Quick-Start Guide - OpenWrt Networking on Raspberry Pi with Qualcomm QMI/RmNet

Simon Mullenger

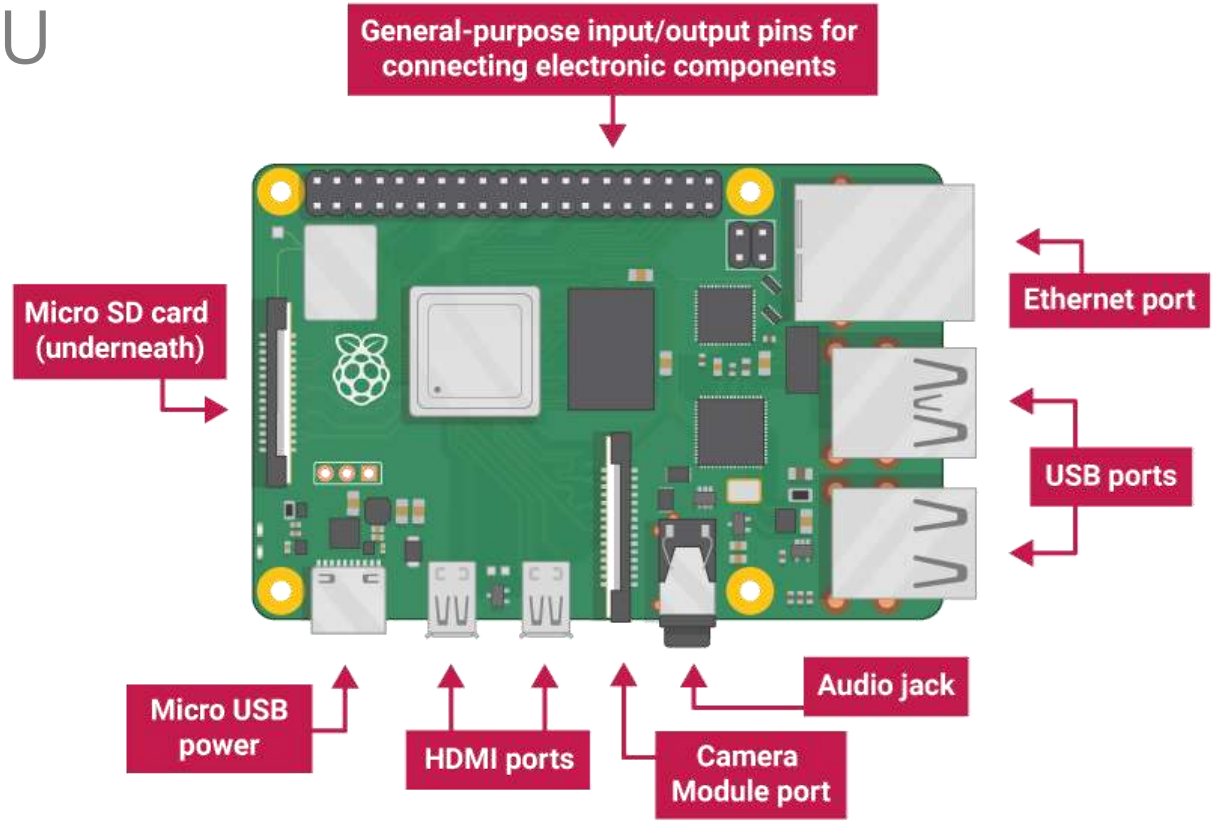
What we are talking about today

- Meet Raspberry Pi
- Configure a Raspberry Pi* for OpenWrt
  - SD card and package preparation
- Configure Qualcomm MSM Interface (QMI/RmNet) networking
  - /etc/config/network
  - /etc/config/firewall
  - /etc/qmi-network.conf
- Connecting to module over USB via /dev/cdc-wdm0
  - Using Qualcomm 4G (PLS83 or PLS63) or 5G (MV31-W / MV32-W) modules
- OpenWrt - LuCI Web Interface essentials

* - R-Pi2 and R-Pi3 have a USB 2.0 internal throughput limit
Raspberry Pi 4 uses USB 3.0 internally thus 5G speeds possible

Raspberry Pi – a single-board computer

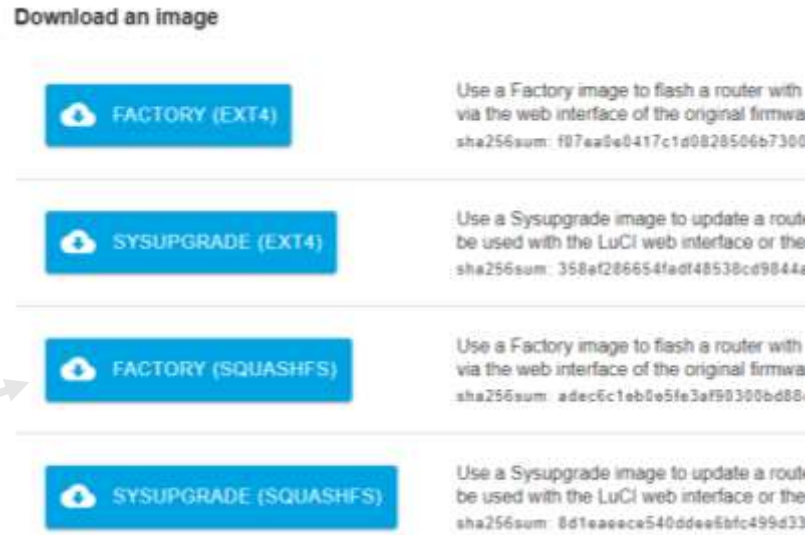
- Nice introduction article here: Getting started with Raspberry Pi
- 1GB RAM, 1GHz ARM Cortex CPU
- Broadcom Wi-Fi/WLAN
- USB 3.0* / 2.0 ports
- SD card slot
- Gigabit Ethernet* port
- Audio jack, HDMI port
- Micro USB power connector
- GPIO ports / HATs & Shields



* - only use Raspberry Pi 4 and above for high speed (>120Mb/s) testing

OpenWrt 22.03.6 – an embedded Linux for routers

- OpenWrt is a small embedded Linux - designed specifically for Routers
- OpenWrt Selector: https://firmware-selector.openwrt.org/
- Type “rasp”, choose model, and the FACTORY (SQUASHFS) image .gz file



- Install OpenWrt by writing .img file to a micro SD Card, for Raspberry Pi

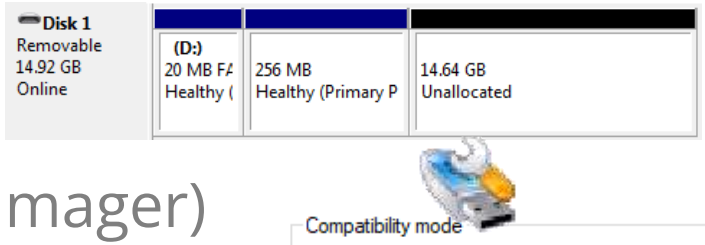
Configure a Raspberry Pi for OpenWrt (1)

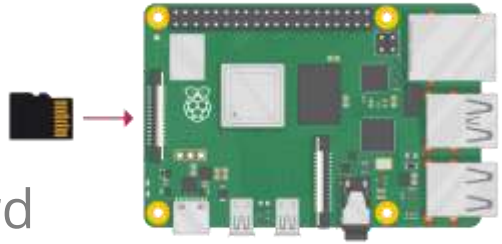
- OpenWrt software components are (1) an image and (2) some packages:
- 1. Download a factory image of OpenWrt firmware for Raspberry Pi
  - squashfs is small (75MB) and has the (really useful) “Perform reset” to factory defaults option
  - ext4 images are larger (200MB) and do not have the reset feature - don't use the ext4 image
- 2. Additional packages* are needed; some drivers and LuCI add-ons
  - luci, luci-proto-qmi, kmod-nls-base, kmod-usb-core, kmod-usb2, kmod-usb3
  - kmod-usb-acm, kmod-mii, kmod-usb-net, kmod-usb-ehci, kmod-usb-xhci-hcd
  - kmod-usb-wdm, kmod-usb-net-qmi-wwan, kmod-usb-serial, kmod-usb-serial-option, kmod-usb-serial-wwan
  - libc, libattr, libffi, zlib, glib2, libqmi, qmi-utils, wwan, uqmi
- SD Card cleaner (Windows use SD-Card-Formatter-5.0.2-Setup)
- SD Card image writer (Windows use win32diskimager-1.0.0-install)
- Tool to copy the packages to Raspberry Pi (Windows use WinSCP-6.1)

* - ZIP file contains all required additional packages for this project

Configure a Raspberry Pi for OpenWrt (2)

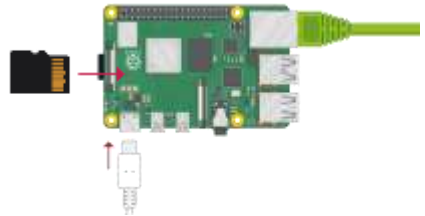
- Prepare a micro SD card for the Raspberry Pi to boot from (2GB or larger)
- Thoroughly clean the first 5% of the SD Card using “SD Card Formatter”
  - Use “Overwrite format” do not just use “Quick format”
- Unpack OpenWrt .img image file from .gz archive
- Write image to the SD Card (on Windows use win32diskimager)
  - Windows 10 run this tool in “Windows 7 compatibility mode”
  - Do a graceful software Eject of the SD card; don’t just “pull it out”
- Insert SD Card into Raspberry Pi and apply power
- Check coloured LEDs – they can alert to problems
  - ACT is the green LED - blinks as Raspberry Pi boots / reads SD Card
  - PWR is the red LED – blinking is BAD, indicating under-voltage conditions*





* - Ensure power supply is 5.1 Volts @ >2.5 Amps, else **Red LED** will not be consistently ON

Configure a Raspberry Pi for OpenWrt (3)

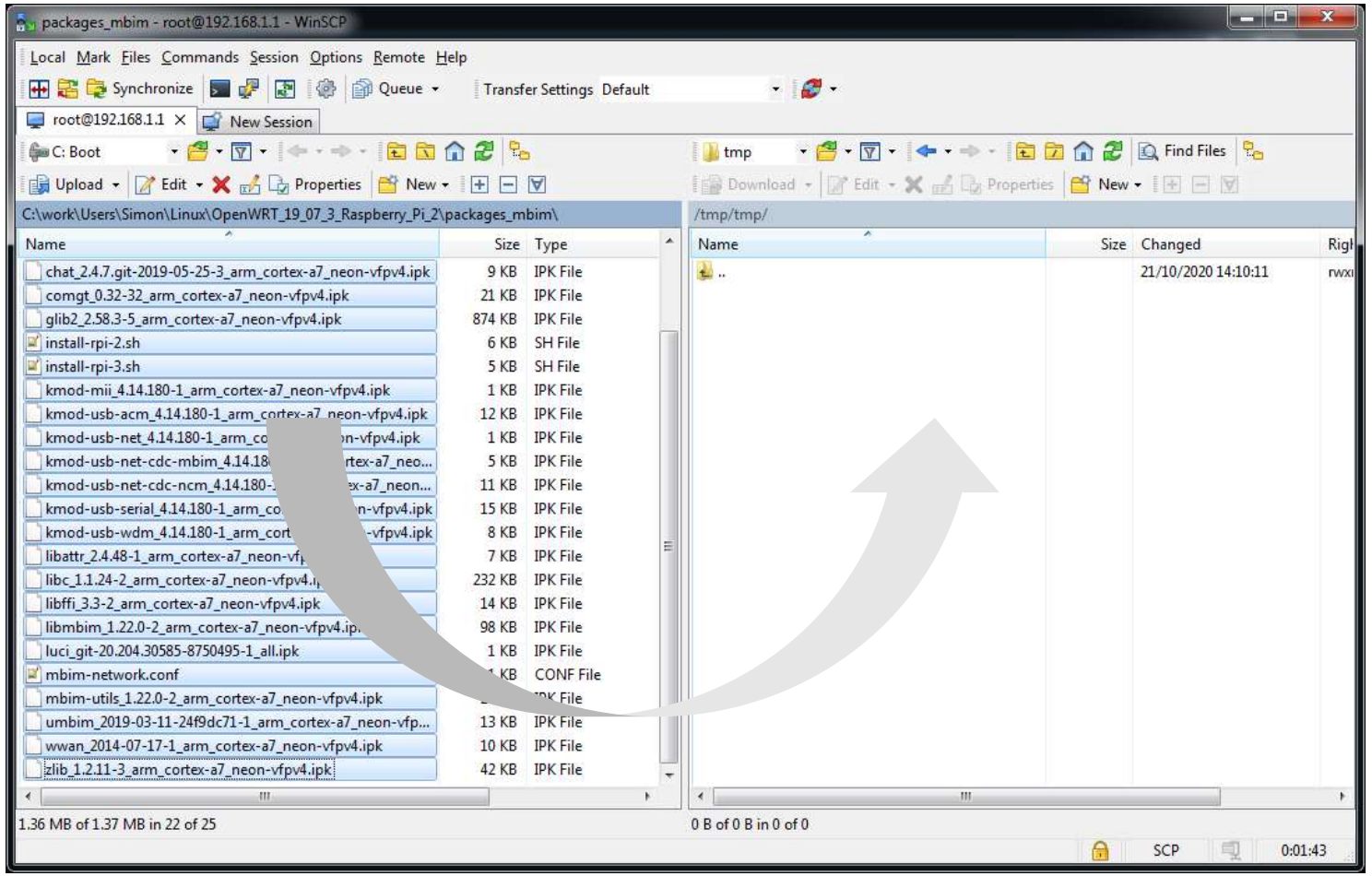


- Connect to OpenWrt router
  - Connect your computer to the Raspberry Pi Ethernet port
  - The default configuration is that the router will be a DHCP server on 192.168.1.1
  - You should receive an IP number, from the router, but will have no Internet access
  - Check you can reliably **`ping 192.168.1.1`** from your computer

- Copy additional packages onto OpenWrt router /tmp/tmp
  - Install WinSCP choosing the “two window” option
  - Use WinSCP to connect to the router using the File Protocol “SCP” (not SFTP)
  - IP address of router is Hostname “192.168.1.1” and Port number is 22
  - User name is “root” and just leave the Password field blank
  - Login, ignore any “Warning: Potential Security Breach” messages; say “Update”

Configure a Raspberry Pi for OpenWrt (4)

- Copy and paste all **`.conf`**, **`.ipk`**, **`.sh`** files to R-Pi router directory **`/tmp/tmp`**



Configure OpenWrt QMI networking (1)

- Using a Secure Shell (ssh) capable terminal, connect and Login to the router
  - IP address is 192.168.1.1 port is 22
  - User is “root”, and no password is currently needed

```
BusyBox v1.35.0 (2023-11-25 18:18:57 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 22.03.6, r20265-f85a79bcb4
 -----------------------------------------------------
=== WARNING! =====================================
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
--------------------------------------------------
root@OpenWrt:~#
```

Configure OpenWrt QMI networking (2)

* - packages shown here are for Raspberry Pi 3B+
See the file "**packages_qmi.zip**"

- List the additional packages*, previously copied to the router

```
root@OpenWrt:~# cd /tmp/tmp
root@OpenWrt:/tmp/tmp# ls -l
-rw-r--r--  1 root  root  1202302 Mar 13  2023 glib2_2.70.5-4_aarch64_cortex-a53.ipk
-rwxrwxrwx  1 root  root     5429 Apr 13  2023 install-rpi-3.sh
-rw-r--r--  1 root  root      957 Mar 13  2023 kmod-mii_5.10.161-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root    12791 Mar 13  2023 kmod-usb-acm_5.10.161-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root     8896 Mar 13  2023 kmod-usb-net-qmi-wwan_5.10.161-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root      954 Mar 13  2023 kmod-usb-net_5.10.161-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root    14873 Mar 13  2023 kmod-usb-serial_5.10.161-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root    10077 Mar 13  2023 kmod-usb-wdm_5.10.161-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root     7718 Mar 13  2023 libattr_2.5.1-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root    17265 Mar 13  2023 libffi_3.4.2-2_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root   126118 Mar 13  2023 libmbim_1.26.4-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root    82674 Mar 13  2023 libpcre_8.45-3_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root   526546 Mar 13  2023 libqmi_1.30.8-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root    13484 Mar 13  2023 libqrtr-glib_1.2.2-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root     2362 Mar 13  2023 luci-proto-qmi_git-21.231.25157-5ff3ef7_all.ipk
-rw-r--r--  1 root  root       56 Mar  1  2023 qmi-network.conf
-rw-r--r--  1 root  root   123346 Mar 13  2023 qmi-utils_1.30.8-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root      255 Apr 13  2023 rc.local
-rwxrwxrwx  1 root  root     4162 Apr 13  2023 reconnect.sh
-rw-r--r--  1 root  root    43314 Mar 13  2023 uqmi_2022-05-04-56cb2d40-1_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root     9822 Mar 13  2023 wwan_2019-04-29-5_aarch64_cortex-a53.ipk
-rw-r--r--  1 root  root    44020 Mar 13  2023 zlib_1.2.11-6_aarch64_cortex-a53.ipk
```

Configure OpenWrt QMI networking (3)

- Install the additional packages, previously copied to the router

```
root@OpenWrt:~# cd /tmp/tmp
root@OpenWrt:/tmp/tmp# chmod 777 *.sh
root@OpenWrt:/tmp/tmp# ./install-rpi-3.sh
```

- The script above, will install the required packages
- It then continues setting up the Raspberry Pi
  - The script performs each of the operations described in the following slides
  - The Raspberry Pi will reboot, with a new IP address, once configured
  - Read on for a detailed description of the rest of the set-up
- Pay attention to the APN defined in file “qmi-network.conf”
  - `+CEER: Requested service option not subscribed` => wrong APN defined

Configure cellular module – USB Mode Switch

- You have now installed all the drivers that you need
- However, the cellular module might also need a “USB mode switch”
  - After the mode switching commands below, the module will reboot
- MV31 / MV32 ship in MBIM mode – switch to QMI mode:

```
root@OpenWrt:~# echo -ne 'AT+USBSWITCH=00B7\r' > /dev/ttyUSB0
root@OpenWrt:~# echo -ne 'AT^SETCONFIG=1\r' > /dev/ttyUSB0
root@OpenWrt:~# echo -ne 'AT+SETCONFIG=1\r' > /dev/ttyUSB0
```

mv31 rel 0.0.5.8
mv31 rel 1.0.0.9 onwards
mv32 rel 1

- PLS83 / PLS63 ship in CDC-ECM mode – switch to QMI mode:

```
root@OpenWrt:~# echo -ne 'AT^SSRVSET="actSrvSet",11\r' > /dev/ttyACM0
root@OpenWrt:~# echo -ne 'AT+CFUN=1,1\r' > /dev/ttyACM0
```

- FN990 – ships in QMI mode; SIM CCIN detection:

```
root@OpenWrt:~# echo -ne 'AT#SIMINCFG=1,1\r' > /dev/ttyUSB0
root@OpenWrt:~# echo -ne 'AT#REBOOT\r' > /dev/ttyUSB0
```

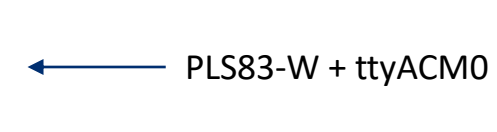
Telit FN990 SIM detection:
AT#SIMDET=0
AT#HSEN=1,0
AT#SIMINCFG=1,1 **<- SIMIN active High**
AT#REBOOT

* - QMI available only in recent Qualcomm based modules

Configure OpenWrt QMI networking (4)

- Plug in the USB cable to one of the USB ports on the Raspberry Pi
- Issue any needed “USB mode switch” commands, from the previous page
- Ensure the module appears as a device called ‘**cdc-wdm0**’

```
root@OpenWrt:~# ls -l /dev/cdc* /dev/ttyACM* /dev/ttyUSB*
ls: /dev/ttyUSB*: No such file or directory
crw-------    1 root     root      180, 176 Jan  3 01:40 /dev/cdc-wdm0
crw-rw----    1 root     dialout   166,   0 Jan  3 01:42 /dev/ttyACM0
crw-rw----    1 root     dialout   166,   1 Jan  3 01:40 /dev/ttyACM1
crw-rw----    1 root     dialout   166,   2 Jan  3 01:40 /dev/ttyACM2
crw-rw----    1 root     dialout   166,   3 Jan  3 01:40 /dev/ttyACM3
root@OpenWrt:~#
```



Configure OpenWrt QMI networking (5)

- Add a new ‘interface’ to the OpenWrt network file
  - Our module is referenced via this interface; it will be called “qmi”
  - Naming the interface “qmi” requires an extra firewall zone ”qmi” rule
  - Backup the default network file, then issue UCI configuration commands:

```
root@OpenWrt:/tmp/tmp# cd /etc/config
root@OpenWrt:/etc/config# cp network network.orig

root@OpenWrt:/etc/config# uci set network.qmi=interface
root@OpenWrt:/etc/config# uci set network.qmi.auth='none'
root@OpenWrt:/etc/config# uci set network.qmi.device='wwan0'
root@OpenWrt:/etc/config# uci set network.qmi.proto='dhcp'
root@OpenWrt:/etc/config# uci set network.qmi.disabled='0'
root@OpenWrt:/etc/config# uci set network.qmi.delegate='0'
root@OpenWrt:/etc/config# uci set network.qmi.dns='8.8.8.8 1.1.1.1'
root@OpenWrt:/etc/config# uci set network.qmi.ipv6='0'
root@OpenWrt:/etc/config# uci set network.qmi.pdptype='ipv4'
root@OpenWrt:/etc/config# uci set network.qmi.peerdns='0'

root@OpenWrt:/etc/config# uci commit network
```

OpenWrt 21 renamed the parameter “ifname” to “device”

If your QMI module does not provide a dhcp server then set proto='**static**' and disabled='**1**'
and must manually grab IPv4 info:
see script /etc/reconnect.sh and
the worker script /etc/renew.sh

Configure OpenWrt QMI networking (6)

- Add our new ‘interface’ to the OpenWrt firewall and dhcp rules
- Backup the default firewall file, then issue UCI configuration commands:

```
root@OpenWrt:/etc/config# cp firewall firewall.orig
root@OpenWrt:/etc/config# uci set firewall.@defaults[0].flow_offloading='1'
root@OpenWrt:/etc/config# uci set firewall.@zone[1].network='wan wan6 qmi'
root@OpenWrt:/etc/config# uci commit firewall
```

- Backup the default dhcp file, then issue UCI configuration commands:

```
root@OpenWrt:/etc/config# cp dhcp dhcp.orig
root@OpenWrt:/etc/config# uci set dhcp.@dnsmasq[0].authoritative=1
root@OpenWrt:/etc/config# uci set dhcp.@dnsmasq[0].sequential_ip='1'
root@OpenWrt:/etc/config# uci set dhcp.@dnsmasq[0].dhcpv6='disabled'
root@OpenWrt:/etc/config# uci commit dhcp.@dnsmasq[0]

root@OpenWrt:/etc/config# uci set dhcp.lan.dhcpv6='disabled'
root@OpenWrt:/etc/config# uci set dhcp.lan.leasetime='12h'
root@OpenWrt:/etc/config# uci set dhcp.lan.ra='disabled'
root@OpenWrt:/etc/config# uci commit dhcp.lan
```

Configure OpenWrt QMI networking (7)

- Optionally, to add APN info, edit your own QMI configuration file

```
root@OpenWrt:/etc/config# cat > /tmp/tmp/qmi-network.conf
APN=mob.asm.net
APN_USER=
APN_PASS=
APN_AUTH=
PROXY=yes
^D
```

Ensure APN, User, Pass details for your SIM card are correct here. Ensure that this file is in Linux text format (no Windows ^M characters on the end of each line).

^D – is actually the Ctrl key and D key together

- This will be read by the "qmi-network" application later, so copy to /etc

```
root@OpenWrt:/etc/config# cd /etc
root@OpenWrt:/etc# cp /tmp/tmp/qmi-network.conf /etc
```

- Using the package kmod-usb-serial-option includes all new VID/PIDs:

```
root@OpenWrt:/etc# ls -l /dev/cdc* /dev/ttyACM* /dev/ttyUSB*
crw-------    1 root     root      180, 176 Nov 25 18:22 /dev/cdc-wdm0
crw-rw----    1 root     dialout   188,   0 Nov 25 18:22 /dev/ttyUSB0
crw-rw----    1 root     dialout   188,   1 Nov 25 18:22 /dev/ttyUSB1
crw-rw----    1 root     dialout   188,   2 Nov 25 18:22 /dev/ttyUSB2
```

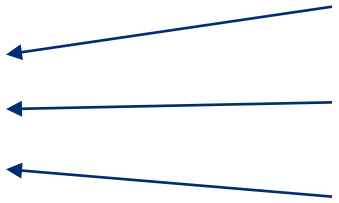
```
'1e2d 00b3' (MV31 MBIM mode)
'1e2d 00b7' (MV31 QMI mode)
'1e2d 00f1' (MV32 MBIM mode)
'1e2d 00f3' (MV32 QMI mode)
'1bc7 1070' (FN990 QMI mode)
```

Configure OpenWrt QMI networking (8)

- Add our own ‘scripts’ to the OpenWrt start-up sequence and runtime

```
root@OpenWrt:/etc/config# cd /etc

root@OpenWrt:/etc# cp /tmp/tmp/rc.local /etc
root@OpenWrt:/etc# cp /tmp/tmp/*.sh /etc
root@OpenWrt:/etc# chmod 777 /etc/reconnect.sh
root@OpenWrt:/etc# chmod 777 /etc/renew.sh
```



Here we modify the OpenWrt user start-up script

It additionally starts the “House Keeping” script that can monitor the connection and light LEDs etc.

Our scripts must be “executable”

- The most important script we add is /etc/reconnect.sh
  - This script sends QMI/RmNet commands to the module using **`qmicli`**
  - The most important is **`qmi-network /dev/cdc-wdm0 start`**
  - You could of course do all this by hand; use “**`ps`**” and “**`kill`**” to find/stop the scripts
- Some QMI enabled modules have no dhcp server in them
  - It is possible to manually set-up the IP address and gateway; see **`/etc/renew.sh`**

Configure OpenWrt QMI networking (9)

- We can lower CPU usage and improve Firewall performance
- Current OpenWrt versions allow significant through-put speed optimisation

- Enable Packet Steering across all CPU cores

```
root@OpenWrt:/etc/config# uci set network.globals.packet_steering='1'
root@OpenWrt:/etc/config# uci commit network
```

- Allow firewall to use Software Offloading

```
root@OpenWrt:/etc/config# uci set firewall.@defaults[0].flow_offloading='1'
root@OpenWrt:/etc/config# uci commit firewall
```

OpenWrt QMI networking (1) - reboot

- Reboot the OpenWrt router to start the new configuration
- When it reboots, log back in and start using qmicli commands

```
root@OpenWrt:/etc# reboot

BusyBox v1.35.0 (2023-11-25 18:18:57 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 22.03.6, r20265-f85a79bcb4
 -----------------------------------------------------
=== WARNING! =====================================
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
--------------------------------------------------
root@OpenWrt:~#
```

OpenWrt QMI networking (2) – format “raw-ip”

```
root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --get-expected-data-format
802-3

root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --set-expected-data-format=raw-ip
error: cannot set expected data format: Failed to write to sysfs file '/sys/class/net/wwan0/qmi/raw_ip': Resource busy

root@OpenWrt:~# uci set network.qmi.disabled='1'
root@OpenWrt:~# service network reload

root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --set-expected-data-format=raw-ip
[/dev/cdc-wdm0] expected data format set to: raw-ip

root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --get-expected-data-format
raw-ip


root@OpenWrt:~# uci set network.qmi.disabled='0'
root@OpenWrt:~# service network reload
```

Use qmicli to talk to the cellular module as /dev/cdc-wdm0 (do not need AT Commands)

Set expected data format to “raw-ip”

This will fail if wwan0 is enabled
So we need to disable wwan0

Run the qmicli command(s) again

Then re-enable wwan0

OpenWrt QMI networking (3) – get ready

```
root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --dms-get-ids
[/dev/cdc-wdm0] Device IDs retrieved:
            IMEI: '35923271000xxxx'
         IMEI SV: '15'
```

← Device Management Service
-get IDs (IMEI)

```
root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --uim-get-slot-status
[/dev/cdc-wdm0] Successfully got slots status
    Logical slot: 1
            ICCID: 894420012365900xxxx
```

← User Identity Module
-get slot status (ICCIDs)

```
root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --dms-get-capabilities
[/dev/cdc-wdm0] Device capabilities retrieved:
        Max TX channel rate: '50000000'
        Max RX channel rate: '100000000'
              Data Service: 'non-simultaneous-cs-ps'
                       SIM: 'supported'
                  Networks: 'gsm, umts, lte, 5gnr'
```

← Device Management Service
-get capabilities (speeds / RaTs)

OpenWrt QMI networking (4) – registered

```
root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --nas-get-serving-system
[/dev/cdc-wdm0] Successfully got serving system:
        Registration state: 'registered'
        CS: 'attached'
        PS: 'attached'
        Selected network: '3gpp'
        Radio interfaces: '1' [0]: 'lte'
        Roaming status: 'off'
        Data service capabilities: '1' [0]: 'lte'
        Current PLMN: MCC: '234' MNC: '20' Description: 'SMARTY'
        Roaming indicators: '1' [0]: 'off' (lte)
        3GPP location area code: '65534'
        3GPP cell ID: '12968541'
        Detailed status: Status: 'available' Capability: 'cs-ps'
        Full operator code info: MCC: '234' MNC: '20' MNC with PCS digit: 'no'
```

Use qmicli to check registration state on the network

Network Access Service*
-get serving system info

* - output is redacted to fit slide

```
root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --nas-get-signal-info
[/dev/cdc-wdm0] Successfully got signal info
LTE:                                5G:
        RSSI: '-87 dBm'
        RSRQ: '-19 dB'              RSRP: 'n/a'
        RSRP: '-124 dBm'             SNR: 'n/a'
        SNR: '-6.6 dB'              RSRQ: 'n/a'
root@OpenWrt:~#
```

Network Access Service
-get signal info

OpenWrt QMI networking (5) – QMI connection

```
root@OpenWrt:~# qmi-network /dev/cdc-wdm0 status
Getting status with 'qmicli -d /dev/cdc-wdm0 --wds-get-packet-service-status  --device-open-proxy'...
Status: disconnected

root@OpenWrt:~# qmi-network /dev/cdc-wdm0 start
Loading profile at /etc/qmi-network.conf...
    APN: mob.asm.net
    APN user: unset
    APN password: unset
    qmi-proxy: yes
Checking data format with 'qmicli -d /dev/cdc-wdm0 --wda-get-data-format --device-open-proxy'...
Device link layer protocol retrieved: raw-ip
Getting expected data format with 'qmicli -d /dev/cdc-wdm0 --get-expected-data-format'...
Expected link layer protocol retrieved: raw-ip
Device and kernel link layer protocol match: raw-ip
Starting network with 'qmicli -d /dev/cdc-wdm0 --wds-start-network=apn='APN' --client-no-release-cid --device-open-
proxy'...
Saving state at /tmp/qmi-network-state-cdc-wdm0... (CID: 3)
Saving state at /tmp/qmi-network-state-cdc-wdm0... (PDH: 2244334000)
Network started successfully

root@OpenWrt:~# qmi-network /dev/cdc-wdm0 status
Loading previous state from /tmp/qmi-network-state-cdc-wdm0...
    Previous CID: 3
    Previous PDH: 2244334000
Getting status with 'qmicli -d /dev/cdc-wdm0 --wds-get-packet-service-status --client-cid=3 --client-no-release-cid --device-open-proxy'...
Status: connected
```

Use qmi-network to
-get Packet Service status

and Start QMI network connection
File `/etc/qmi-network.conf` is OK

Use qmi-network to
-get Packet Service status (again)

State: Loading previous state from /tmp/qmi-network-state-cdc-wdm0...

OpenWrt QMI networking (6) – DHCP and wwan0

```
root@OpenWrt:~# qmicli -d /dev/cdc-wdm0 --wds-get-current-settings
[/dev/cdc-wdm0] Current settings retrieved:
            IP Family: IPv4
         IPv4 address: 10.94.67.75
     IPv4 subnet mask: 255.255.255.248
IPv4 gateway address: 10.94.67.76
     IPv4 primary DNS: 188.31.250.128
   IPv4 secondary DNS: 188.31.250.129
                  MTU: 1380

root@OpenWrt:~# ip a
5: wwan0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN qlen 1000
    link/[65534]
    inet 10.94.67.75/29 brd 10.94.67.79 scope global wwan0

root@OpenWrt:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=115 time=43.433 ms
64 bytes from 8.8.8.8: seq=1 ttl=115 time=44.130 ms
64 bytes from 8.8.8.8: seq=2 ttl=115 time=44.957 ms
64 bytes from 8.8.8.8: seq=3 ttl=115 time=33.724 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 33.724/41.561/44.957 ms
```

Use qmicli to retrieve the IP network settings; shows IPv4 and IPv6

We should at least see values for IPv4

Use ip a to check IPv4 info has been successfully added to wwan0 via dhcp

Finally try a real world ping to 8.8.8.8 or 1.1.1.1 for example

^C – is actually the Ctrl key and C key together

Hint: if above did not work; read on…

OpenWrt QMI networking (7) – get DHCP

```
root@OpenWrt:~# udhcpc -i wwan0
udhcpc: started, v1.35.0
udhcpc: broadcasting discover
udhcpc: broadcasting select for 10.94.67.75, server 10.94.67.76
udhcpc: lease of 10.94.67.75 obtained from 10.94.67.76, lease time 7200
udhcpc: ip addr add 10.94.67.75/255.255.255.248 broadcast + dev wwan0
udhcpc: setting default routers: 10.94.67.76

root@OpenWrt:~# uci set network.qmi.ifname='wwan0'
root@OpenWrt:~# uci set network.qmi.disabled='0'
root@OpenWrt:~# uci set network.qmi.proto='dhcp'
root@OpenWrt:~# uci commit network
root@OpenWrt:~# service network restart

root@OpenWrt:~# ping 8.8.8.8 -I wwan0
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=115 time=165.632 ms
64 bytes from 8.8.8.8: seq=1 ttl=115 time=50.149 ms
64 bytes from 8.8.8.8: seq=2 ttl=115 time=63.083 ms
64 bytes from 8.8.8.8: seq=3 ttl=115 time=44.932 ms
64 bytes from 8.8.8.8: seq=4 ttl=115 time=40.459 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 40.459/72.851/165.632 ms
root@OpenWrt:~#
```

Get DHCP info

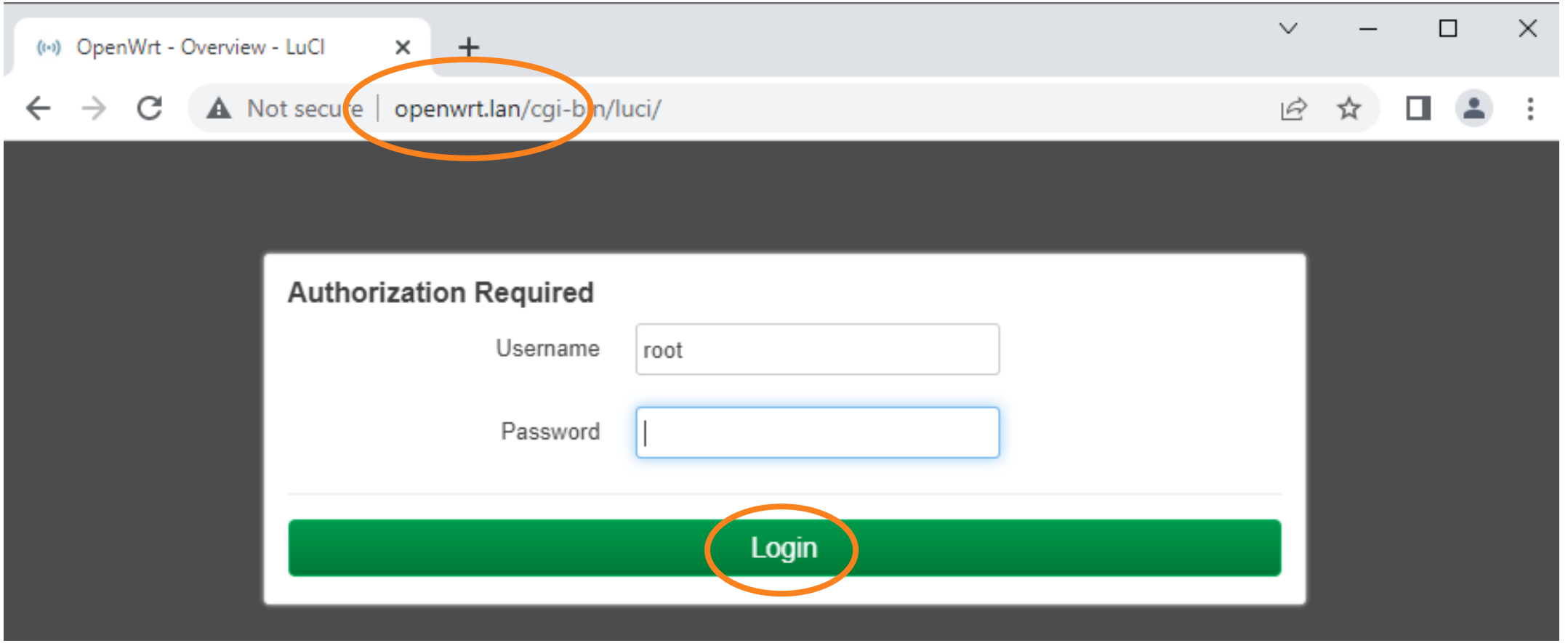
Enable the UCI interface ‘QMI’
Then commit and restart networking

Finally try a real world ping to
8.8.8.8 or 1.1.1.1 for example, again

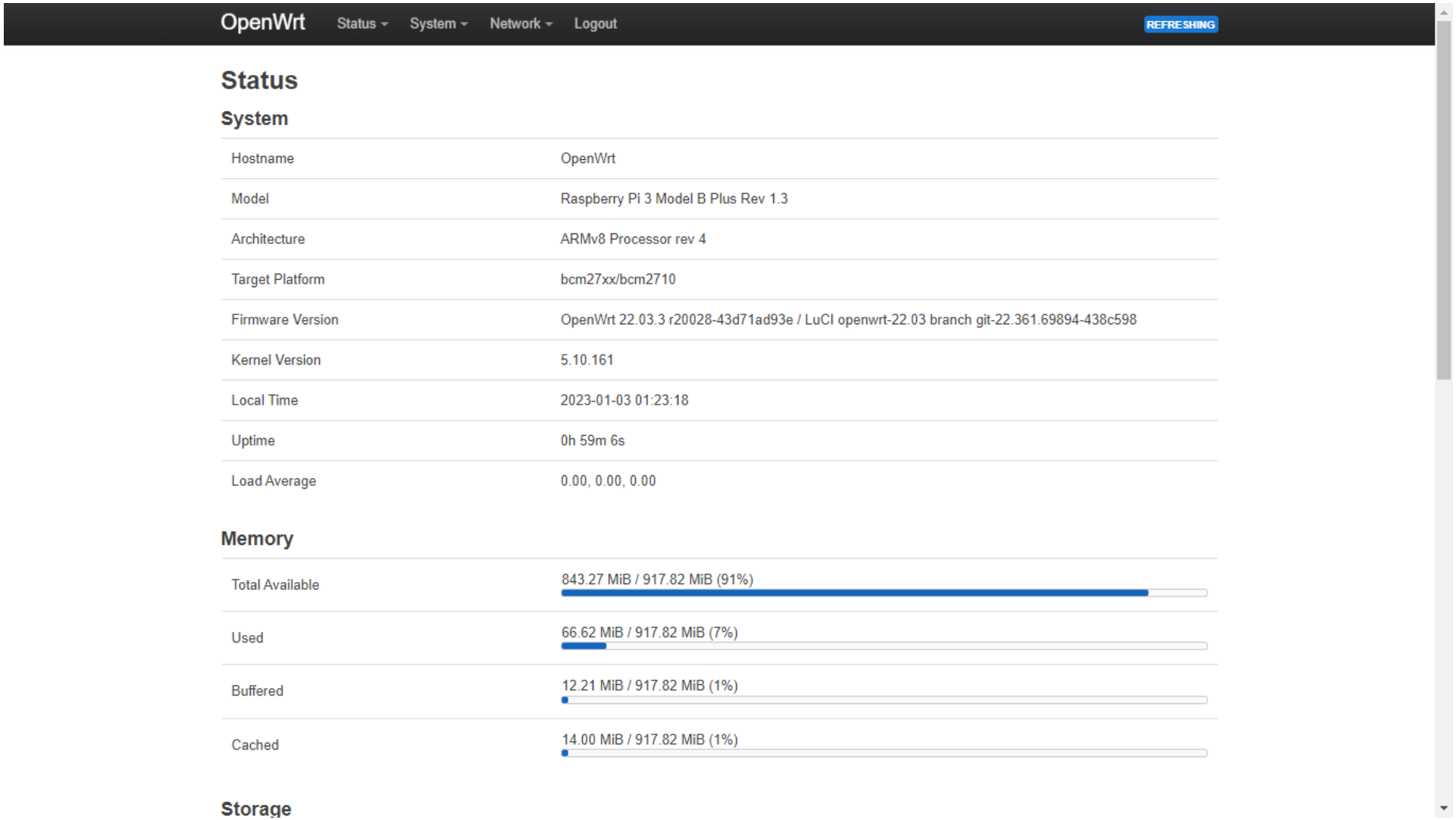
^C – is actually the Ctrl key and C key together

OpenWrt - LuCI Web Interface – management

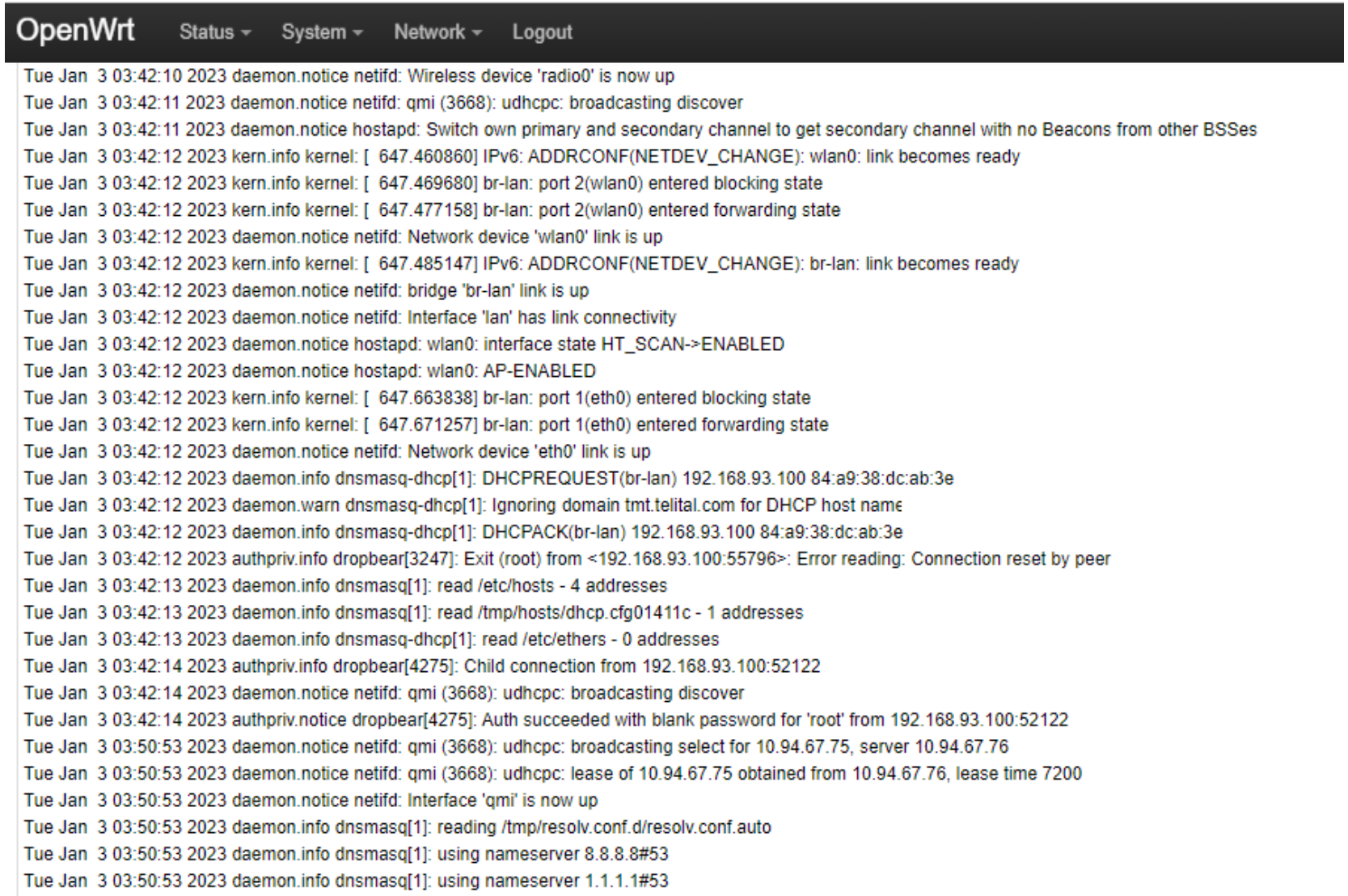
- Browse to http://192.168.1.1 or http://openwrt.lan



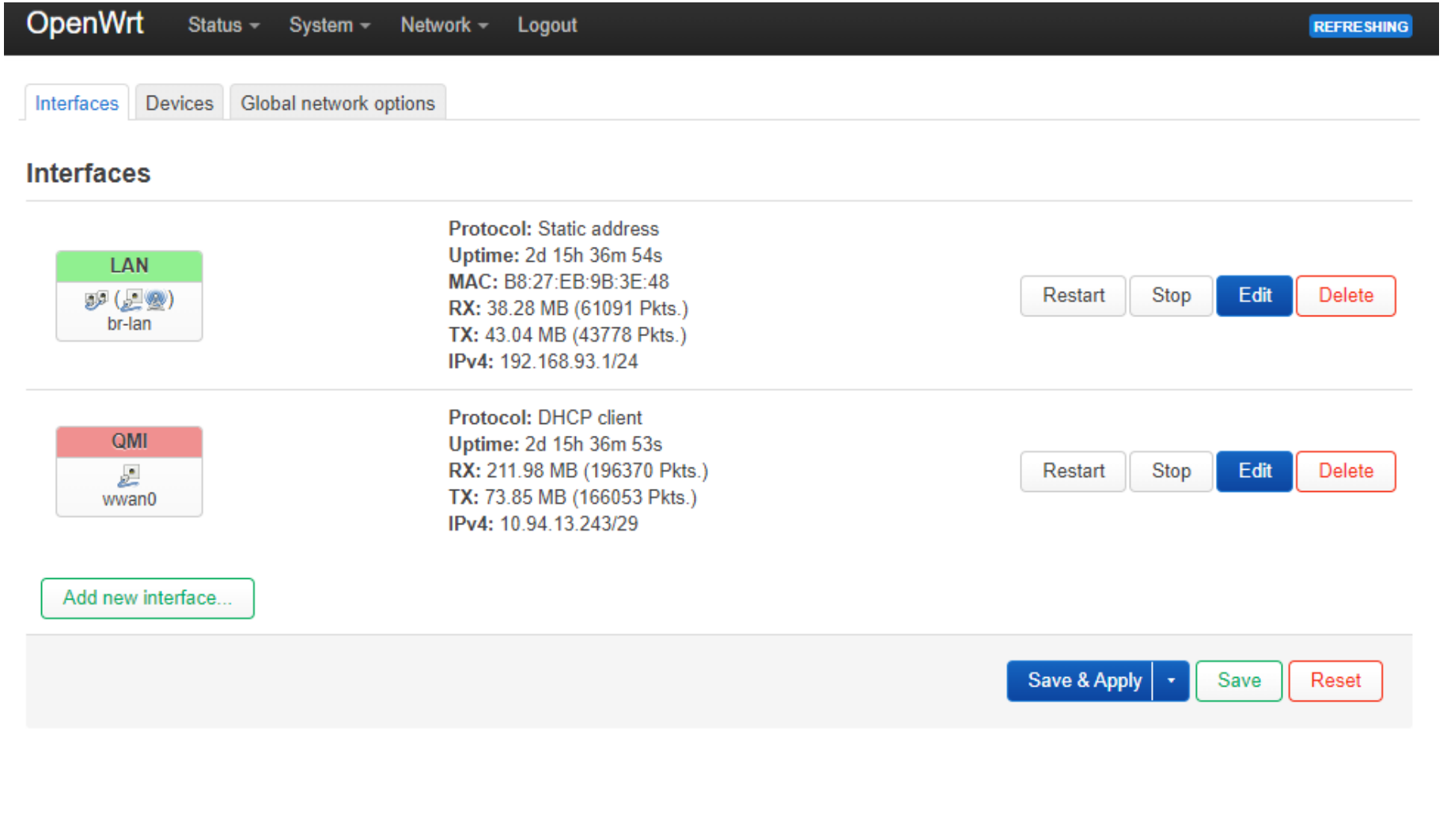
OpenWrt - LuCI Web Interface - Status -> Overview



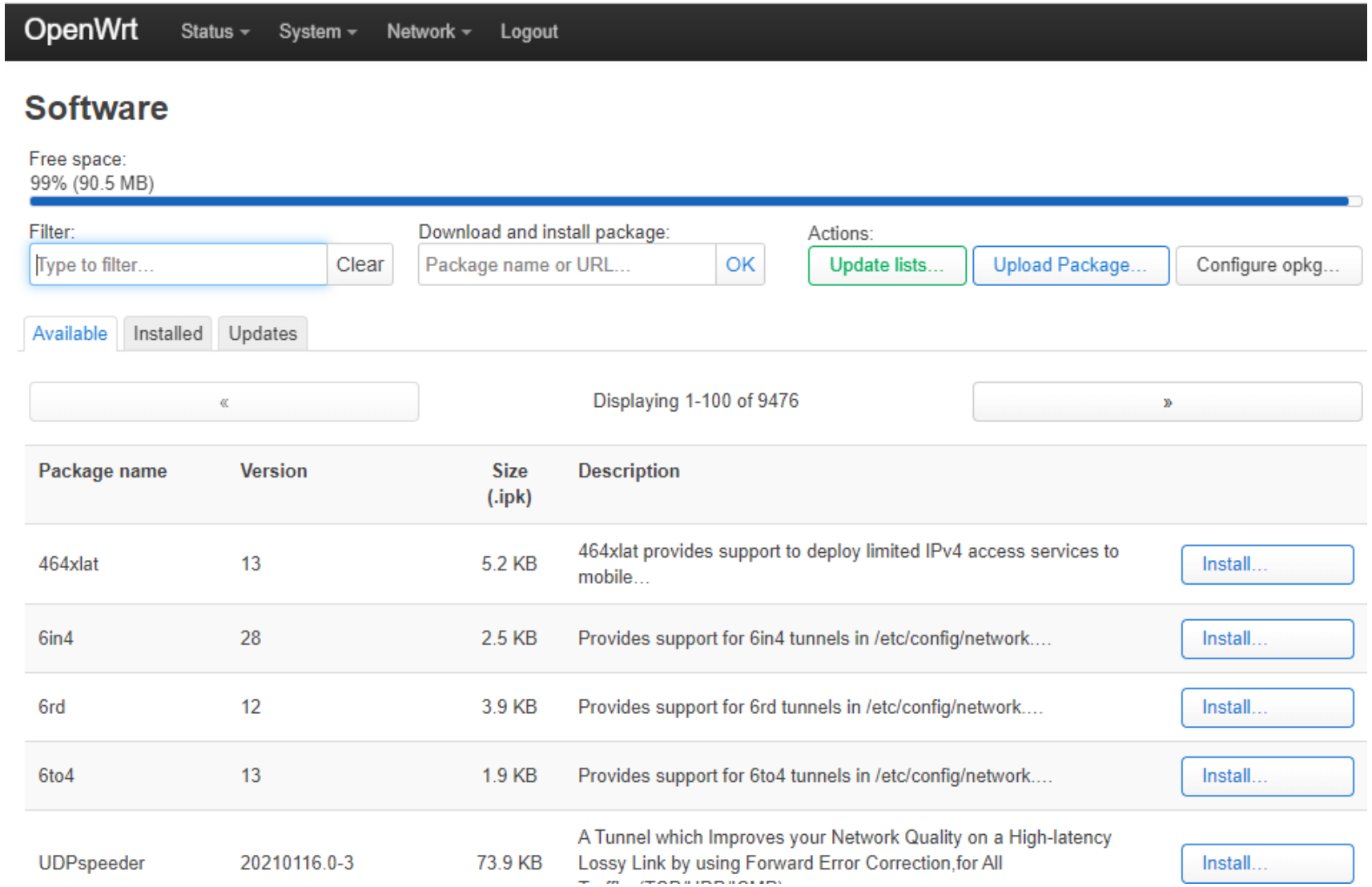
OpenWrt - LuCI Web Interface - System log



OpenWrt - LuCI - Interfaces LAN and QMI



OpenWrt - LuCI - System -> Software Installation

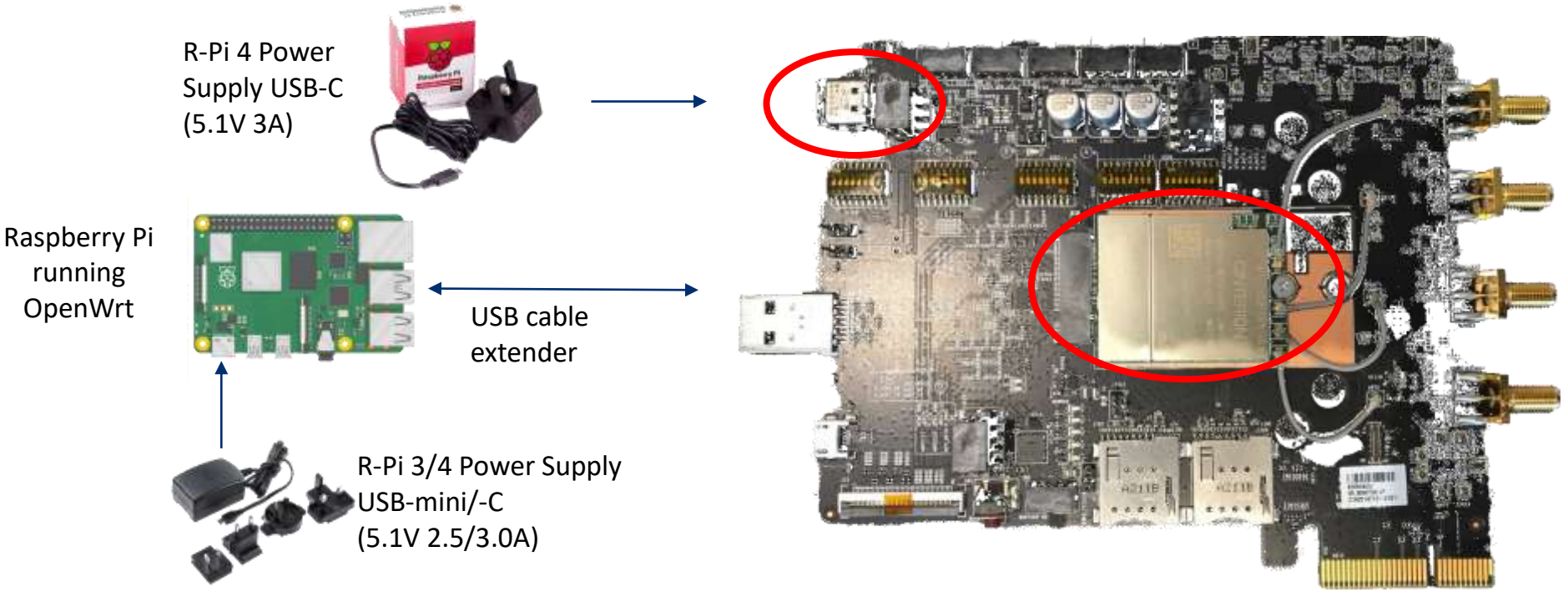


OpenWrt - LuCI - Software Add-ons of interest

- luci-app-ttyd and htop
  - ttyd provides a CLI shell from the Services menu in the LuCI web browser
  - htop shows real-time CPU usage
- picocom - access AT Command interface at /dev/ttyACM0 or USB0
- nano - a small and useful text editor for scripts and config files
- Open VPN, Open SSL and SoftEtherVPN5 server
  - Secure VPN Tunnelling and Bridging (network to network) over UDP/TCP
- sqm - Smart Queues on your router
- python-codecs python-db python python-pip
  - MQTT data exchange possible using ‘pip install paho-mqtt’

Raspberry Pi - 5G module - hardware dev kit

- USB connected 5G cellular module with 4G and 3G fall-back
  - The wireless module is connected via USB 3.0 / USB 2.0
  - Using the 5G module - on our development kit (sub-7GHz antennas)
  - Power supply is critical – both Raspberry Pi and 5G module own supplies



Configure cellular module – reset to factory

- You can reset your module to the factory shipping USB Mode
- Do this using the “USB mode switch” commands (again)
  - After the mode switching commands below, the module will reboot
- MV31/MV32 ship in MBIM mode

```
root@OpenWrt:~# echo -ne 'AT+USBSWITCH=00B7\r' > /dev/ttyUSB0
root@OpenWrt:~# echo -ne 'AT^SETCONFIG=1\r' > /dev/ttyUSB0
root@OpenWrt:~# echo -ne 'AT+SETCONFIG=1\r' > /dev/ttyUSB0
```

mv31 rel 0.0.5.8
mv31 rel 1.0.0.9 onwards
mv32 rel 1

- PLS83/PLS63 ship in CDC-ECM mode

```
root@OpenWrt:~# echo -ne 'AT^SSRVSET="actSrvSet",1\r' > /dev/ttyACM0
root@OpenWrt:~# echo -ne 'AT+CFUN=1,1\r' > /dev/ttyACM0
```

- ELS61/PLS62/PLS8/PLAS9 – do not support QMI

Thank you