Linux kernel support DRM and Wayland

This page introduce building the Linux kernel with DRM features and run Wayland on Linux. The Linux kernel can be built on your workstation or ODROID-N2 natively but, in this page, we are assuming to build it and run on ODROID-N2 natively.

Setting up the Ubuntu minimal

The new Linux kernel for DRM can be run and tested on the Ubuntu minimal which does not have Linux desktop packages. Before starting to build and run the Linux kernel for DRM, you must complete the booting with stock Linux kernel in the minimal OS image and please double check if the minimal OS can run properly.

```
$ wget https://dn.odroid.com/S922X/ODROID-N2/Ubuntu/https://dn.odroid.com/S922X/ODROID-N2/Ubuntu/ubuntu-18.04.4-4.9-minimal-odroid-n2-20200229.img.xz
$ unxz https://dn.odroid.com/S922X/ODROID-N2/Ubuntu/ubuntu-18.04.4-4.9-minimal-odroid-n2-20200229.img.xz
$ sudo dd if=https://dn.odroid.com/S922X/ODROID-N2/Ubuntu/ubuntu-18.04.4-4.9-minimal-odroid-n2-20200229.img of=/dev/disk/by-id/<your/memory/card>
```

Building Linux kernel

In order to build the Linux kernel, the building environment has to be set and please visit the link for detail. Link

```
$ sudo apt install git build-essential
```

The Linux kernel to enable DRM features are in the same repository of Hardkernel's Github and the same branch **odroidn2-4.9.y**, the source tree can be downloaded with git command.

```
$ git clone --depth 1 https://github.com/tobetter/linux.git -b odroidg12-4.9.y_drm
$ cd linux
$ make odroidg12_drm_defconfig
$ make <-j16>
```

The Linux kernel image and the device trees are created when the build the Linux kernel is completed.

```
$ ls arch/arm64/boot/Image.gz
arch/arm64/boot/Image.gz
$ ls arch/arm64/boot/dts/amlogic/meson64_odroidn2*.dtb
arch/arm64/boot/dts/amlogic/meson64_odroidn2.dtb
```

```
arch/arm64/boot/dts/amlogic/meson64_odroidn2_drm.dtb
```

Running on Ubuntu minimal

Replacing the kernel blobs

Once you have the kernel blobs, Image.gz and meson64_odroidn2_drm.dtb, they can be copied to the same directory where the current kernel blobs are.

```
$ sudo cp arch/arm64/boot/Image.gz /media/boot
$ sudo cp arch/arm64/boot/dts/amlogic/meson64_odroidn2_drm.dtb
/media/boot/amlogic/meson64_odroidn2.dtb
$ sudo make modules_install firmware_install
$ sync
```

In order to show up the decent color on the display, the booting logo must be copied into the same **BOOT** partition.

```
$ sudo wget
https://github.com/codewalkerster/android_device_hardkernel_odroidn2/blob/s9
22_9.0.0_master/files/hardkernel-720.bmp.gz -o /media/boot/boot-logo.bmp.gz
```

Everything is copied to your memory card and it's time to boot the new Linux kernel.

Running Wayland

Please note that the new Linux kernel won't show anything to the display, so USB-to-UART debugging tool is required. Alternatively, you can connect through SSH.

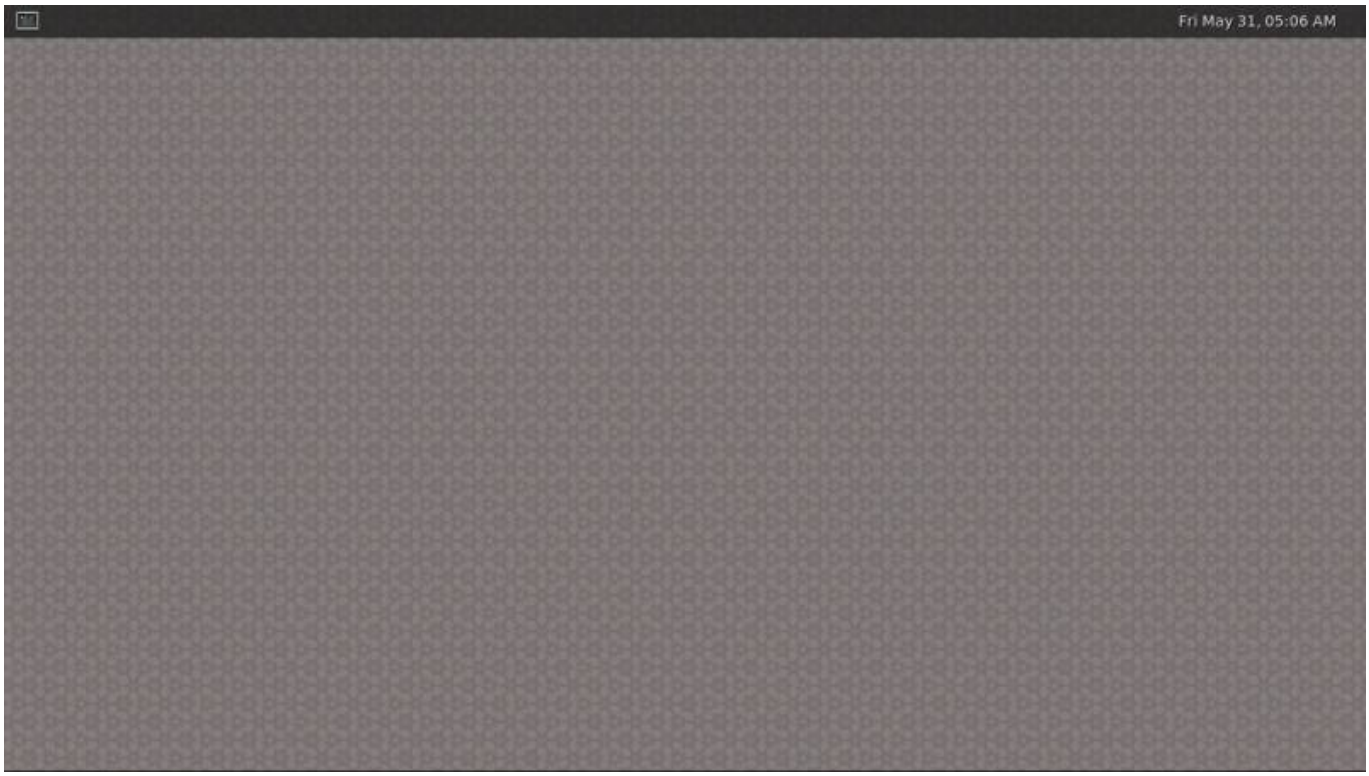
Some packages must be removed and/or replaced before running Wayland and rebooted with new packages.

```
$ sudo apt -y update
$ sudo apt -y upgrade
$ sudo apt -y remove mali-fbdev
$ sudo apt -y autoremove
$ sudo apt -y install weston
$ wget http://ppa.linuxfactory.or.kr/pool/non-free/m/mali-bifrost-
driver/mali-bifrost-w
ayland-driver_0.1-5+202001290959~bionic_arm64.deb
$ sudo dpkg -i mali-bifrost-wayland-
driver_0.1-5+202001290959~bionic_arm64.deb
$ sudo reboot
```

For quick test to run DRM and Wayland applications, simply **weston** can be launched.

```
$ sudo -s
```

```
# weston --tty=1 &
```



For testing if Wayland client can run, **glmark2 for Wayland** can be installed and run.

```
# apt -y install glmark2-es2-wayland
# glmark2-es2-wayland
glmark2-es2-wayland
=======================================================
    glmark2 2014.03+git20150611.fa71af2d
=======================================================
    OpenGL Information
    GL_VENDOR:     ARM
    GL_RENDERER:   Mali-G52
    GL_VERSION:    OpenGL ES 3.2
v1.r16p0-01rel0.2943fc4ef9657d91ee32c9a58dec6cd2
=======================================================
```

Running Chromium on Wayland

Installation prebuilt Chromium binaries

There is no Chromium package installable for ODROID-N2 yet, but Amlogic provide the prebuilt binary which is run on Wayland. The files can be downloaded and installable with these commands.

```
$ sudo apt install git
```

```
$ git clone --depth=1
https://gitlab.com/tobetter/buildroot_linux_amlogic_chromium.git -b
aml64_buildroot_master chromium
$ cd chromium
$ sudo ./install.sh
```

Launching Chromium

Once installation is done, ODROID-N2 has the files in **/usr/bin/chromium-browser** and **/etc/init.d/chromium** to launch the Chromium. The default page to load with Chromium by default can be set in another file **/etc/chromium.default**. For example,

```
$ cat /etc/chromium.default
#DEFAULT_URL=https://webglsamples.org/aquarium/aquarium.html
#DEFAULT_URL=https://bookcase.chromeexperiments.com/
#DEFAULT_URL=https://webglsamples.org/dynamic-cubemap/dynamic-cubemap.html
DEFAULT_URL=https://webglsamples.org/google-io/2011/1000-objects.html
```

```
$ sudo /etc/init.d/chromium start
```

Video

Known issues

There are a couple of issues to run DRM features on the Linux kernel 4.9 for ODROID-N2.

No point cursor

A point cursor just shows a second and disappear immediately.

Blurry screen or blank screen after 'Weston' is started

The DRM driver in the Linux kernel 4.9 seems not to initialize the display properly on boot. As a workaround, a booting logo must be displayed by U-boot so **boot-logo.bmp.gz** must be copied to the partition **BOOT**.

Running on another OS distros

Currently, the Linux kernel with DRM is tested with Debian based Linux distros and Mali Userspace Blobs can be downloaded.

- Debian Stretch
- Debian Buster
- Ubuntu Bionic
- Ubuntu Cosmic

Other than Debian based distros, Mali Userspace blobs can be copied manually and they can be

downloaded from Hardkernel's FTP server.