory@0 {
14         device_type = "memory";
15         reg = <0x0 0x10000000>; //256MB RAM
16         // reg = <0x0 0x80000000>; //128MB RAM
17         // reg = <0x0 0x40000000>; //64MB RAM
18     };
19
20     pinctrl {
21

```

（由于这个文件内容比较多，这里只做部分截图）

1) 修改支持的内存大小

JS7628 开发板有 3 种内存配置，分别是 64MB、128MB、256MB，读者可以在这个 DTS 文件大约第 16~18 行找到这个配置

```

16         reg = <0x0 0x10000000>; //256MB RAM
17         // reg = <0x0 0x80000000>; //128MB RAM
18         // reg = <0x0 0x40000000>; //64MB RAM

```

根据后面的注释，自行去掉对该行的注释然后启用这个内存配置，重新编译出固件。

注意：选择和硬件不匹配的配置，可能会导致系统运行不正常。

2) 修改支持的 Flash 大小

找到大约第 118~120 行

```

118         // reg = <0x50000 0x7b0000>; //8MB flash
119         // reg = <0x50000 0xfb0000>; //16MB flash
120         reg = <0x50000 0x1fb0000>; //32MB flash

```

根据后面的注释，自行去掉对该行的注释然后启用这个 flash 配置，重新编译出固件。

注意：选择和硬件不匹配的配置，可能会导致系统运行不正常。

3) 配置管脚复用功能

MT7628 芯片里面有很多管脚是复用，这里我们以“spis”功能为例做介绍如何配置 JS7628 开发板的管脚功能复用。

找到 DTS 文件中的

```

43     spis {
44         ralink,group = "spis";
45         ralink,function = "pwm_uart2";
46     };

```

从上图可以看出 MT7628 芯片的 SPIS 管脚默认配置为“pwm_uart2”功能,具体可以配置哪几种功能是在

“openwrt/build_dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/linux-ramips_mt7628/linux-3.18.29/arch/mips/ralink/mt7620.c”中定义的，打开该文件找到下图所示行

```

179 static struct rt2880_pmx_func spis_grp_mt7628[] = {
180     FUNC("pwm_uart2", 3, 14, 4),
181     FUNC("util", 2, 14, 4),
182     FUNC("gpio", 1, 14, 4),
183     FUNC("spis", 0, 14, 4),
184 };

```

由上图可知，我们可以把“spis”管脚组定义为“pwm_uart2”、“util”、“gpio”、“spis”4 中功能，这个和 MT7628 的 datasheet

（“JS7628 开发板配套资料\芯片元器件手册\MT7628\MT7628 Datasheet.pdf”）中介绍的完全对应，如下图所示

2.3.20 SPIS pin share scheme					
Controlled by the EPHY_APGPIO_AIO_EN[4:1] and SPIS_MODE registers					
	4'b0000	4'b1111			
Pin Name		2'b00	2'b01	2'b10	2'b11
MDI_TP_P1	MDI_TP_P1	SPIS_CS	GPIO#14		PWM_CHO
MDI_TN_P1	MDI_TN_P1	SPIS_CLK	GPIO#15		PWM_CH1
MDI_RP_P1	MDI_RP_P1	SPIS_MISO	GPIO#16		UART_TXD2
MDI_RN_P1	MDI_RN_P1	SPIS_MOSI	GPIO#17		UART_RXD2

如果读者需要将这几个管脚恢复为 GPIO 功能，可以把 DTS 文件修改为

```

43         spis {
44             ralink,group = "spis";
45             ralink,function = "gpio";
46         };

```

然后重新编译固件烧录，这样这组管脚就变成了 GPIO 功能，可以通过之前“GPIO 简单控制”一节介绍的方法或者编程进行控制。其他组管脚和这里介绍的方法大同小异，请读者自行测试。

4) 使能硬件网口灯管脚功能

提示：本小节适用于采用我们的源码、核心板并需要自行设计底板用到多个网口指示灯的读者。直接用我们开发板的用户，一般不需要关心这一节的配置。

WAN LED	WAN LED	13	EPHY_LED0_N_JTDO/GPIO43
LAN1 LED	LAN1 LED	12	EPHY_LED1_N_JTDI/GPIO42
USER KEY2	USER KEY2	11	EPHY_LED2_N_JTMS/GPIO41
LAN2 LED	LAN2 LED	10	EPHY_LED3_N_JTCLK/GPIO40
WLAN LED	WLAN LED	9	EPHY_LED4_N_JTRST N/GPIO39

```

p0_led {
    ralink,group = "p0_led_an";
    ralink,function = "gpio";
};
p1_led {
    ralink,group = "p1_led_an";
    ralink,function = "gpio";
};
p2_led {
    ralink,group = "p2_led_an";
    ralink,function = "gpio";
};
p3_led {
    ralink,group = "p3_led_an";
    ralink,function = "gpio";
};
p4_led {
    ralink,group = "p4_led_an";
    ralink,function = "gpio";
};

```

如上面两图所示，在 JS7628 开发板底板原理图中和 DST 文件中，网口指示灯管脚“EPHY_LEDx_N_xxx”默认配置为“GPIO”模式，并且部分管脚作为按键功能，部分管脚作为网络通信指示。由于目前在 openwrt 系统中，管脚在这种“GPIO”模式下，只能指示网络数据通信，无法指示网口的连接状态。

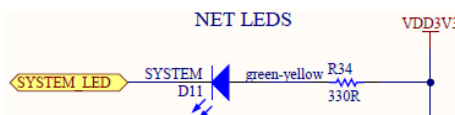
为了让网口灯的管脚能够实现“亮灭指示网口连接”、“闪烁指示数据通信”的这种硬件网口灯功能（EPHY_LEDx_N_xxx 对应指示网口 x），下面示范通过修改 DTS 文件配置，使能 5 个硬件网口指示灯

```

p0_led {
    ralink,group = "p0_led_an";
    ralink,function = "ephy";
};
p1_led {
    ralink,group = "p1_led_an";
    ralink,function = "ephy";
};
p2_led {
    ralink,group = "p2_led_an";
    ralink,function = "ephy";
};
p3_led {
    ralink,group = "p3_led_an";
    ralink,function = "ephy";
};
p4_led {
    ralink,group = "p4_led_an";
    ralink,function = "ephy";
};

```

如上图所示将之前配置中的“gpio”改为“ephy”，修改完成后，这 5 个网口灯管脚就可以指示 5 个网口的连接、通讯状态了。读者可以自行编译固件测试。这些硬件网口灯管脚的驱动电路和开发板底板原理图的一致（低电平有效），参考下图



注意：如果读者是在 JS7628 开发板上测试这个编译后的固件，由于线路连接原因（读者可以自行参考 JS7628 底板原理图），开发板将只能指示网口灯 0、网口灯 1。

提示：在“JS7628 开发板配套资料\学习资料\linux 入门学习资料\Device Tree Usage - eLinux.pdf”是有关 DTS 的详细介绍,感兴趣的读者可以阅读学习。

4.8. 编译第一个“Hello World”应用程序

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

```
mipsel-openwrt-linux-gcc hello_world.c -o hello_world //用 mipsel-openwrt-linux-gcc 编译
“hello_world.c” 文件，并生成 “hello_world” 可执行文件。
```

最后将文件传输到开发板的 “/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 //添加可执行权限
/tmp/hello_world //执行 “hello_world” 可执行文件
```

效果如下图所示。

```
root@zhuotk:/# ./tmp/hello_world
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
```

用户可以用 “Ctrl+c” 组合键，停止程序的运行。

4.9. 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 源码根目录，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 “编译第一个 ‘helloworld’ 应用程序” 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))
```

保存退出，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/ramips/packages/base/hello_world_1.0_ramips_24kec.ipk”

安装此安装包后，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“JS7628 开发板配套资料\JS7628 开发板源码\hello_world IPK 源码”目录中。

提示：在“JS7628 开发板配套资料\学习资料\openwrt 学习资料\ Creating packages [OpenWrt Wiki].pdf”是 openwrt 官方创建 IPK 包的介绍，有兴趣的读者可以自行阅读。

4.10. 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”，不过这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“JS7628 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_3.18.29-1_ramips_24kec.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
                007 1
```

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：

“JS7628 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”

“JS7628 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“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“JS7628 开发板配套资料\开发板源码\gpio_control_app”到“openwrt 源码根目录/package/utis/”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_ramips_24kec.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_control_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 GPIO18、GPIO19 这两个 GPIO 进行测试，并且也需要烧录 IOT device 固件。测试 GPIO18 命令如下

```
gpio_control_app demo1 18
```

执行以上命令后，GPIO18 将会出现“1 秒高电平 1 秒低电平 1 秒高电平...”的循环（ctrl+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）。测试 GPIO19 的方法同上。

参考文档：

“JS7628 开发板配套资料\学习资料\openwrt 学习资料\ Creating packages [OpenWrt Wiki].pdf”

4.12. 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“JS7628 开发板配套资料\开发板源码\net_control_server”到“openwrt 源码根目录/package/utis/”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_ramips_24kec.ipk”，将此 IPK 传输到开发板并安装。安装完成后，执行

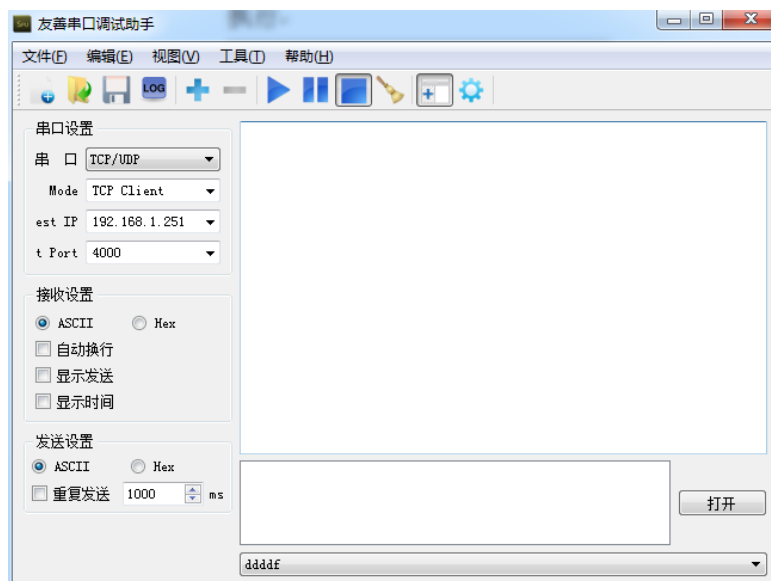
```
net_control_server tcp_server 4000
```

如下图所示

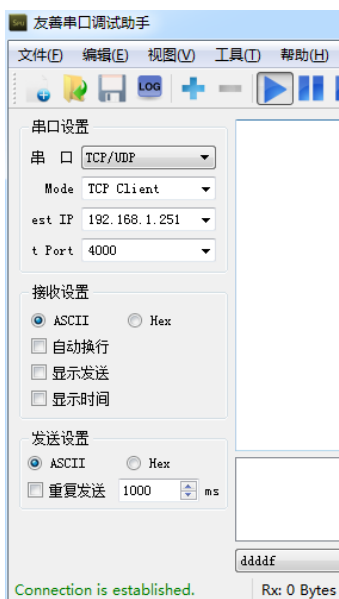
```
root@zhuoTK:/# net_control_server tcp_server 4000
***net_control_server***
listening...
■
```

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



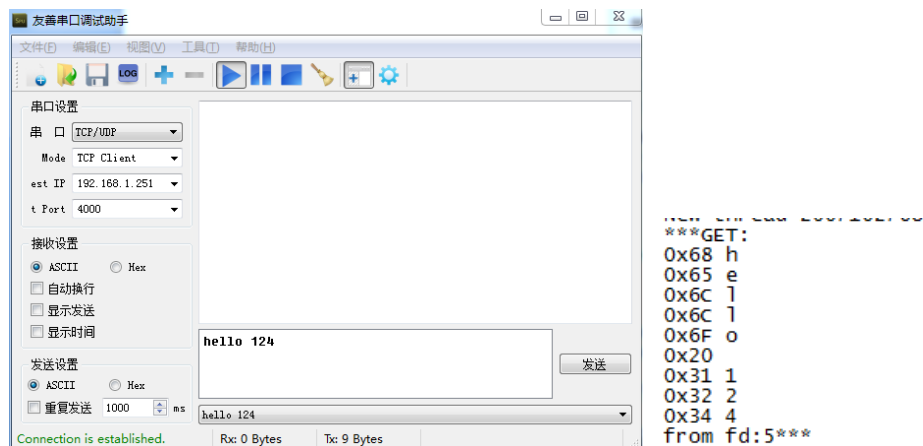
然后点击  连接服务器，结果如下图



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：

“JS7628 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”

“JS7628 开发板配套资料\学习资料\openwrt 学习资料\ Creating packages [OpenWrt Wiki].pdf”

4.13. 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“JS7628 开发板配套资料\开发板源码\net_control_client”到“openwrt 源码根目录/package/utis/”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
```

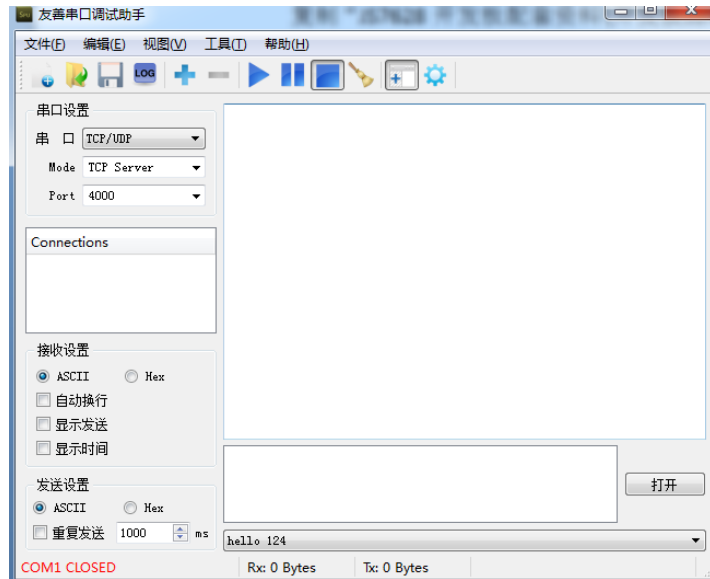
保存退出，执行

```
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_ramips_24kec.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

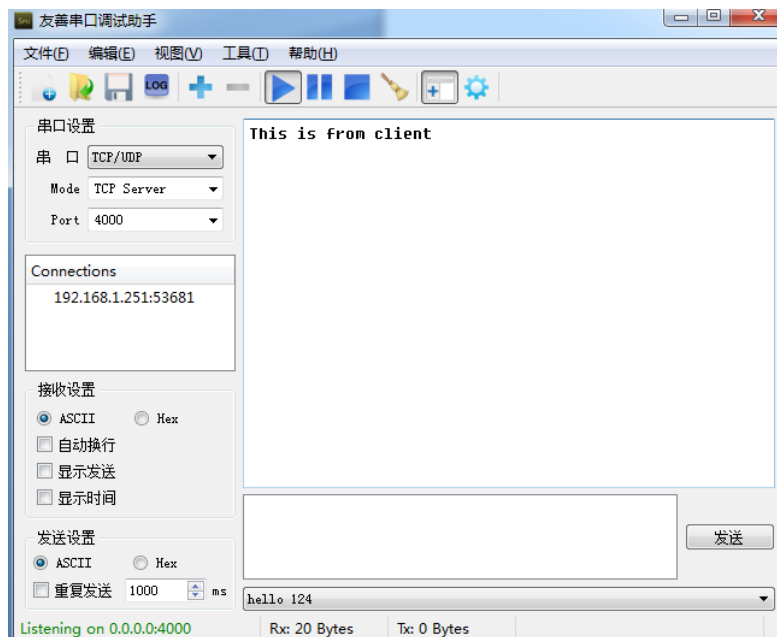


开发板执行

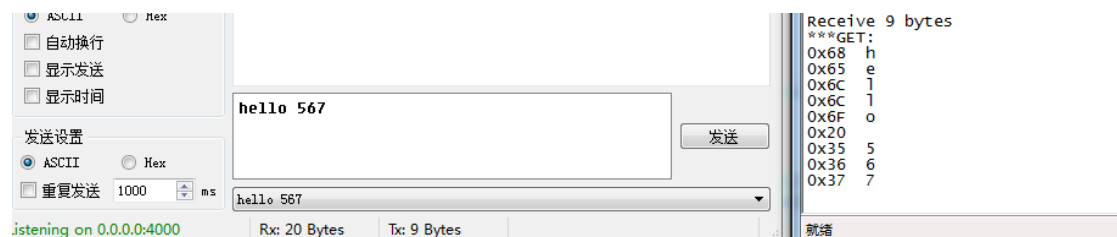
`net_control_client tcp_connect 192.168.1.xxx 4000` #192.168.1.xxx 为读者自己电脑的 IP
结果如下图所示（这里作者电脑 IP 是“192.168.1.110”）

```
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：

“JS7628 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”

“JS7628 开发板配套资料\学习资料\openwrt 学习资料\ Creating packages [OpenWrt Wiki].pdf”

4.14. LUCI 界面修改

请读者参考“JS7628 开发板配套资料\学习资料\openwrt 学习资料\openwrt 学习资料\luci 学习笔记本.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”。

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，dev.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，不过还是可以找到许多的 openwrt 资料。
- 4) 百度。

6. 常见问题及解答

7. 历史版本说明

版本	时间	修改说明
V1.0	2016.12.27	JS7628 开发板入门教程初始版本。
V1.1	2017.01.02	增加了一些说明
V1.2	2017.01.12	增加了“基本硬件功能测试说明”，修改了“交换机子菜单”的说明
V1.3	2017.01.15	增加了“打造本地音乐播放器”的说明。增加了“挂载 U 盘、micro SD（TF）卡”的说明等其他说明。
V1.4	2017.03.17	增加了“挂载 4G 网卡上网”、“开发板挂 VPN 上网”、“网络串口透传”等一部分之前预告章节的内容
V1.4.1	2017.09.28	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”的说明
V1.4.2	2017.09.28	修改了复制“dl”文件夹的说明。目录修整。添加了“服务器中转访问”、“网络通讯服务器”、“网络通讯客户端”、“openwrt 源码简介”等节
V1.4.3	2018.08.10	修正了“服务器中转访问”里面的说明
V1.4.4	2018.09.04	添加了安装 openwrt 源码更新时候的红字提示。

		添加“使能硬件网口灯管脚”的说明。 添加 <code>hello_world</code> 执行权限的命令。 添加了有关“STA 模式设置”的 IP 网段注意事项。
--	--	--