



一、引言

二、实验过程与分析

实验过程

OpenWrt

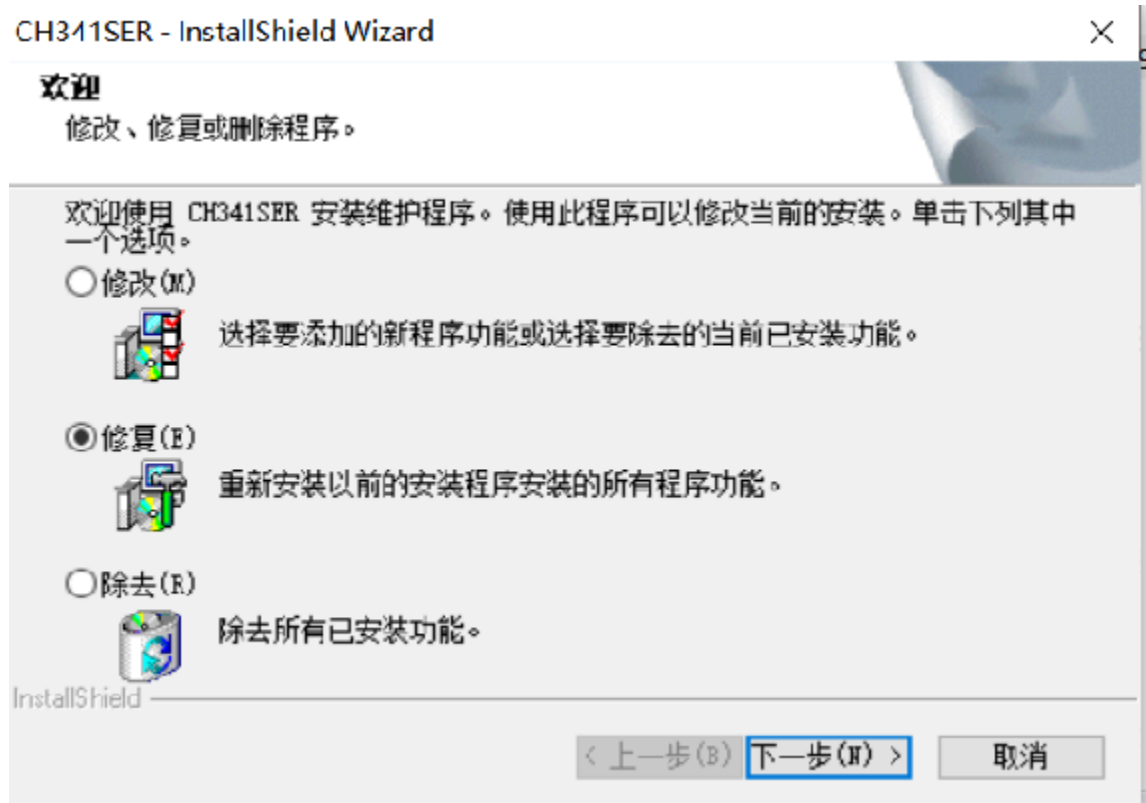
- 优势:
 - OpenWRT支持各种处理器架构, 无论是对ARM, X86, PowerPC, 到工具链(toolchain), 到内核(linux kernel), 到软件包/packages, 到命令即可方便快速地定制一个具有特定功能的嵌入式系统来定制。
- 劣势:

- 由于CPU内核体系不同，造成很多应用程序移植到OpenWrt。
- 由于ADSL硬件模块的驱动程序没有开放源代码，造成很多（DB120除外）。
- 由于OpenWRT并不是官方发布的路由器固件，所以要使用（高）。

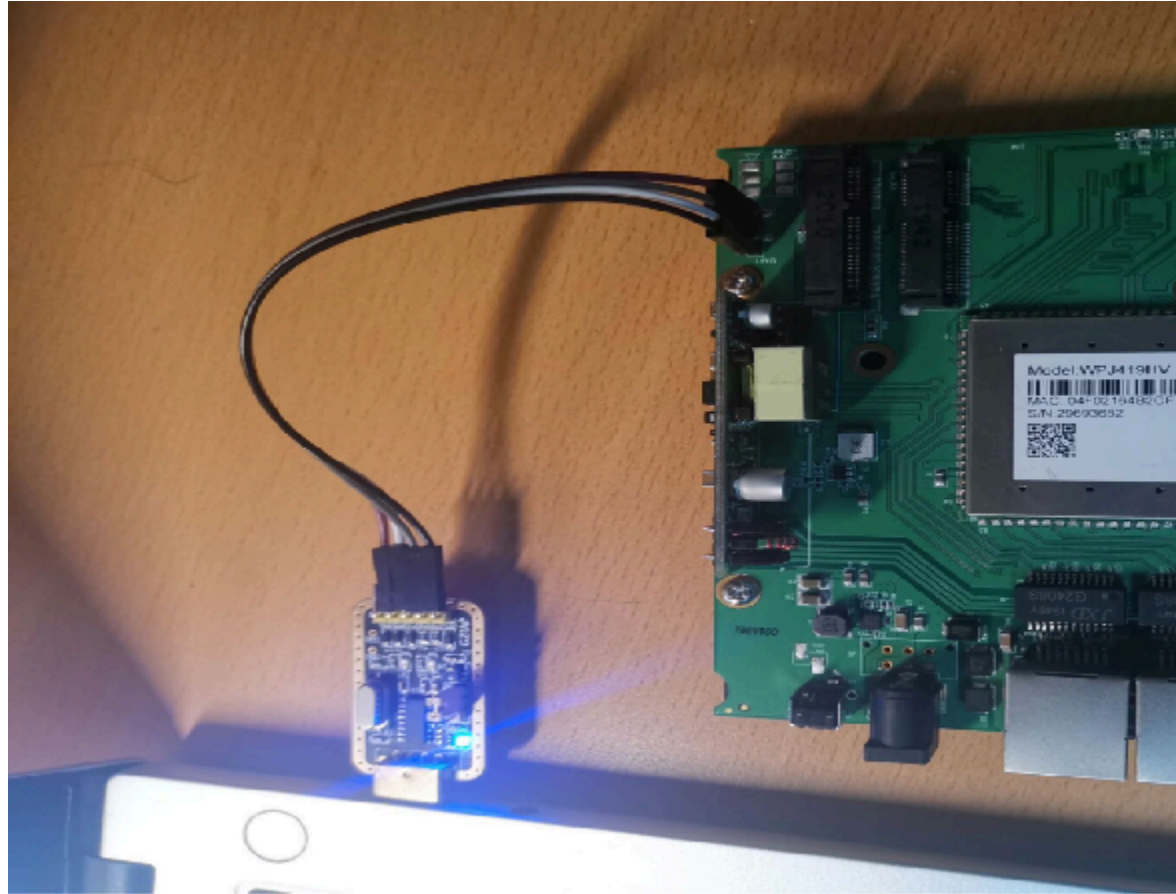
对于Openwrt的劣势，我们在实验过程中有深入的体会。由于操作不当，我们不止一

安装CH340驱动程序

CH340 是一个USB 总线的转接芯片，实现USB 转串口、USB 转IrDA 红外或者USB 转在串口方式下，CH340 提供常用的MODEM联络信号，用于为计算机扩展异步串口，需要安装USB转TTL的芯片CH340的驱动装置才能通过USB驱动串口连接开发板。



连接开发板



注意RX和TX不要接反，否则无法写入。

这里正确连接信号线与各个端口十分重要，否则可能会造成烧毁接口甚至毁坏开发板。

CX属性

类别(C):

- 连接
 - 用户身份验证
 - 登录提示符
 - 登录脚本
 - SSH
 - 安全性
 - 隧道
 - SFTP
 - TELNET
 - RLOGIN
 - 目 |
 - 代理
 - 保持活动状态
 - 终端
 - 键盘
 - VT 模式
 - 高级
 - 外观
 - 窗口
 - 交出
 - 高级
 - 跟踪
 - 响铃
 - 日志记录
 - 文件传输
 - X/YMODEM
 - ZMODEM

连接 > 串口

常规

端口号(P): COM3

波特率(B): 115200

数据位(D): 8

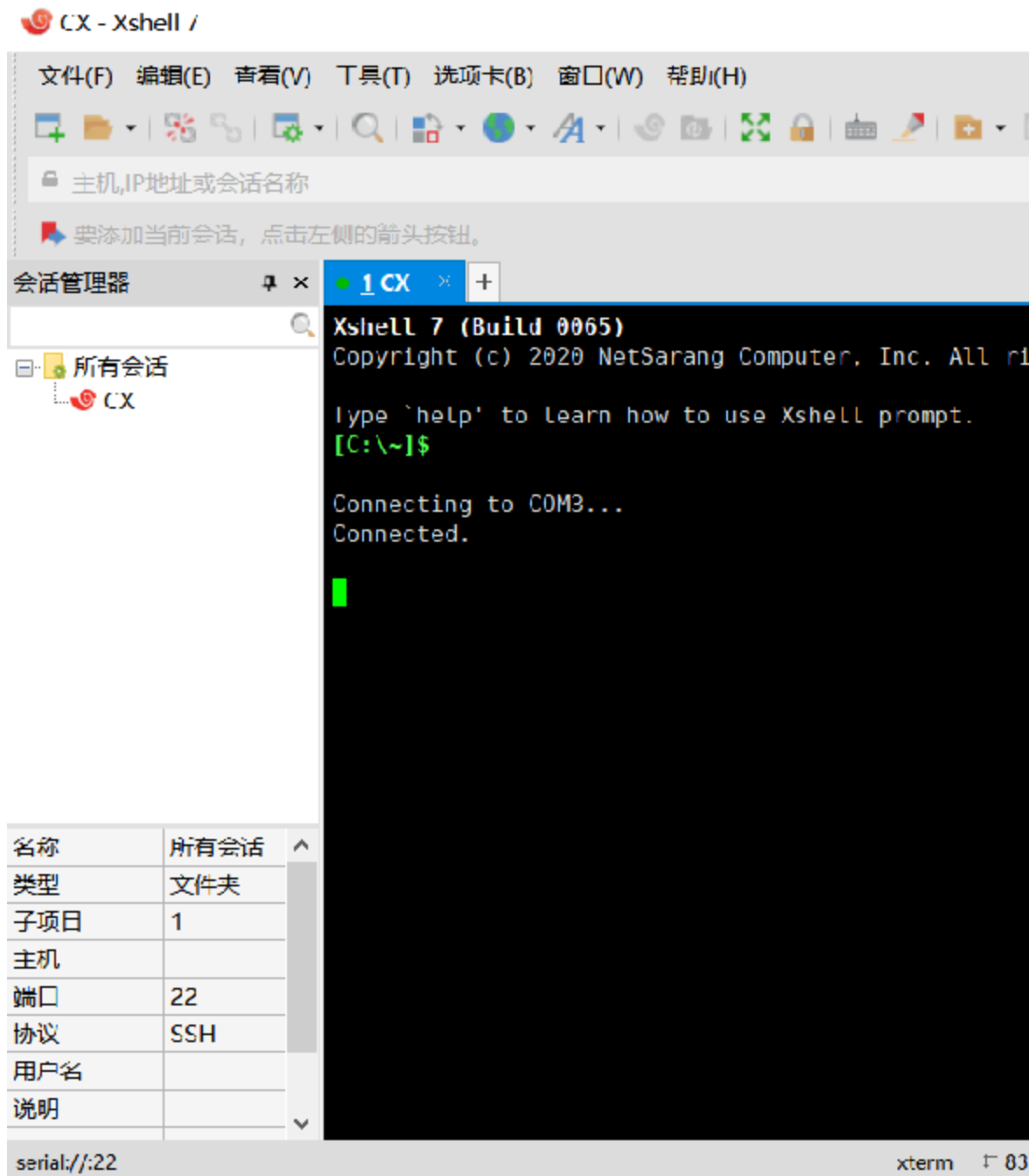
停止位(S): 1

奇偶校验(A): 无

流控制(F): 无

连接

确定



启动TFTP服务

TFTP

- TFTP (Trivial File Transfer Protocol, 简单文件传输协议) 是TCP/IP协议族中一个简单、开销不大的文件传输服务。
- TFTP是一个传输文件的简单协议，它基于UDP协议而实现，但是我们通常是在进行小文件传输的。因此它不具备通常的FTP的许多功能，它只能传输8位数据。传输中有三种模式：netascii，这是8位的ASCII码形式，另

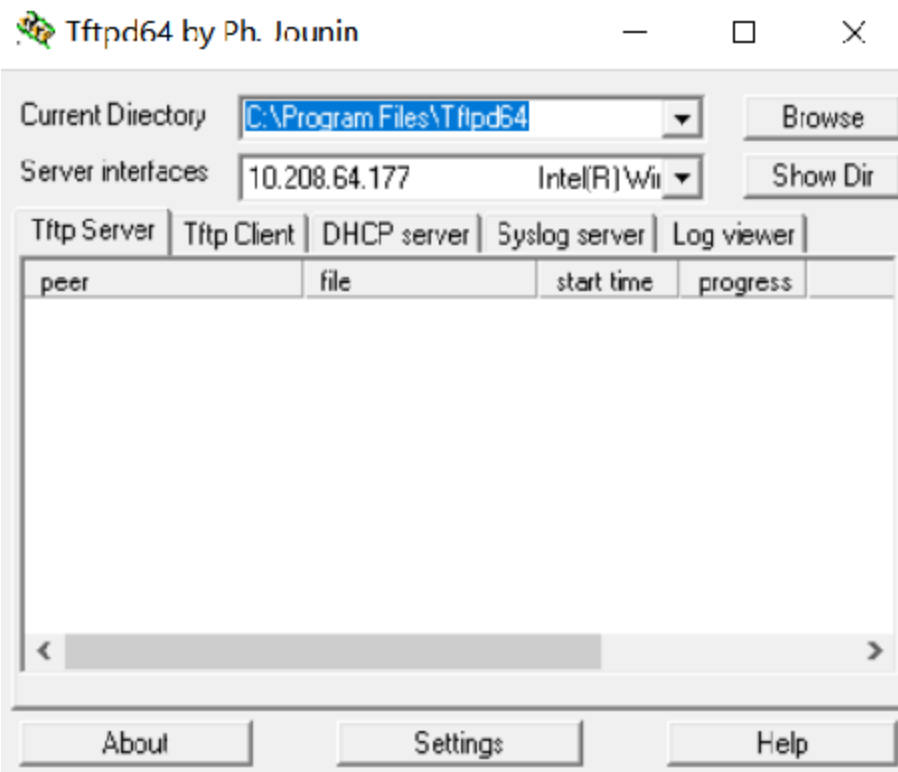
将返回的数据直接返回给用户而不是保存为文件。

在windows中开启该服务



手动设置IP地址

手动设置IP地址，连接开发板和电脑



编辑 IP 设置

手动

IPv4

开

IP 地址

192.168.1.10

子网前缀长度

24

网关

192.168.1.1

首选 DNS

备用 DNS

保存

取消

U-boot刷机

U-boot

- Das U-Boot 是一个主要用于嵌入式系统的引导加载程序，可以支持多种处理器，如68k、Nios与MicroBlaze。这也是一套在GNU通用公共许可证之下发布的。
- 其有许多优点：
 - 开放源码；
 - 支持多种嵌入式操作系统内核，如Linux、NetBSD, VxWorks；
 - 支持多个处理器系列，如PowerPC、ARM、x86、MIPS；
 - 较高的可靠性和稳定性；
 - 高度灵活的功能设置，适合U-Boot调试、操作系统不同引导

- 丰富的设备驱动源码，如串口、以太网、SDRAM、FLASH。
- 较为丰富的开发调试文档与强大的网络技术支持；
- U-Boot可支持的主要功能:
 - 系统引导支持NFS挂载、RAMDISK(压缩或非压缩)形式的根文件系统。
 - 基本辅助功能强大的操作系统接口功能；可灵活设置、传递环境变量；支持目标板环境参数多品种发布，尤以Linux支持最为强劲；支持目标板环境参数多品种发布。
 - CRC32校验可校验FLASH中内核、RAMDISK镜像文件是否完整。
 - 设备驱动串口、SDRAM、FLASH、以太网、LCD、NVRAM。
 - 上电自检功能SDRAM、FLASH大小自动检测；SDRAM故障检测。
 - 特殊功能XIP内核引导；

刷机

- 进入U-boot界面后，先将固件通过TFTP传入开发板中，再按照刷机指令

```
(IPQ40xx) # tftpboot 1.ubi
eth0 PHY0 Down Speed :10 Half duplex
eth0 PHY1 Down Speed :10 Half duplex
eth0 PHY2 Down Speed :10 Half duplex
eth0 PHY3 Down Speed :10 Half duplex
eth0 PHY4 up Speed :1000 Full duplex
Using eth0 device
TFTP from server 192.168.1.10; our IP address
Filename '1.ubi'.
Load address: 0x84000000
Loading: #####
#####
#####
#####
#####
#####
#####
#####
#####
done
Bytes transferred = 7602176 (740000 hex)
```



```
config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config globals 'globals'
    option ula_prefix 'fd32:0000:0000:0000::'

config interface 'lan'
    option type 'bridge'
    option ifname 'eth1'
    option proto 'static'
    option ipaddr '192.168.1.1'
    option netmask '255.255.255.0'
    option ip6assign '60'

config interface 'wan'
    option ifname 'eth0'
    option proto 'dhcp'
```

固件更新

- 不进行固件更新的话，luci安装可能会失败
- 先下载最新的固件
- 再使用sysupgrade更新

```
root@OpenWrt:/# wget http://downloads.openwrt.org/snapshots/targets/ipq40xx-generic/openwrt-ipq40xx-generic-compex_wpj419-squashfs-nand-sysupgrade.bin
Downloading 'http://downloads.openwrt.org/snapshots/targets/ipq40xx-generic/openwrt-ipq40xx-generic-compex_wpj419-squashfs-nand-sysupgrade.bin'
Connecting to 168.119.138.211:80
Redirected to /snapshots/targets/ipq40xx-generic/openwrt-ipq40xx-generic-compex_wpj419-squashfs-nand-sysupgrade.bin on downloads.openwrt.org
Writing to 'openwrt-ipq40xx-generic-compex_wpj419-squashfs-nand-sysupgrade.bin' 100% |*****|
Download completed (7106860 bytes)
root@OpenWrt:/# ls
bin
dev
etc
lib
mnt
openwrt-ipq40xx-generic-compex_wpj419-squashfs-nand-sysupgrade.bin
overlay
proc
rom
root
sbin
sys
tmp
usr
var
www
root@OpenWrt:/# sysupgrade openwrt-ipq40xx-generic-compex_wpj419-squashfs-nand-sysupgrade.bin
Image not in /tmp, copying...
```

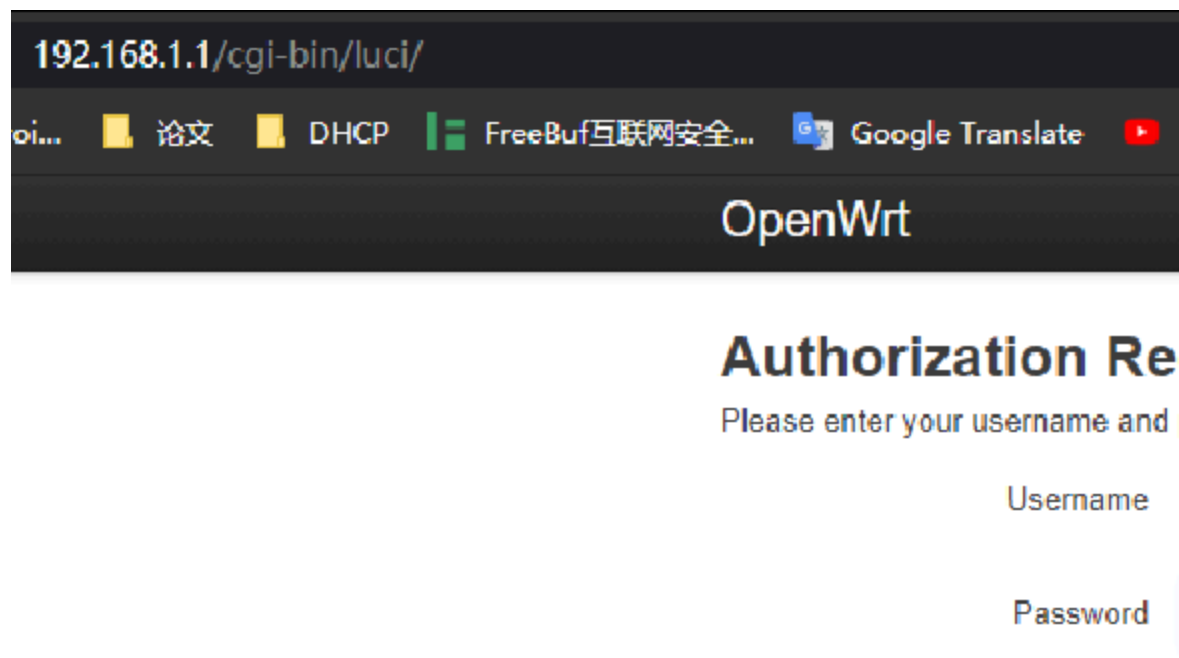
安装

- 使用opkg安装 luci

```
root@OpenWrt:/# opkg install luci
Installing luci (git-20.074.84698-ead5e81) to root...
Downloading https://downloads.openwrt.org/snapshots/packages-20.074.84698-ead5e81_all.ipk
Installing luci-lib-base (git-20.232.39649-1f6dc29) to root...
Downloading https://downloads.openwrt.org/snapshots/packages-20.232.39649-1f6dc29_all.ipk
Installing rpcd-mod-file (2021-05-05-7a560a1a-1) to root...
Downloading https://downloads.openwrt.org/snapshots/packages-2021-05-05-7a560a1a-1_arm_cortex-a7_neon-vfpv4.ipk
Installing rpcd-mod-luci (20201107) to root...
Downloading https://downloads.openwrt.org/snapshots/packages-20201107_arm_cortex-a7_neon-vfpv4.ipk
Installing cgi-io (2020-10-27-ab4c3471-19) to root...
Downloading https://downloads.openwrt.org/snapshots/packages-2020-10-27-ab4c3471-19_arm_cortex-a7_neon-vfpv4.ipk
Installing luci-base (git-21.154.28269-e35041e) to root...
Downloading https://downloads.openwrt.org/snapshots/packages-21.154.28269-e35041e_arm_cortex-a7_neon-vfpv4.ipk
Installing libiwinfo-lua (2021-04-30-c45f0b58-1) to root...
Downloading https://downloads.openwrt.org/snapshots/packages-2021-04-30-c45f0b58-1_arm_cortex-a7_neon-vfpv4.ipk
```

启动luci

- /etc/init.d/uhttpd enable 将uhttpd加入开机启动
- /etc/init.d/uhttpd start 启动uhttpd服务
此时既可以在浏览器中进行路由器配置。
- 打开 <http://192.168.1.1>
 - 192.168.1.1属于IP地址的C类地址，属于保留IP，专门用于路由器



路由器配置

接入校园网

- 这里可以选择将网线插入路由器连接入网，同时也可以使用路由器的两个简单的物理层的无线中继器，而是多了更多应用层的内容。多了更多
- 在wireless界面
- 让一个网口接入校园网
- 另一个网口当作AP
- 实现无线中继器

Wireless Overview

 radio0	Qualcomm Atheros QCA9560 802.11bgn Channel: 11 (2.462 GHz) DRate: 20 Mbit/s	Restart	Scan	Add
 disabled	SSID: OpenWrt Mode: Master Wireless is disabled	Enable	Info	Remove
 -47/-95 dBm	SSID: HUAWEI-DW13MB_HiLink Mode: Client BSSID: 04:F0:21:9A:79:0A Encryption: WPA2 PSK (COMP)	Disable	Edit	Remove
 radio1	Qualcomm Atheros QCA9880 802.11nac Channel: 30 (5.100 GHz) DRate: ? Mbit/s	Restart	Scan	Add
 -4/-100 dBm	SSID: OpenWrt Mode: Master BSSID: D8:25:B0:01:88:43 Encryption: None	Disable	Info	Remove

Associated Stations

Network	MAC Address	Host	Signal / Noise	RX Rate / TX Rate
 Client "HUAWEI-DW13MB_HiLink" (vlen 1)	44:D7:91:BF:38:54	?	 -50/-55 dBm	52.0 Mbit/s, 20 MHz, MCS 11 28.9 Mbit/s, 20 MHz, MCS 9, Short GI
Save & Apply				
Save				
Remove				

Wireless Network: Client "SEU-WLAN" (radio0.network3)

Device Configuration

[General Setup](#) [Advanced Settings](#)

Status: Mode: Client | SSID: SEU-WLAN
 --- After Wireless is not associated

Wireless network is enabled: Disable

Operating frequency: Mode: N Channel: 1 (2412 MHz) Width: 20 MHz

Allow legacy 802.11b rates: ☐
 Legacy or badly behaving devices may require legacy 802.11b rates to interoperate. Airtime efficiency may be significantly reduced where these are used.

Maximum transmit power: driver default Current power: unknown
 Specifies the maximum transmit power the wireless radio may use. Depending on regulatory requirements and wireless usage, the actual transmit power may be lower.

Interface Configuration

[General Setup](#) [Wireless Security](#) [Advanced Settings](#)

Mode: Client

ESSID: SEU-WLAN

BSSID:

Network: wlan0
 Choose the network(s) you want to attach to this wireless interface or fill out the custom field to define a new network.

Iperf3测速

Iperf3

iperf3是一个网络速度测试工具，支持IPv4与IPv6，支持TCP、UDP、SCTP传输协议，是个简单又实用的小工具。

Iperf3安装

- 在openwrt中使用opkg安装 Iperf3

```
root@OpenWrt:~# opkg install iperf
Installing iperf (2.0.13-3) to root...
Downloading https://downloads.openwrt.org/snapshots/packages/mipsel_24kc/openwrt-24.03.0/packages/iperf_2.0.13-3_mipsel_24kc.ipk
Installing libstdc++6 (8.4.0-3) to root...
Downloading https://downloads.openwrt.org/snapshots/targets/mipsel_24kc/openwrt-24.03.0/targets/mipsel_24kc/libstdc++6_8.4.0-3_mipsel_24kc.ipk
Configuring libstdc++6.
Configuring iperf.
root@OpenWrt:~# iperf -v
iperf version 2.0.13 (21 Jan 2019) pthreads
```

- 在手机上安装可以使用Iperf3的软件
(例如 Network Tools)



监听

- 在 openwrt 上使用 iperf3 -s 开启监听

```
root@OpenWrt: # iperf3 -s
-----
Server listening on 5201 (test #1)
-----
Accepted connection from 192.168.1.189, port 41212
[ 5] local 192.168.1.1 port 5201 connected to 192.168.1.189
[ ID] Interval           Transfer     Bitrate
[ 5] 0.00-1.00    sec   37.7 MBytes  316 Mbits/sec
[ 5] 1.00-2.00    sec   41.0 MBytes  344 Mbits/sec
[ 5] 2.00-3.00    sec   42.7 MBytes  359 Mbits/sec
[ 5] 3.00-4.00    sec   39.5 MBytes  331 Mbits/sec
[ 5] 4.00-5.00    sec   29.1 MBytes  244 Mbits/sec
[ 5] 5.00-6.00    sec   33.0 MBytes  277 Mbits/sec
[ 5] 6.00-7.00    sec   34.5 MBytes  289 Mbits/sec
[ 5] 7.00-8.00    sec   24.6 MBytes  206 Mbits/sec
[ 5] 8.00-9.00    sec   42.5 MBytes  355 Mbits/sec
[ 5] 9.00-10.00   sec   38.6 MBytes  325 Mbits/sec
[ 5] 10.00-10.00  sec    177 KBytes  402 Mbits/sec
-----
[ ID] Interval           Transfer     Bitrate
[ 5] 0.00-10.00   sec   363 MBytes  305 Mbits/sec
-----
Server listening on 5201 (test #2)
-----
```

测速

- 在手机上使用 iperf3 -c 192.168.1.1 进行测速



192.168.1.1:5201 (TCP)

Interval	Transfer	Bandwidth
0.00-1.00 sec	39.0 MBytes	327 Mbits/sec
1.00-2.00 sec	41.6 MBytes	349 Mbits/sec
2.00-3.00 sec	42.7 MBytes	359 Mbits/sec
3.00-4.00 sec	39.9 MBytes	335 Mbits/sec
4.00-5.00 sec	29.4 MBytes	247 Mbits/sec
5.00-6.00 sec	32.6 MBytes	274 Mbits/sec
6.00-7.00 sec	34.4 MBytes	289 Mbits/sec
7.00-8.00 sec	25.1 MBytes	211 Mbits/sec
8.00-9.00 sec	41.6 MBytes	349 Mbits/sec
9.00-10.00 sec	39.2 MBytes	329 Mbits/sec



修改luci界面

LuCI 基础

Controller 位于： /usr/lib/lua/luci/controller/ 下——定义模块的入口

Model 位于： /usr/lib/lua/luci/model/cbi/ 下——配置模块实际的代码

第一步：定义模块入口：

控制器名/路径：

不带路径的控制器名默认存在于/usr/lib/lua/luci/controller/下，否则以controller/为根目录

entry表示添加一个新的模块入口，官方给出了entry的定义，其中后两项都是可以为空

```
entry(path, target, title=nil, order=nil)
```

path：

如果这样写{"click", "here", "now"}，那么就可以在浏览器里访问“<http://192.168.x.1/cgi-bin/>

[下方式编写](#) {"admin", "一级菜单名", "菜单项名"}，系统会自动在对应的菜单中生成菜单项

以写为“network”。

target：

调用目标分为三种，分别是执行指定方法Action、访问指定页面Views以及调用CBI Module

第一种可以直接调用指定的函数，比如点击菜单项就直接重启路由器等等，比如写为重启路由器函数就可以调用了。

第二种可以访问指定的页面，比如写为template(“myapp/mymodule”)就可以调用/usr/lib/lua/luci/view/

而如果要编写配置页面，那么使用第三种方法无非是最方便的，比如写为cbi(“myapp/myconfig”)就可以调用myapp/myconfig.lua文件了。

title和order:

```
module("luci.controller.LuoYeLuCI", package.seeall)
```

```
function index()
```

```
entry({"admin", "network", "LuoYeconfig"}, cbi("LuoYeCBI"), _("LuoYeTest"), 100)
```

```
end
```

第二步：配置CBI Module

首先要需要映射与存储文件的关系

```
m = Map("配置文件文件名", "配置页面标题", "配置页面说明")
```

第一个参数即为配置文件存储的文件名，不包含路径。

第二与第三个参数则是用在来页面上显示的

接下来需要创建与配置文件中对应的Section

Section分为两种，NamedSection和TypedSection，前者根据配置文件中的Section名，后者根据配置文件的Section类型。

创建配置文件

文件需要存储在/etc/config（如果配置文件不存在的话，访问配置页面将会报错）

内容格式如下：

```
config login
option username "
option password "
option ifname 'eth0'
option domain "
```

LuCI 页面修改

简单的文件配置，路由上路径主要是/usr/lib/lua/luci/下子目录：/controller/、/model/、/view/。要编译自定义LuCI页面的固件，修改如下OpenWRT源码结构路径内的LuCI文件。

需求分析

在使用OpenWrt路由器的过程中，经常需要根据需要改改配置文件然后重新启动服务。后使用/etc/init.d/xxxx restart 来重启服务，次数多了就会觉得很繁琐，所以我们考虑

过程

```
config arguments
```

```
module("luci.controller.wy777", package.seeall)
function index()
    if not nixio.fs.access("/etc/config/wyzzz") then
        return
    end
    entry({"admin", "services", "Settings"}, cbi("wyzzz"), _("Settings"))
end
```

```

local fs = require "nixio.fs"
local sys = require "luci.sys"
local m,s,o,i,a
m=Mop("wyzzz",translate("Settings"),translate("designed by wyzzz"))
s=m:section(TypedSection,"arguments","")
s.addremove = false
s.anonymous = true
s:tab("config", translate("Firmware version"),translate(""))
conf = s:taboption("config", Value, "editconf", nil, translate("# or : are regarded as comments;delay"))
conf.template = "cbi/tvalue"
conf.rows = 20
conf.wrap = "off"
function conf.cfgvalue(self, section)
    return fs.readfile("/etc/openwrt_release") or ""
end
function conf.write(self, section, value)
    if value then
        value = value:gsub("\r\n?", "\n")
        fs.writefile("/tmp/openwrt_release", value)
        if (luci.sys.call("cmp -s /tmp/openwrt_release /etc/openwrt_release") == 1) then
            fs.writefile("/etc/openwrt_release", value)
        end
        fs.remove("/tmp/openwrt_release")
    end
end
end

if nixio.fs.access("/etc/config/network") then
s:tab("config2", translate("network"),translate("set up /etc/config/network"))
conf = s:taboption("config2", Value, "editconf2", nil, translate("# or : are regarded as comments;delay"))
conf.template = "cbi/tvalue"
conf.rows = 20
conf.wrap = "off"
function conf.cfgvalue(self, section)
    return fs.readfile("/etc/config/network") or ""
end
function conf.write(self, section, value)
    if value then
        value = value:gsub("\r\n?", "\n")
        fs.writefile("/tmp/network", value)
        if (luci.sys.call("cmp -s /tmp/network /etc/config/network") == 1) then
            fs.writefile("/etc/config/network", value)
        end
        fs.remove("/tmp/network")
    end
end
end

```

```
/usr/lib/lua/luci/dispatcher.lua:1345: module 'luci.cbi' not
    no field package.preload['luci.cbi']
    no file './luci/cbi.lua'
    no file '/usr/share/lua/luci/cbi.lua'
    no file '/usr/share/lua/luci/cbi/init.lua'
    no file '/usr/lib/lua/luci/cbi.lua'
    no file '/usr/lib/lua/luci/cbi/init.lua'
    no file './luci/cbi.so'
    no file '/usr/lib/lua/luci/cbi.so'
    no file '/usr/lib/lua/loadall.so'
    no file './luci.so'
    no file '/usr/lib/lua/luci.so'
    no file '/usr/lib/lua/loadall.so'
stack traceback:
  [C]: in function 'require'
  /usr/lib/lua/luci/dispatcher.lua:1345: in function '
  /usr/lib/lua/luci/dispatcher.lua:1022: in function '
  /usr/lib/lua/luci/dispatcher.lua:479: in function </
```

```
1 | opkg update
2 | opkg install luci luci-base luci-compat
```

成果

- 修改network相关配置

Settings

designed by wyzzz

Firmware version

network

Wireless


set up /etc/config/network

```
config interface 'loopback'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'
    option device 'lo'

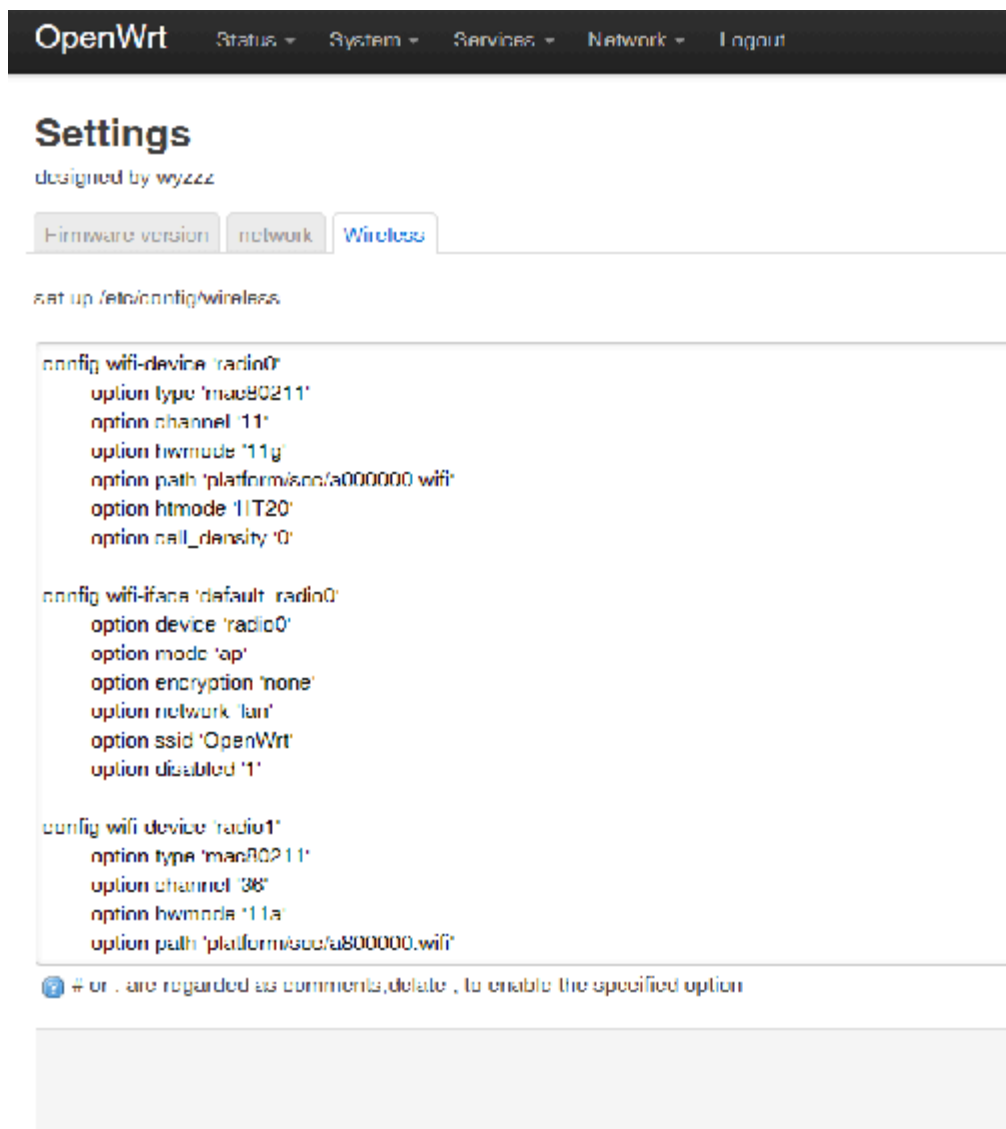
config global 'globals'
    option ula_prefix 'fd03:89b9:6a9d::/48'

config interface 'lan'
    option proto 'static'
    option ipaddr '192.168.2.1'
    option netmask '255.255.255.0'
    option ipassign '60'
    option device 'eth1'
    option type 'bridge'

config route
    option interface 'lan'
    option target '192.168.1.228'
    option gateway '192.168.1.1'
```

 # or : are regarded as comments; delete ; to enable the specified option

- 修改wireless相关配置



实验分析

三、实验总结