OpenWRT and the Linksys WRT 1900 ACS Router

Linksys WRT 1900 ACS OpenWRT Firmware

Linksys WRT 1900 ACS OpenWRT Firmware

Default OpenWRT Admin Interface

http://192.168.1.1

Additional Package Installation

Install:

1. Huawei E3372 Hi-Link LTE Dongle: kmod-usb-net-cdc-ether usb-modeswitch
2. Mosquitto: mosquitto-nossl
3. Material Theme: luci-theme-material
4. Additional Packages: nano, git, htop

```
opkg update && opkg install kmod-usb-net-cdc-ether usb-modeswitch luci-theme-material mosq
```

Update Existing Packages

```
opkg list-upgradable | cut -f 1 -d ' ' | xargs opkg upgrade
```

Configure Huawei E3372 Hi-Link LTE Dongle

Using the Huawei E3372 Hi-Link LTE Dongle with OpenWRT

Plug in Huawei E3372 Hi-Link LTE Dongle to USB1

From Network -> Interfaces -> Add New Interface

- Name of new interface: Dongle
- Protocol: DHCP Client
- Cover the following interface: eth2

From Network -> Interfaces -> Edit Dongle -> Firewall Settings

- Add Dongle to **wan** group

Setting WAN Priority

Generally want to use Broadband first then fall back to LTE Dongle. This is done by setting a

gateway metric - low numbers = highest priority

Network -> Interfaces -> WAN -> Advanced Settings

- Use gateway metric: 10

Network -> Interfaces -> Dongle -> Advanced Settings

- Use gateway metric: 20

Activate Huawei E3372 Hi-Link LTE Dongle

1. Ensure dongle is activated with the SIM Carrier
2. On Dongle Teal light will glow solidly (no blink)
3. You may need to disconnect Internet/Broadband connection and reboot
4. Try **Stopping** and **Connecting** the Dongle Interface
5. Eventually Dongle will provide a DHCP address to the router
   Similar to the following:

```
Protocol: DHCP client
Uptime: 0h 15m 11s
MAC: 0C:5B:8F:27:9A:64
RX: 28.90 MB (21750 Pkts.)
TX: 1.22 MB (11751 Pkts.)
IPv4: 192.168.8.100/24
```

Admin Interface for the Huawei E3372 Hi-Link LTE Dongle

http://192.168.8.1

Find Data Usage for Dongle (on Optus Australia)

1. From Dongle Admin interface, select SMS
2. Text 1 to 9999
3. Review information in the reply

Extending OpenWRT with USB Flash Storage

Extroot configuration

To maximise performace use a USB3 flash. SanDisk and PNY sell small devices so they sit near flush on the back of the router.

USB Flash storage used as follows:

1. Extend the base system storage
2. FTP mount point

Easiest from Ubuntu Disk app. Create two partitions on the USB Flash and then plug Flash drive into the router.

Routers with ≥ 8 MiB flash

These devices should have enough space to install the packages we need.

Remove all packages you have installed to add functionality, as they are only wasting space now. After you make the extroot you will have all space you need.

From the command line interface write (on a single line):

```
opkg update && opkg install block-mount kmod-fs-ext4 kmod-usb-storage e2fsprogs kmod-usb-o
```

This installs packages needed for a partition with ext4 filesystem.

Risk-adverse users may wish to create a custom image (as described in the pervious section) containing these tools and especially the kernel modules that are consistent with the firmware kernel so that they are available in failsafe mode.

2. Configuring rootfs_data

Connect with ssh to the device.
Configure `/etc/config/fstab` to mount the `rootfs_data` in another directory in case you need to access the original root overlay to change your extroot settings:

```
DEVICE="$(awk -e '/\s\/overlay\s/{print $1}' /etc/mtab)"
uci -q delete fstab.rwm
uci set fstab.rwm="mount"
uci set fstab.rwm.device="${DEVICE}"
uci set fstab.rwm.target="/rwm"
uci commit fstab
```

Or you can identify the `rootfs_data` partition manually:

```
grep -e rootfs_data /proc/mtd
```

The directory `/rwm` will contain the original root overlay, which is used as the main root overlay until the extroot is up and running.
Later you can edit `/rwm/upper/etc/config/fstab` to change your extroot configuration (or temporarily disable it) should you ever need to.

3. Configuring extroot

See what partitions you have:

```
# block info
/dev/mtdblock2: UUID="9fd43c61-c3f2c38f-13440ce7-53f0d42d" VERSION="4.0" MOUNT="/rom" TYPE
/dev/mtdblock3: MOUNT="/overlay" TYPE="jffs2"
/dev/sda1: UUID="fdacc9f1-0e0e-45ab-acee-9cb9cc8d7d49" VERSION="1.4" TYPE="ext4"
```

Here we see `mtdblock` devices (partitions in internal flash memory), and a partition on `/dev/sda1` that is on a usb flash drive.

Format the partition `/dev/sda1` as ext4 if required:

```
mkfs.ext4 /dev/sda1
```

Now we configure `/dev/sda1` as new overlay via fstab uci subsystem:

```
DEVICE="/dev/sda1"
eval $(block info "${DEVICE}" | grep -o -e "UUID=\S*")
uci -q delete fstab.overlay
uci set fstab.overlay="mount"
uci set fstab.overlay.uuid="${UUID}"
uci set fstab.overlay.target="/overlay"
uci commit fstab
```

If you have a swap partition it will also get recognized and added automatically.

4. Transferring the data

We now transfer the content of the current overlay inside the external drive:

```
mount/dev/sda1 /mnt
cp -a -f /overlay/. /mnt
umount /mnt
```

Reboot the device:

```
reboot
```

Testing

Web interface

1. **LuCI → System → Mount Points** should show USB partition mounted as `overlay`.
2. **LuCI → System → Software** should show free space of overlay partition.

Command-line interface

The USB partition should be mounted to `/overlay`.
Free space for `/` should be the same as `/overlay`.

```
# grep -e /overlay /etc/mtab
/dev/sda1 /overlay ext4 rw,relatime,data=ordered
overlayfs:/overlay / overlay rw,noatime,lowerdir=/,upperdir=/overlay/upper,workdir=/overla

# df /overlay /
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/sda1              7759872    477328   7221104   6% /overlay
overlayfs:/overlay     7759872    477328   7221104   6% /
```

Auto Mounting a USB Flash Drive for FTP Service

This guide assumes you created two partictions on the USB Flash drive.

Using storage devices

Packages installed for USB 3 Flash support.

```
opkg update && opkg install e2fsprogs kmod-usb-storage kmod-usb2 kmod-usb3 kmod-usb-storag
```

To configure external disk space, follow the procedures of this page:

1. Verify storage drivers
2. Verify that the OS recognizes the attached disk and its partitions
3. Create a partition on the USB disk
4. Create a file system in the partition

5. Automount the partition
6. Idle spin down of hard disks

Install and verify USB drivers

This step ensures that required USB storage drivers are properly installed.

1. Start by refreshing the list of available software packages:

```
opkg update
```

2. The typical OpenWrt package already has core USB device drivers installed (if your device has USB ports at all), but might not yet have an USB storage device driver installed. Install this storage driver first (if it is already installed, the following command will just say “is already installed”:

```
opkg install kmod-usb-storage
```

3. Some USB storage devices may require the UAS driver:

```
opkg install kmod-usb-storage-uas
```

4. To check, if the whole USB driver chain is working correctly, install the optional **usbutils** package:

```
opkg install usbutils
```

5. Now connect your USB disk/stick and list your connected devices with a command from these **usbutils**:

```
lsusb -t
```

6. This will output a list of device USB hub ports and connected external storage devices:

```
/:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci-mtk/1p, 5000M
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci-mtk/2p, 480M
    |__ Port 1: Dev 5, If 0, Class=Mass Storage, Driver=usb-storage, 480M
```

- “Bus…”-Lines represent the host chip. Here, the “Driver” will be xhci“ for USB3.0, “ehci” for USB2.0 and “uhci” or “ohci” for USB1.1.

- Lines with “Class=Mass Storage” represent connected USB devices. Here the “Driver” is either “usb-storage” for storage of type Bulk only Transport or “usb-storage-uas” for storage of type USB_Attached_SCSI

In step 4, verify that the output prints no error and has at least one output line for **root_hub** and **Mass Storage** and that each **Driver=** lists a driver name. If not, then refer to the Installing USB Drivers for more suggestions on drivers.

Verify that the OS recognizes the attached disk and partitions

This optional verification step can be used, to check that the OS can properly detect a connected external drive.

1. Ensure your USB disk/stick is stick connected
2. Run in a command line:

```
ls -l /dev/sd*
```

3. This should now show a list of block devices known to the OS

```
brw-------    1 root     root        8,   0 Oct 30 12:49 /dev/sda
brw-------    1 root     root        8,   1 Oct 30 12:49 /dev/sda1
brw-------    1 root     root        8,   1 Oct 30 12:49 /dev/sda2
```

   This should print at least a connected disk like ”/dev/sda“ or ”/dev/sdb“. If no disk at all is listed, recheck USB driver installation and reboot your OpenWrt device once.
4. Install the **block** tool to get more info about existing partitions

```
opkg install block-mount
```

5. Run the **block** tool:

```
block info | grep "/dev/sd"
```

   and you should see output like this, if your disk already has partitions:

```
/dev/sda1: UUID="7feefc8c-badd-4aa3-bf66-a2b1a9a2a586" VERSION="1.0" MOUNT="/ove
/dev/sda2: UUID="7d63798d-8d70-42c1-9a03-bced4013ed07" VERSION="1.10" MOUNT="/mn
```

If a disk already has existing partitions, they get listed as **/dev/sda1**, **/dev/sda2** ,**/dev/sda3** and so on.
If we had connected more than one storage device we would have also a **/dev/sdb1** (first partition of second device), **/dev/sdc1** (first partition of third device) and so on.

Create a partition on the USB disk

if the previous chapter did not list any existing partitions (like ”/dev/sda1“, ”/dev/sda2“, ”/dev/sdb1“…), you have to create a partition first for further storage usage.

1. To do so, install **gdisk**:
   ```
   opkg install gdisk
   ```
2. Start **gdisk** with the disk name identified in the previous chapter:

Given two partitions created this will like be /dev/sda2.

```
```bash
gdisk /dev/sda2
```
```

3. In the interactive gdisk menu, create a partition with gdisk command
   ```
   n
   ```
   This triggers an interactive dialogue: Use the suggested defaults for the partition creation (number, starting sector, size, Hex code)
4. When done, confirm the changes with gdisk interactive command
   ```
   w
   ```
   and then confirm your choice with
   ```
   Y
   ```
5. Keep a note of the created partition name for the next step

Refer to the gdisk help text (write ”?“) in case you need additional help. Stick to a single partition, to stay aligned to the following HowTo.

Install file system drivers and create a file system in the partition

To use a partition for data storage, it needs to be formatted with a file system.

The following is the most simplest (and recommended) default configuration for OpenWrt file system usage.
For advanced users, there are further optional file system options available.

WARNING: This step deletes existing data in that partition. Ensure you have a backup of important files before starting!

- For SSD drives and thumb drives, install F2FS file system and use F2FS to format the partition (in this example '/dev/sda1'):

```
opkg install f2fs-tools
opkg install kmod-fs-f2fs
mkfs.f2fs /dev/sda1
```

- For USB hard disks, install EXT4 file system and use EXT4 to format the partition (in this example '/dev/sda1'):

```
opkg install e2fsprogs
opkg install kmod-fs-ext4
mkfs.ext4 /dev/sda1
```

Automount the partition

Automount ensures that the external disk partition is automatically made available for usage when booting the OpenWrt device

1. Generate a config entry for the fstab file:

```
block detect | uci import fstab
```

2. Now enable automount on that config entry:

```
uci set fstab.@mount[-1].enabled='1'
uci commit fstab
```

3. Optionally enable autocheck of the file system each time the OpenWrt device powers

up:

```
uci set fstab.@global[0].check_fs='1'
uci commit fstab
```

4. Reboot your OpenWrt device (to verify that automount works)
5. After the reboot, check your results: Run

   ```
   uci show fstab
   ```

   to see something like this

   ```
   fstab.@global[0]=global
   fstab.@global[0].anon_swap='0'
   fstab.@global[0].anon_mount='0'
   fstab.@global[0].auto_swap='1'
   fstab.@global[0].auto_mount='1'
   fstab.@global[0].check_fs='0'
   fstab.@global[0].delay_root='5'
   fstab.@mount[0]=mount
   fstab.@mount[0].target='/mnt/sda1'
   fstab.@mount[0].uuid='49c35b1f-a503-45b1-a953-56707bb84968'
   fstab.@mount[0].enabled='1'
   ```

6. Check the “enabled” entry. It should be '1'.
7. Note the “target” entry. This is the file path, where your attached USB storage drive can be accessed from now on. E.g. you can now list files from your external disk:

   ```
   ls -l /mnt/sda2
   ```

8. Run the following command, to verify that the disk is properly mounted at this path

   ```
   block info
   ```

   The result will be:

   ```
   ...
   /dev/sda1: UUID="2eb39413-83a4-4bae-b148-34fb03a94e89" VERSION="1.0" MOUNT="/mnt
   ```

9. Your external storage is now ready for further usage:

   ```
   service fstab boot
   ```

Very Secure FTP Server (vsftpd) with Anonymous Access

Very Secure FTP Server

Install:

```
opkg update && opkg install vsftpd
```

nano ***/etc/vsftpd.conf***

replace existing contents with thg following

```
background=YES
listen=YES
anonymous_enable=YES
ftp_username=nobody
anon_root=/mnt/sda1/lab
no_anon_password=YES
local_enable=NO
write_enable=NO
local_umask=022
check_shell=NO
dirmessage_enable=YES
ftpd_banner=Welcome to Glovebox FTP service.
session_support=NO
#syslog_enable=YES
#userlist_enable=YES
#userlist_deny=NO
#userlist_file=/etc/vsftpd/vsftpd.users
#xferlog_enable=YES
#xferlog_file=/var/log/vsftpd.log
#xferlog_std_format=YES
###
### TLS/SSL options
### example key generation: openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/vsftpd/vsftpd
#ssl_enable=YES
#allow_anon_ssl=NO
#force_local_data_ssl=NO
#force_local_logins_ssl=NO
#ssl_tlsv1=YES
#ssl_sslv2=NO
#ssl_sslv3=NO
#rsa_cert_file=/etc/vsftpd/vsftpd_cert.pem
#rsa_private_key_file=/etc/vsftpd/vsftpd_privkey.pem

#local_root=/mnt/sda1/lab
```

Reload and Start Very Secure FTP Services

```
/etc/init.d/vsftpd reload
/etc/init.d/vsftpd start
```

Install wget

```
opkg update && opkg install wget libustream-openssl ca-bundle ca-certificates
```

wget Visual Studio Code Insider Builds

```
wget https://go.microsoft.com/fwlink/?LinkID=760865  # .deb64
wget https://go.microsoft.com/fwlink/?LinkId=723966  # macOS
wget https://aka.ms/win32-x64-user-insider           # Windows 64
```

GitHub

Note: use **git:** rather than **https:** transport

```
git clone --depth 1 git://github.com/gloveboxes/PyCon-Hands-on-Lab.git
```

Python3 and Pip Usage

```
python3 -m pip install ...
```

Example:

```
python3 -m pip install flask paho-mqtt
```

CPU Temperature

```
cat /sys/class/thermal/thermal_zone0/temp
```