OpenWrt (One) build system: lessons in *all* the compliance and how to broadly apply them

Denver Gingerich

Linux Plumbers Conference 2025

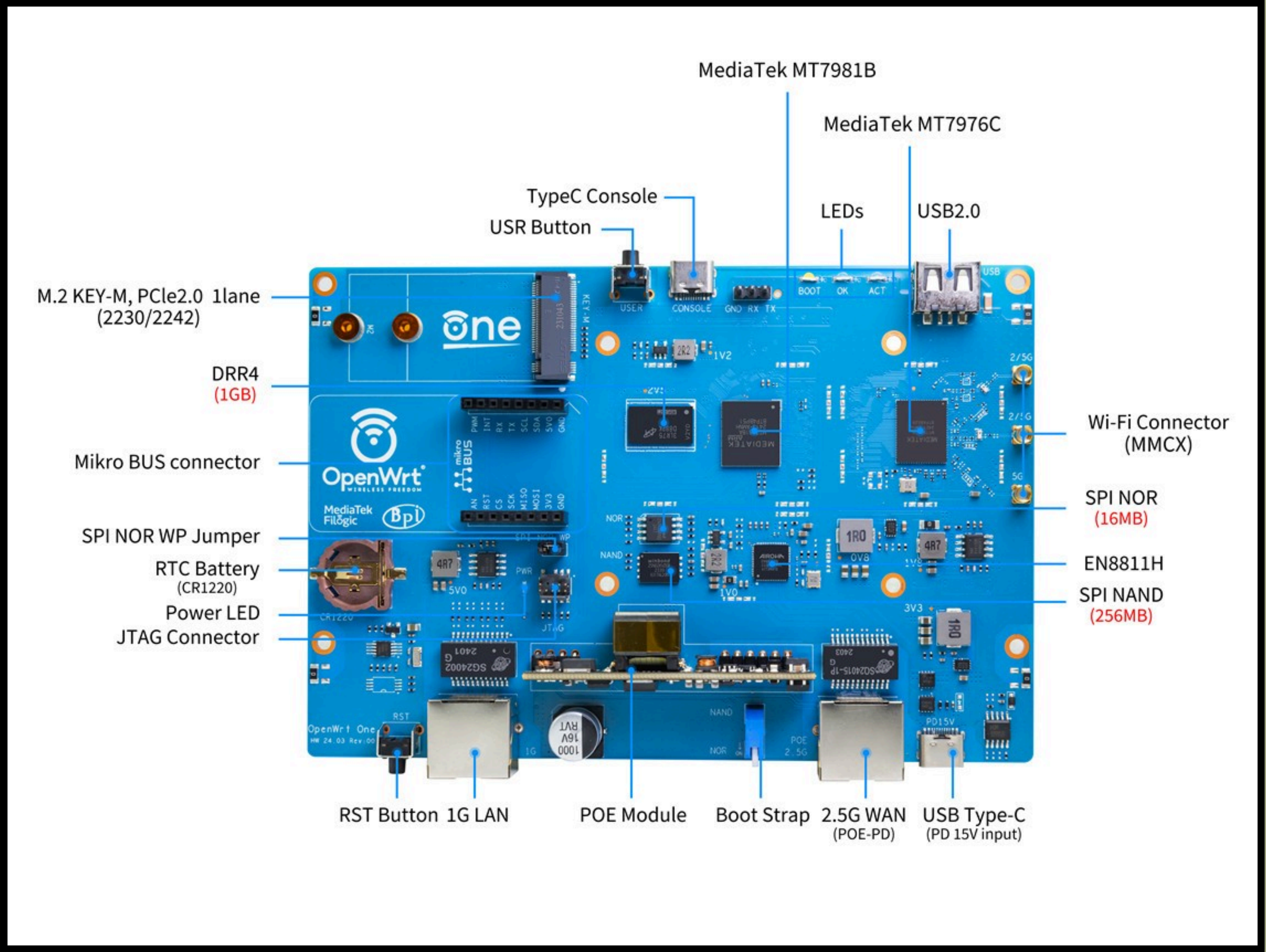
Thursday 11 December 2025

https://ossguy.com/talks/20251211_lpc/

introduction

introduction

introduction



the "why" of embedded build systems

the "why" of embedded build systems

build an embedded device

the "why" of embedded build systems

build an embedded device

sell an embedded device

the "why" of embedded build systems

build an embedded device

sell an embedded device

to sell you must comply

all compliance starts with: what's in it?

all compliance starts with: what's in it?

package versions

all compliance starts with: what's in it?

package versions

out-of-tree patches

all compliance starts with: what's in it?

package versions

out-of-tree patches

non-free stuff

all compliance starts with: what's in it?

package versions

out-of-tree patches

non-free stuff

linking

example: OpenWrt (One) build system

example: OpenWrt (One) build system

start with OpenWrt base tree

example: OpenWrt (One) build system

start with OpenWrt base tree

```
make menuconfig
```

example: OpenWrt (One) build system

start with OpenWrt base tree

```
make menuconfig
```

```
make download
```

example: OpenWrt (One) build system

start with OpenWrt base tree

`make menuconfig`

`make download`

create `how_to_*` files

what does this give us?

what does this give us?

a minimal set of things we can reason about

what does this give us?

a minimal set of things we can reason about

simple base, simple config, rest is upstream

what does this give us?

a minimal set of things we can reason about

simple base, simple config, rest is upstream

rest of development process steered toward simplicity

what does this give us?

a minimal set of things we can reason about

simple base, simple config, rest is upstream

rest of development process steered toward simplicity

easy-to-produce self-contained source tarball

not part of your build system/process?

not part of your build system/process?

those `how_to_*` files

not part of your build system/process?

those `how_to_*` files

let's fix that

not part of your build system/process?

those `how_to_*` files

let's fix that

examples: https://sfconservancy.org/usethesource/

build and install (OpenWrt One, 1 of 3)

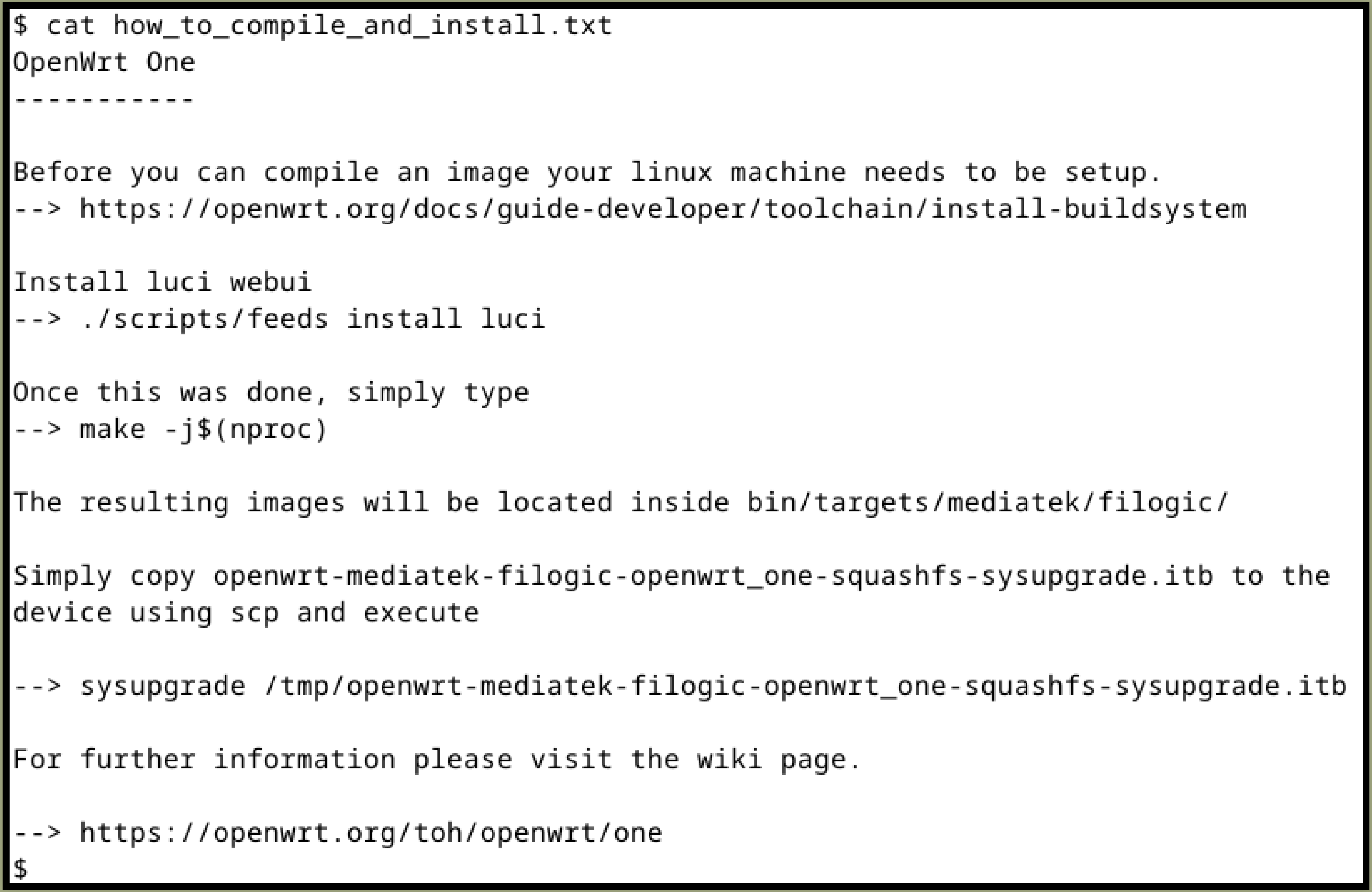
```
 BSDmakefile
+.config
 config
 Config.in
 COPYING
 .devcontainer
+dl
+feeds
 feeds.conf.default
 .gitattributes
 .github
 .gitignore
+how_to_basic_wifi_config.txt
+how_to_build_system_setup.txt
+how_to_compile_and_install.txt
 include
 LICENSES
 Makefile
```

build and install (OpenWrt One, 2 of 3)

```
$ cat how_to_build_system_setup.txt
Build system setup

 * Assuming a GNU/Linux environment, otherwise see alt guides.
...
Package prerequisites for various Linux distributions:
...
* Debian / Ubuntu / Mint:
  sudo apt update
  sudo apt install build-essential clang flex bison g++ gawk \
       gcc-multilib g++-multilib gettext git libncurses5-dev \
       libssl-dev python3-setuptools rsync unzip zlib1g-dev
```

build and install (OpenWrt One, 3 of 3)



```
$ cat how_to_compile_and_install.txt
OpenWrt One
-----------

Before you can compile an image your linux machine needs to be setup.
--> https://openwrt.org/docs/guide-developer/toolchain/install-buildsystem

Install luci webui
--> ./scripts/feeds install luci

Once this was done, simply type
--> make -j$(nproc)

The resulting images will be located inside bin/targets/mediatek/filogic/

Simply copy openwrt-mediatek-filogic-openwrt_one-squashfs-sysupgrade.itb to the
device using scp and execute

--> sysupgrade /tmp/openwrt-mediatek-filogic-openwrt_one-squashfs-sysupgrade.itb

For further information please visit the wiki page.

--> https://openwrt.org/toh/openwrt/one
$
```

build and install (Samsung, 1 of 4)

```
* Building linux
    * Unpack the linux tarball and cd into it.
    * "cp amber.rel.config .config"
    * "make oldconfig" and "make Image"
    * "cp arch/arm/boot/Image ../uImage"

* Building busybox
    * Unpack the busybox tarball and cd into it.
    * "make CROSS_COMPILE=arm-v5t-le-".
```

build and install (Samsung, 2 of 4)

```
# unsquashfs -d rootfs rootfs.img
# rm ./rootfs/bin/busybox
# cp [path_to_busybox]/busybox ./rootfs/bin/
# rm -f rootfs.img
# mksquashfs ./rootfs rootfs.img
# rm -rf ./rootfs
```

build and install (Samsung, 3 of 4)

```
1. Connect serial cable to Ex-Link(serial) port.

2. Make folder 'update' in usb memory drive and copy boot.img,
   rootfs.img and uImage files into the 'update' folder.

3. Connect usb memory drive to the usb port of your TV.

4. In the status of unpluged power cable, push 'Shift' key
   and '~'key at the same time, then plug in power cable.
   After 2 seconds later, press enter key.
   You can find uboot menu in serial message.

5. Press '0' key(in keyboard) and Enter.
```

build and install (Samsung, 4 of 4)

```
6. Type "bbm usb", then Enter.
                [BHPLCD ]# bbm usb
                ...
                [BBM:   ]  4 : kernel image    ...   "uImage"
                [BBM:   ]  5 : root file system..."rootfs.img"
7. In step 6, select '4 : Kernel image'.
8. Type "/update/uImage".
                 ......   [ONW:   ]  | IMAGE WRITE FINISHED!
9. In step 6, select '5 : root file system'.
10. Type "/update/rootfs.img".
                 .......  [ONW:   ]  | IMAGE WRITE FINISHED!
```

build and install (ThinkPenguin, 1 of 2)

```
tar fzxv libreCMC-v1.5.14-src.tar.gz

cd librecmc

Simply running "make" will build the first firmware image.
The build system will extract all included sources, build the
cross-compile toolchain, kernel and all chosen applications.

When the build completes without any issues, the resulting
image will be found in : bin/targets/ath79/generic/
```

build and install (ThinkPenguin, 2 of 2)

```
Open a web browser and go to : https://192.168.10.1

Enter the admin password for the router (default is : none )

Navigate to "System -> Backup / Flash Firmware".

Upload the new firmware by clicking the "browse" button under
"Flash new firmware image".  Click the "Flash Image" button.

Wait a about 1 - 2.5 min. The router will restart itself then
the page should refresh, bringing you back to the login page.
```

build and install (AVM, 1 of 3)

```
We ourselves have carried out the installation according to
  these instructions under Ubuntu 22.04.4.

The following commands are now executed on the Ubuntu machine:

  git clone https://github.com/Freetz-NG/freetz-ng ~/freetz-ng
  cd ~/freetz-ng
  tools/prerequisites install # -y

Execute the command 'make menuconfig' and select the
  appropriate router model and save the configuration.

Now some more tools have to be installed by executing
  'make tools'
```

build and install (AVM, 2 of 3)

```
The original firmware image must now be unpacked into a new
folder using the commands:

  mkdir unpacked_firmware
  ./fwmod -u -d unpacked_firmware FRITZ.Box_4020.07.03.image

Within the directory './unpacked_firmware/original/filesystem'
we now replace the desired files with specially generated
files (we have replaced uClibc here, as you intended by you in
your project) with the following command:

  cp path-to-custom-file/ld-uClibc-1.0.14.so \
    ./unpacked_firmware/original/filesystem/lib/ld-uClibc-1.0.14.so
```

build and install (AVM, 3 of 3)

```
The firmware must now be packed back into an image file by
entering the following command:

  ./fwmod -p -d unpacked_firmware FRITZ.Box_4020.07.03.image

Carry out the following sub-steps:

* Connect the FRITZ!Box 4020 to the Ubuntu PC via Ethernet,
  but keep it disconnected from the power supply.
* Use 'ifconfig' to check whether an IPv4 address from
  192.168.178.0/255 is configured on the Ethernet adapter
* Call up the flash tool of the Freetz project:
  'sudo tools/push_firmware \
     unpacked_firmware/4020_07.03.ger_20240315-102855.image'
* Supply the FRITZ!Box 4020 with power.
```

summary

summary

make it easy to reason about

summary

make it easy to reason about

make it easy to build

summary

make it easy to reason about

make it easy to build

make it easy to install

how about your build system?

https://ossguy.com/talks/20251211_lpc/

become a Sustainer:

https://sfconservancy.org/sustainer/