

# ODROID

Year Four  
Issue #38  
Feb 2017

## Magazine



# Get moving VU8-C

Take your ODRROID-C0, C1 or C2 on the road with you

- Setting up your ODRROID-XU4 as a general-purpose NAS

- Control your ODRROID power usage with the SmartPower2



# What we stand for.

We strive to symbolize the edge of technology, future, youth, humanity, and engineering.

Our philosophy is based on Developers.  
And our efforts to keep close relationships with developers around the world.

For that, you can always count on having the quality and sophistication that is the hallmark of our products.

Simple, modern and distinctive.  
So you can have the best to accomplish everything you can dream of.



## HARDKERNEL



We are now shipping the ODROID-C2 and ODROID-XU4 devices to EU countries! Come and visit our online store to shop!

Address: Max-Pollin-Straße 1  
85104 Pförring Germany

Telephone & Fax  
phone: +49 (0) 8403 / 920-920  
email: [service@pollin.de](mailto:service@pollin.de)

Our ODROID products can be found at  
<http://bit.ly/1tXPXwe>





**W**e recently built an **ODROID-VU8** kit (USD\$90 <http://bit.ly/2k8bML5>), and now have a great 64-bit Ubuntu 16.04 touchscreen tablet with Gigabit Ethernet. The 1024 x 768 resolution is perfect for both Android and Linux, and the case allows an **ODROID-C0, C1, C1+ or C2** to be mounted inside while giving access to two USB ports. It can be used as a gaming tablet, entertainment system, portable programming workstation, network diagnostic kit, and much more. Hardkernel has also released an updated version of the popular SmartPower bench power supply (USD\$40 <http://bit.ly/2j3hhcv>), which includes a built-in web server.

To make using your **ODROID-VU8** easier to use, we show two methods of setting up a dual or triple boot system. Adrian details setting up an **ODROID-XU4** as a Network Attached Storage (NAS), @korinel presents a guide to using HomeBridge to integrate the technologies in your home, and @Brian.K discusses using BuildRoot for building Linux to generate a disk image. We also reveal a simple tweak to reduce the power consumption of the **ODROID-C2** even more, and Tobias compares the speed of the PPSSPP emulator on several **ODROID** models.

ODROID Magazine, published monthly at <http://magazine.odroid.com>, is your source for all things ODROIDian. Hard Kernel, Ltd. • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815. Hardkernel manufactures the ODROID family of quad-core development boards and the world's first ARM big.LITTLE single board computer. For information on submitting articles, contact [odroidmagazine@gmail.com](mailto:odroidmagazine@gmail.com), or visit <http://bit.ly/1yplmXs>. You can join the growing ODROID community with members from over 135 countries at <http://forum.odroid.com>. Explore the new technologies offered by Hardkernel at <http://www.hardkernel.com>.



**HARDKERNEL**



Hundreds of products available online for the professional developer and hobbyist alike



**ODROID-XU4**



**ODROID-C1+**



**ODROID-C0**



**OWEN ROBOT KIT**



**ODROID-C2**



**VU7 TABLET KIT**



## **Rob Roy, Chief Editor**

I'm a computer programmer in San Francisco, CA, designing and building web applications for local clients on my network cluster of ODROIDS. My primary languages are jQuery, Angular JS and HTML5/CSS3. I also develop pre-built operating systems, custom kernels and optimized applications for the ODROID platform based on Hardkernel's official releases, for which I have won several Monthly Forum Awards. I use my ODROIDS for a variety of purposes, including media center, web server, application development, workstation, and gaming console. You can check out my 100GB collection of ODROID software, prebuilt kernels and OS images at <http://bit.ly/1fsaXQs>.

---



## **Bruno Doiche, Senior Art Editor**

Funny issue trivia: Bruno and Rob were talking about the VU8 images for the cover. Bruno wanted to get two angles to showcase the VU8 as a versatile platform, but all the material we had available was in low resolution. Rob snapped some photos of his brand new VU8 and sent them to Bruno. Due to the timezone difference, Bruno received them at 2AM and didn't notice that they were not in perfect focus, so he had to toil a lot to get the machine presentable for the cover. But it was worth it!

---



## **Manuel Adamuz, Spanish Editor**

I am 31 years old and live in Seville, Spain, and was born in Granada. I am married to a wonderful woman and have a child. A few years ago I worked as a computer technician and programmer, but my current job is related to quality management and information technology: ISO 9001, ISO 27001, and ISO 20000. I am passionate about computer science, especially microcomputers such as the ODROID and Raspberry Pi. I love experimenting with these computers. My wife says I'm crazy because I just think of ODROIDS! My other great hobby is mountain biking, and I occasionally participate in semi-professional competitions.

---



## **Nicole Scott, Art Editor**

Nicole is a Digital Strategist and Transmedia Producer specializing in online optimization and inbound marketing strategies, social media management, and media production for print, web, video, and film. Managing multiple accounts with agencies and filmmakers, from web design and programming, Analytics and Adwords, to video editing and DVD authoring, Nicole helps clients with the all aspects of online visibility. Nicole owns an ODROID-U2, a number of ODROID-U3's, and Xu4's, and looks forward to using the latest technologies for both personal and business endeavors. Nicole's web site can be found at <http://www.nicolecscott.com>.

---



## **James LeFevour, Art Editor**

I'm a Digital Media Specialist who is also enjoying freelance work in social network marketing and website administration. The more I learn about ODROID capabilities, the more excited I am to try new things I'm learning about. Being a transplant to San Diego from the Midwest, I am still quite enamored with many aspects that I think most West Coast people take for granted. I live with my lovely wife and our adorable pet rabbit; the latter keeps my books and computer equipment in constant peril, the former consoles me when said peril manifests.

---



## **Andrew Ruggeri, Assistant Editor**

I am a Biomedical Systems engineer located in New England currently working in the Aerospace industry. An 8-bit 68HC11 microcontroller and assembly code are what got me interested in embedded systems. Nowadays, most projects I do are in C and C++, or high-level languages such as C# and Java. For many projects, I use ODROID boards, but I still try to use 8bit controllers whenever I can (I'm an ATMEL fan). Apart from electronics, I'm an analog analogue photography and film development geek who enjoys trying to speak foreign languages.

---



## **Venkat Bommakanti, Assistant Editor**

I'm a computer enthusiast from the San Francisco Bay Area in California. I try to incorporate many of my interests into single board computer projects, such as hardware tinkering, metal and woodworking, reusing salvaged materials, software development, and creating audiophile music recordings. I enjoy learning something new all the time, and try to share my joy and enthusiasm with the community.

---



## **Josh Sherman, Assistant Editor**

I'm from the New York area, and volunteer my time as a writer and editor for ODROID Magazine. I tinker with computers of all shapes and sizes: tearing apart tablets, turning Raspberry Pis into PlayStations, and experimenting with ODROIDS and other SoCs. I love getting into the nitty gritty in order to learn more, and enjoy teaching others by writing stories and guides about Linux, ARM, and other fun experimental projects.

# INDEX



**DUAL BOOT ODROID-C2 - 6**



**GENERAL PURPOSE NAS - 8**



**ODROID-VU8 - 16**



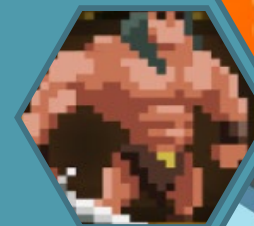
**BUILDROOT - 19**



**ANDROID GAMING: SKY FORCE RELOADED- 20**



**HOMEBRIDGE - 20**



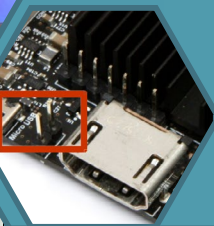
**ANDROID GAMING: TAP N SLASH - 21**



**LINUX GAMING - 23**



**SMARTPOWER2 - 29**



**POWER CONSUMPTION - 29**



**MEET AN ODROIDIAN - 30**

# DUAL AND TRIPLE BOOT FOR ODROID-C2

## EASILY SWITCH BETWEEN OPERATING SYSTEMS ON A SINGLE ODROID

by @Molorius and @loboris



There are a couple of different methods available for setting up the ODROID-C2 as a dual boot system, depending on your technical expertise. Both techniques will allow you to run any combination of operating systems, such as Android and Ubuntu, OpenELEC and Lakka, or Kodiuntu and Debian. If you want to use a boot menu to select the operating system, or have a triple boot system, use the automated method created by @loboris. To have full control over the switching scripts, or to do so programmatically, use the manual method as described by @Molorius, which can be adapted to have any number of operating systems available.

### Manual method

First, create a bootable Android image on either an eMMC or SD card by downloading it from <http://bit.ly/2be993R> and copying it to the media using Win32 Disk Imager or a similar method. Then, do the same with the Ubuntu image from <http://bit.ly/2b58GEe>.

1. Flash the Ubuntu image onto the emmc. Do not put this card in your Odroid (just yet). We need to edit the file `/etc/fstab` on this card, which requires root. On my computer, I typed in terminal:

```
$ sudo gedit /media/username/rootfs/etc/fstab
```

Change `"LABEL=boot"` to `"LABEL=VFAT"`. Save and quit.

2. Save everything in the boot partition in a separate directory on your computer.

3. For the next section, I used a program called Disks, which comes with Ubuntu. All that we're doing is creating a .img of the rootfs partition. On Disks, I navigated to the rootfs partition, and select "Create Partition Image". Save this somewhere on your computer. Write down how big the file is, you will

need this later. You only need to do steps 1-3 once to get the data. If you need to reflash Ubuntu, start from step 4.

4. Flash Android onto this card. Yep, just do it. I couldn't figure out any other way. Load it into your Odroid, and let it install. Open up Terminal Emulator, and punch in:

```
$ vi storage/internal/boot.ini
```

Change `"root=/dev/mmcblk0p2"` to `"root=/dev/mmcblk1p2"`. Save and quit.

5. Put this card back in your computer. In Disks (or whatever you're using), delete the cache and rootsystem partitions. You can put another partition here in their place, if you wish. Delete the last partition as well. Make a new partition in its place, the same size as the .img you made previously (see step 3). In Disks, I selected an option to "Restore Partition Image" on the partition I just made. Choose the .img you made in step 3. This should only replace the last partition, not everything on the card.

6. In the VFAT partition, rename "boot.ini" to "boot.ini.android". Copy the files you saved during step 2 into this partition.

NEVER MOVE THE VFAT PARTITION. Yes, it's weird to have the giant space in front of it. But I couldn't figure out any other way to do it. Even moving it, then putting it back, still made it so Android wouldn't boot. Ubuntu loads no matter where this is. If anyone finds another way, please let me know. It would be great to not have to flash Android each time.

Flash the Android image to your sd card. Load it into your Odroid. Open up Terminal Emulator, and punch in:

```
$ su
$ mount -o rw,remount /
```

```
$ vi fstab.odroidc2
```

Change “/dev/block/mmcblk0p4” to /dev/block/mmcblk1p4”. Change “/dev/block/mmcblk0p3” to /dev/block/mmcblk1p3”. Shutdown your Odroid. Put the card in your computer, and delete the VFAT partition. You can fill this in later for additional space. DO NOT MOVE ANY OF THE OTHER PARTITIONS. You’re now able to switch OS’s! To do so, you just need to change the “boot.ini” file to whichever you wish to boot to. Here is a way to do that quickly on Ubuntu and Android.

On Ubuntu, type the following in a Terminal window:

```
$ pluma boot_android.sh
```

Add the following lines to the file, save and exit:

```
#!/bin/bash
mv /media/boot/boot.ini /media/boot/boot.ini.ubuntu
mv /media/boot/boot.ini.android /media/boot/boot.ini
reboot
```

Make the script executable:

```
$ sudo chmod 777 boot_android
```

Run the file by typing the following command in a Terminal window. I suggest making a desktop launcher for this command:

```
$ ./boot_android.sh
```

On Android, type the following command into the Terminal app:

```
$ su
$ mount -o rw,remount /
$ vi /bin/boot_ubuntu.sh
```

Then, create the script:

```
#!/bin/sh
mv /storage/internal/boot.ini /storage/internal/boot.ini.android
mv /storage/internal/boot.ini.ubuntu /storage/internal/boot.ini
reboot
```

Make it executable:

```
$ chmod 777 /bin/boot_ubuntu.sh
```

Run the file by typing this command in the Terminal app:

```
$ boot_ubuntu.sh
```

I suggest using the Term Shortcut widget to be able to run this script from the main screen. For comments, questions and suggestions, please visit the original post at <http://bit.ly/2iGKvzb>.

## Automated method

For those who wish to create a dual boot system using an automated script instead, download the ODROID Installer scripts from <http://bit.ly/2jtlVvYX>. You can either use the pre-built images by unzipping the file multiboot\_install\_images.zip and running the following command:

```
$ sudo dd if=multiboot_install_[c2|xu4].img \
of=/dev/sdX bs=1MB oflag=direct
```

Alternatively, you can install the image yourself by building the installer image:

```
$ sudo ./prepare_selfinst <destination_
card>|<destination_image_name> c2|xu4
```

Once the image file is created, you can write the image to SD card using the dd command as described above.

Next, copy your installation sources (Android update image “update.zip”, the Linux .img file, and/or the OpenELEC .tar file) to the first partition of your USB drive. Rename the Linux installation image to “linux.img”, rename the OpenELEC installation to oelec.tar, and rename the Android installation archive to update.tar.gz or update.zip. Connect your USB drive to the ODROID, insert the SD card containing the installer image, and power on the ODROID-C2. Select “Install” from the menu, then follow the instructions to select the desired partition sizes and installation destination (SD card or eMMC module). After the installation has completed, remove the installer SD card and boot the device from the main media.

## Tips

You can select the desired Android resolution by manually editing the boot.ini.android file, since the Hardkernel ODROID Utility does not work with this setup. There is also an option to describe the operating systems in the boot menu by adding the following line to the boot.ini.\* files:

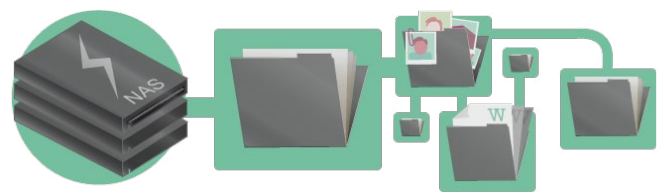
```
#DESCRIPTION “<description goes here>”
```

For comments, questions and suggestions, please visit the GitHub repository at <http://bit.ly/2jtlVvYX>.

# SETTING UP YOUR XU4 AS A GENERAL-PURPOSE NETWORK ATTACHED STORAGE (NAS) DEVICE

## YOUR HOME SWISS ARMY KNIFE SERVER

by Adrian Popa (@mad\_ady)



I purchased my ODROID-XU4 with the intent of converting it into a NAS. However, I did not want to settle on a specialized NAS distro like OpenMediaVault because I wanted my ODROID-XU4 to do much more than being a plain old NAS. For instance, I plan on transcoding TV shows recorded from my TV to the H264 standard, using the ODROID-XU4's hardware encoder (as described in <http://bit.ly/2jnv4Za>), and also make use of the GPIO pins later on. One more issue I had with OpenMediaVault is that it runs on top of Debian, and I wanted to keep using Ubuntu in order to benefit from newer packages.

I would be losing much of the convenience of using a specialized distro and consequently have to discover alternate ways of doing things in a simple and user-friendly way. This presents an opportunity to gain new knowledge.

### These are the steps we will need to take:

**Install the mainline kernel Ver. 4.9 (optional)**

**Install Webmin (<http://bit.ly/J5Wtfl>) for easier management**

**Mount the disks**

**Set up network shares (Samba/NFS)**

**Install Owncloud**

**Secure and optimize the OS**

These instructions presume that you have medium or higher level system expertise.

### Install the mainline kernel (4.9)

The ODROID-XU4 has the benefit of having pretty good mainline kernel support, which I need specifically for video

transcoding. Mainline kernel has the benefit of newer drivers and better support. However, it also comes with new issues: buggy HMP support, USB3 instabilities, and no sound over HDMI, to name a few. This implies that you have to weigh the pros and cons and make a decision according to your needs.

In order to install the mainline kernel, you will need to follow either the official instructions (Hardkernel is working on an official 4.9 kernel for ODROID-XU4 and most likely will release it as a deb package) or follow the general instructions at <http://bit.ly/2jnv4Za>. Remember to fully unplug your ODROID-XU4 from all power sources (power, HDMI, USB) otherwise you will miss your network when you first boot.

### Webmin

Every NAS needs a nice web GUI. Unfortunately, OpenMediaVault's GUI is not an option, and after searching for a long while for alternatives I settled on using Webmin. Webmin has been around since 1997 and has solid support for general server maintenance tasks. It has the advantage that even inexperienced users can find their way around and with the integrated help in order to set set up and manage all kinds of servers like Apache, MySQL, Mail, DNS, and more. It has solid support for RAID and LVM management, and also supports Samba and NFS file sharing. Unfortunately, it lacks support for newer services like Transmission or Owncloud, but I can always configure them manually.

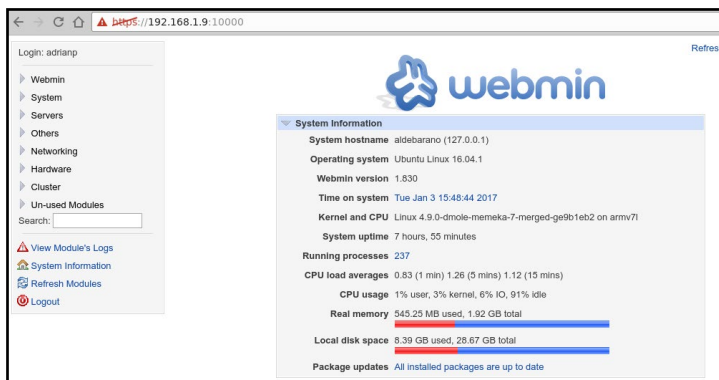
To install it, follow the steps below:

```
$ echo "deb http://download.webmin.com/download/re-
pository \"
```



```
sarge contrib" | sudo tee /etc/apt/sources.list.d/
webmin.list
$ wget http://www.webmin.com/jcameron-key.asc
$ sudo apt-key add jcameron-key.asc
$ rm jcameron-key.asc
$ sudo apt-get update
$ sudo apt-get install libapt-pkg-perl \
libnet-ssleay-perl libauthen-pam-perl \
libio-pty-perl apt-show-versions \
apt-transport-https
$ sudo apt-get install webmin
$ sudo systemctl enable webmin
$ sudo systemctl start webmin
```

You can login to your device's IP address on port 10000 to use the web interface: <https://odroid-ip:10000>. However after you log-in (with any system user with sudo access) you will likely be unimpressed by the default interface. It looks like it is out of the 1990's.



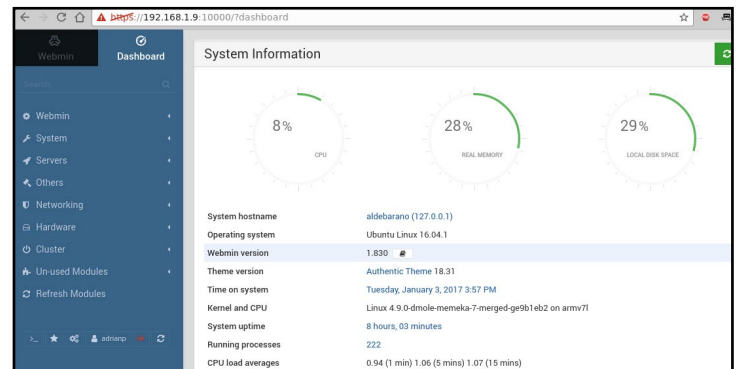
Stock Webmin interface

The first thing we must do is beautify it via a theme. The best-looking theme is called "Authentic Theme", which brings in a lot of features, including being mobile-friendly. You can get the latest version from <http://bit.ly/2jf468e> and install it using the following command:

```
$ wget http://bit.ly/2jRfecC
```

Navigate inside Webmin to "Webmin Configuration -> Webmin Themes -> Install themes -> From uploaded file" and select the newly downloaded theme. After a short wait and refresh later, you will be presented with the page shown next column.

You can explore Webmin's features by using the search tool in the interface. Note that you can install third-party modules available from <http://bit.ly/2jf6KLd>. There is one thing you should change as soon as possible: by default webmin has a background process that monitors disk temperature, and for me it caused my disks to wake up every 5 minutes. After a



NAS - Webmin with Authentic theme

laborious search I found a solution at <http://bit.ly/2kyzXBv>. Simply add the line "collect\_notemp=1" to `/etc/webmin/system-status/config`, and restart the webmin process.

## Mounting disks

First, you will need to decide if you are going to use RAID or LVM with your disks, and which filesystem you would use. I will not go into details about setting RAID/LVM because the subject has been discussed in previous articles. However, even without having a lot of expertise, you can use Webmin to do the heavy lifting for you and use the built-in help to learn more. Webmin will prompt you to install any missing dependencies. Once you have your partitions ready, you can start mounting them.

The traditional method of mounting is to use `/etc/fstab` (and Webmin has a comprehensive module to handle that as well), but you may run into problems if you start your system with the disk not attached (systemd likes to wait around for the disk). I prefer to use `autofs`, which mounts disks (local or network-based) on demand and unmounts them when not in use. Unfortunately, it's not managed by webmin, so you will need to use the shell:

```
$ sudo apt-get install autofs
```

You will need to edit `/etc/auto.master` and add a mount entry for your disk, specifying the base directory and its configuration file:

```
$ sudo vi /etc/auto.master
# add at the end your mountpoint
/media/yourdisk /etc/auto.yourdisk --timeout 20
```

In the command above, replace your disk with the path you want to use. Next, edit this configuration file and add your partitions and their mount parameters, using the command "blkid" to find the correct UUID for the disk:

```
$ sudo blkid
```

```
$ sudo vi /etc/auto.yourdisk
xfs-partition -fstype=xfs,dev,exec,suid,noatime
:UUID=9d2d675d-cb08-45b2-b222-c981a8d00c06
```

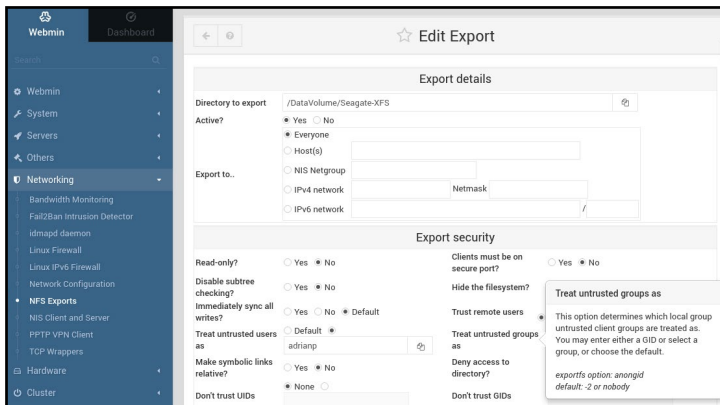
Restart autofs, and when you access /media/yourdisk/xfs-partition your partition will be mounted automatically:

```
$ sudo service autofs restart
```

You will need to take care of the mount parameters because each filesystem has their own parameters and they might impact performance. For instance, without activating the parameter `big_writes` on NTFS, you will get very poor performance. If in doubt, you can cheat and use Webmin to create entries in `/etc/fstab`, test them to ensure the parameters are ok, and migrate them to autofs's layout later (that's what I used). To force automounted disks to be unmounted, you can simply restart the autofs service.

## Set up file shares

To set up Samba shares (and also install Transmission for torrent downloading) you can follow the guide "Designing

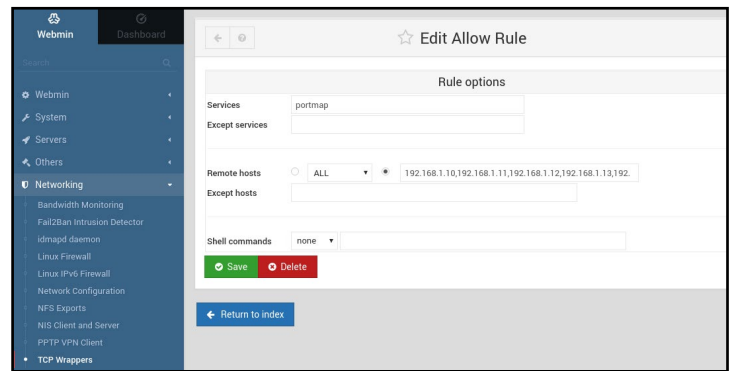


### Create NFS share

your own seedbox" featured in Odroid Magazine <http://bit.ly/2j3xpaK>. You also can also experiment with Webmin's interface and easily create shares and users with a few clicks. For example, Figure 3 shows the "Create NFS share" dialog. Clicking on the form items shows a contextual help menu that explains well, what that item does. This can help you with things you might not be familiar with.

When creating Samba/NFS shares, take security into consideration from the start. Samba authenticates by userid and password, but NFS authenticates users only by IP. If you know which hosts in your network may have access to specific shares, specify it in the configuration. For example, an NFS share might be exported to "Everyone", but access can still be limited with iptables or `/etc/hosts.allow` and `/etc/hosts.deny` (which are used by TCP Wrappers Webmin module).

Used with its default configuration, Samba will give decent



### /etc/hosts.allow config for NFS to limit access from a few hosts

performance, but with the tweaks below, extracted from the ODROID forums, you should get fewer "pauses" in large file transfers. Add the lines below to the [global] section of your `/etc/samba/smb.conf`:

```
write cache size = 524288
getwd cache = yes
use sendfile = yes
min receivefile size = 16384
```

## Install Owncloud

Owncloud is a personal "cloud" service that lets you share files with people over the Internet. I am not going to go into installation details, because they have been discussed in a previous Magazine article <http://bit.ly/2kgVZpn>, but there are some things I would like to point out.

First of all, the installation is quite simple on Ubuntu 16.04. I used the guide at <http://do.co/2bzxhxG> and was up and running in 10 minutes. If you have a DNS name (e.g. dynamic DNS for your home) you should take the time to get a valid SSL certificate from Let's Encrypt (<http://bit.ly/1qmIXIY>) using steps listed at <http://do.co/2bQpv4M>.

You basically need to install the following prerequisites before installing OwnCloud:

```
$ sudo apt-get install php \
libapache2-mod-php php-mcrypt \
php-mysql php-bz2 php-curl \
php-gd php-imagick php-intl \
php-mbstring php-xml php-zip \
mysql-server apache2
```

Next you install the OwnCloud repository for Ubuntu and refresh the available packages:

```
$ sudo curl https://download.owncloud.org/download/\
repositories/stable/Ubuntu_16.04/Release.key \
| sudo apt-key add -
```

```
$ echo `deb https://download.owncloud.org/download/\
repositories/stable/Ubuntu_16.04/ '/' \
| sudo tee /etc/apt/sources.list.d/owncloud.list
$ sudo apt-get update
```

Finally, you can install OwnCloud:

```
$ sudo apt-get install owncloud
$ sudo systemctl reload apache2
```

You will also need to create a database user for OwnCloud:

```
$ sudo mysql -u root
> CREATE DATABASE owncloud;
> GRANT ALL ON owncloud.* to 'owncloud'@'localhost'
IDENTIFIED BY 'databasePassword';
> FLUSH PRIVILEGES;
> exit
```

After all this work, you can login through the web interface at <https://<odroid-ip>/owncloud> and finish the installation. Since the point of OwnCloud is to be accessible to the Internet, you should take some time to harden your installation, as described at <http://bit.ly/2jOTe1F>. In my case, I want to run the OwnCloud service on a different port (so that external users don't have access to my internal sites), to set iptables rules to allow access only from my country (based on geo-ip data), and set up fail2ban to protect me against automated password guesses.

In order to run the OwnCloud virtual host on a different port you need to make a few adjustments to your apache config:

```
$ sudo cp /etc/apache2/sites-available/default-ssl.conf \
/etc/apache2/sites-available/owncloud.conf
$ cd /etc/apache2/sites-available
$ sudo ln -s ../sites-available/owncloud.conf \
020-owncloud.conf
```

Next, edit `/etc/apache2/sites-available/owncloud.conf` and make the following changes:

**Add “Listen 8443” as the first row**  
**Change the VirtualHost definition to use port 8443 instead of 443 (<VirtualHost \_default\_:8443>)**  
**Change DocumentRoot to point to your owncloud installation**  
“DocumentRoot /var/www/owncloud”

When done, you can restart the Apache daemon, and you should be able to access only your OwnCloud instance on <https://<odroid-ip>:8443/>.

To get started with GeoIP firewall rules, you'll need to have the kernel sources (or kernel headers) available. Next, you can install the extra iptables modules with the following command:

```
$ sudo apt-get install \
xtables-addons-dkms \
xtables-addons-common \
xtables-addons-source
```

The dkms package will fail to install cleanly because some of the modules fail to compile against kernel 4.9. You can disable the failed modules and recompile the rest by setting the following settings to “n” instead of “m” in the file `/var/lib/dkms/xtables-addons/2.10/build/mconfig`:

```
$ sudo vi /var/lib/dkms/xtables-addons/2.10/build/
mconfig
build_ACCOUNT=n
build_LOGMARK=n
build_SYSRQ=n
build_pknock=n
build_psd=n
```

Next you will need to manually compile the rest:

```
$ cd /var/lib/dkms/xtables-addons/2.10/build/
$ sudo autoconf
$ sudo ./configure
$ sudo make
$ sudo make install
```

Before using the geoip module, you will need to initialize the geoip database (the prefix to country mapping). You may need to repeat this step from time to time to benefit from the latest data:

```
$ sudo apt-get install libtext-csv-xs-perl
$ sudo mkdir /usr/share/xt_geoip
$ sudo /usr/lib/xtables-addons/xt_geoip_dl
$ sudo /usr/lib/xtables-addons/xt_geoip_build -D /
usr/share/xt_geoip /root/GeoIPCountryWhois.csv
```

All that is left to do now is to create and test the iptables rules to allow only traffic that you want to reach your owncloud setup. An example rule looks like this:

```
$ sudo iptables -N geo-owncloud
$ sudo iptables -A INPUT -p tcp -m tcp --dport 8443
-j geo-owncloud
$ sudo iptables -A geo-owncloud -s 192.168.1.0/24 -j
ACCEPT
```

```
$ sudo iptables -A geo-owncloud -m geoip ! --src-cc
RO -j DROP
```

Do not forget to save your rules and apply them at startup (either with iptables-save or with webmin). More details about geoip can be found at <http://bit.ly/2jnwUJD>.

```
adrianp@aldebaran:~$ sudo tail -f /var/log/fail2ban.log
2017-01-03 07:53:19.143 fail2ban.actions [1466]: INFO Set banTime = 600
2017-01-03 07:53:19.145 fail2ban.filter [1466]: INFO Set maxRetry = 5
2017-01-03 07:53:19.194 fail2ban.jail [1466]: INFO Jail 'sshd' started
2017-01-03 07:53:19.255 fail2ban.jail [1466]: INFO Jail 'owncloud' started
2017-01-04 14:38:45.757 fail2ban.filter [1466]: INFO [owncloud] Found 172.22.22.2
2017-01-04 14:38:55.097 fail2ban.filter [1466]: INFO [owncloud] Found 172.22.22.2
2017-01-04 14:39:02.457 fail2ban.filter [1466]: INFO [owncloud] Found 172.22.22.2
2017-01-04 14:39:07.463 fail2ban.filter [1466]: INFO [owncloud] Found 172.22.22.2
2017-01-04 14:43:49.032 fail2ban.filter [1466]: INFO [owncloud] Found 172.22.22.2
2017-01-04 14:43:49.407 fail2ban.actions [1466]: NOTICE [owncloud] Ban 172.22.22.2
```

## Fail2Ban doing its job on failed logins

Configuring fail2ban is not very complicated once you follow the tutorial at <http://bit.ly/2kipXxn>. Remember to install fail2ban first (and test it with some false credentials):

```
$ sudo apt-get install fail2ban
```

Since we have added a special port for owncloud, we will need to tweak fail2ban's configuration to account for that. Edit `/etc/fail2ban/jail.local` and append "port 8443" to the port line and restart fail2ban:

```
$ sudo vi /etc/fail2ban/jail.local
port = http,https,8443
$ sudo service fail2ban restart
```

To manually lift the ban for a blacklisted IP address you can run the following command:

```
$ sudo fail2ban-client set owncloud unbanip
172.22.22.2
```

## Emulating HMP

HMP, which stands for Heterogeneous Multi-Processing Device, is an optional enhancement that will give you a snappier experience. The ODROID-XU4 comes with two types of CPU cores: 4 little cores that are low power and are best suited for background tasks and 4 big cores which are designated for more powerful tasks. The official 3.10 kernel comes with a "magic" scheduler from Samsung which knows the processor's true power, and can switch tasks from the little cores to the big cores when load is high. However, Samsung's patches have been rejected for the mainline kernel (<http://bit.ly/2iTbUhr>) which means that the kernel will randomly assign tasks to different CPU and you will get inconsistent performance. There are plans to add this functionality for big.Little systems (<http://bit.ly/2kiveoJ>), but there is nothing mature right now. We will need to emulate the functionality of HMP somehow. We can use cgroups as noted in <http://bit.ly/2jP6KIU>.

"cgroups" is a feature of modern kernels that allows allocation of resources for various processes. In our case we will need the "cpuset" cgroup to create a "littlecores" and a "bigcores" group. Each group will force processes to run on specific cores by setting the affinity. So, littlecores will have cpus 0-3 and bigcores 4-7. Fortunately, creating the cgroups is easy:

```
# mkdir -p /sys/fs/cgroup/cpuset/littlecores \
/sys/fs/cgroup/cpuset/bigcores
# echo "0-3" > /sys/fs/cgroup/cpuset/\
littlecores/cpuset.cpus
# echo "0"> /sys/fs/cgroup/cpuset/\
littlecores/cpuset.mems
# chmod -R 777 /sys/fs/cgroup/cpuset/\
littlecores
# echo "4-7"> /sys/fs/cgroup/cpuset/\
bigcores/cpuset.cpus
# echo "0"> /sys/fs/cgroup/cpuset/\
bigcores/cpuset.mems
# chmod -R 777 /sys/fs/cgroup/cpuset/\
bigcores
```

Unfortunately, the commands will only last until the next reboot. So, let us create a service to set them as early as possible on boot:

```
$ sudo wget -O /etc/systemd/system/cpuset.service
https://raw.githubusercontent.com/mad-ady/\
odroid-ODROID-XU4-optimizations/master/cpuset.service
$ sudo systemctl enable cpuset
$ sudo systemctl start cpuset
```

At this point, the cgroups are created, but they are not actively used by anyone. To manually start a process in a specific cgroup, you can use cgexec:

```
cgexec -g cpuset:bigcores sysbench --test=cpu \
--cpu-max-prime=100000 --num-threads=8 run
```

## 8 sysbench threads are forced to run on 4 specific cores

```
$ sudo apt-get install cgroup-tools
$ cgexec -g cpuset:bigcores sysbench --test=cpu \
--cpu-max-prime=100000 --num-threads=8 run
```

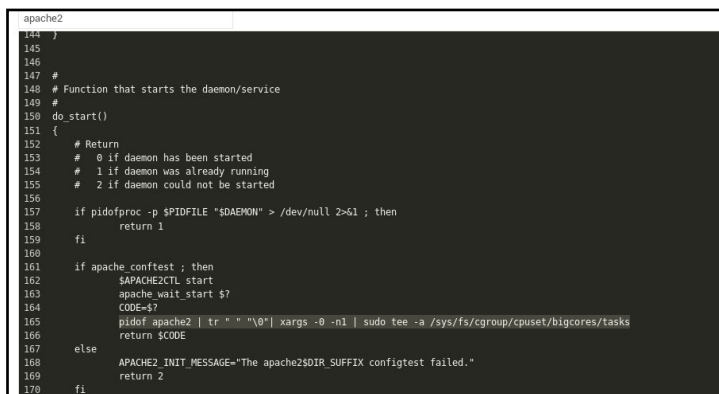
We are only halfway there. We will need to tell specific processes to run on the little cores and the others to run on the big cores. This is where you need to make a list and decide what you want. Start with a list of active services from webmin (System -> Bootup and Shutdown) and disable anything

you will not be using. In my case I have disabled the following services: ModemManager, NetworkManager-wait-online, NetworkManager, accounts-daemon, alsa-restore, alsa-state, appport, appport-forward.socket, bluetooth, cups-browsed, cups.path, cups.service, cups.socket, lightdm, lxc-net, lxc, lxcfs, plymouth\*, rsync, saned, speech-dispatcher and whoopsie.

You will need to edit the startup scripts for the services you want and have them add their PID to the correct cgroup. Once the main process (and its children) are part of the correct cgroup, any new children will inherit the cgroup. My plan was to add things like MySQL, Apache, Samba, NFS and even Webmin to the big group and things like SSH (and all my shell activity), cron, Munin, and Transmission to the little group. This allows processes that are involved in the NAS functionality to be snappy, while other tasks can happily run on the little cores. If you are also using the X11 GUI, you might want to add lightdm to the “bigcores” group as well.

There are two types of startup scripts - systemd native scripts and legacy sys-v (/etc/init.d/). When editing a systemd script (for example nfs-mountd.service) you will need to add something like this to the [Service] section:

```
ExecStartPost=-/bin/sh -c 'echo $MAINPID | tee -a /
sys/fs/cgroup/cpuset/bigcores/tasks'
```



```
apache2
[Unit]
144 }
145
146
147 #
148 # Function that starts the daemon/service
149 #
150 do_start()
151 {
152     # Return
153     # 0 if daemon has been started
154     # 1 if daemon was already running
155     # 2 if daemon could not be started
156
157     if pidofproc -p $PIDFILE "$DAEMON" > /dev/null 2>&1; then
158         return 1
159     fi
160
161     if apache_configtest ; then
162         $APACHECTL start
163         apache_wait_start $?
164         CODE=$?
165         pidof apache2 | tr " " "\0" | xargs -0 -n1 | sudo tee -a /sys/fs/cgroup/cpuset/bigcores/tasks
166         return $CODE
167     else
168         APACHE2_INIT_MESSAGE="The apache2$DIR_SUFFIX configtest failed."
169         return 2
170     fi
171 }
```

### Changing apache's startup configuration

When editing an older sys-v script, it is trickier. You will need to find the start function, extract the PID(s) of the newly started process and add it to the tasks list. Below is an example for changing the apache startup script:

```
pidof apache2 | tr " " "\0" | xargs -0 -n1 | sudo tee
-a /sys/fs/cgroup/cpuset/bigcores/tasks
```

Take care to restart each service after changing it and make sure to check that the process PID is in the correct cpuset tasks file. Do a full system reboot and check again after restart. If this sounds too complicated and you are using kernel 4.9, there is a way to cheat and run all the tasks on the big cores. You can

simply set systemd's affinity, and all of its children processes will inherit it. The affinity can be controlled by the CPUAffinity parameter in /etc/systemd/system.conf, but keep in mind you'll be wasting CPU cores.

## Disk longevity

In order to prolong the life of your disk(s), you may want to spin them down after a period of inactivity. If you are using SSDs, you can skip this section because it only applies to old mechanical disks. Disks may receive a “stop” command to spin down either from an internal controller, from the USB-SATA bridge or directly from the operating system. However, sometimes the controllers are not tuned correctly and a stop command never arrives. This causes the disk to keep spinning which generates a lot of heat and can cause the drive to fail sooner than normal.

The normal way to handle this is to tell the disk to spin down after a period of inactivity, which can be done with hdparm:

```
$ sudo apt-get install sdparm hdparm
```

To manually set the disk to sleep after 10 minutes of inactivity, you can run the following command:

```
$ sudo hdparm -S 120 /dev/sda
```

If you get errors (like “bad/missing sense data”), hdparm might not help you for that disk.

To handle disk mobility, it would be better to let udev run the command after a disk has been plugged in. Since different disks might have different roles, and you may want different sleep timers (e.g. one disk is for backups and should sleep sooner, other is active and should sleep later), I decided on setting the UDEV rule based on the disk's serial number. You can get this serial number by looking in dmesg when plugging in a disk:

```
[1885221.800435] usb 4-1.3: Product: My Passport 0730
[1885221.800436] usb 4-1.3: Manufacturer: Western
Digital
[1885221.800437] usb 4-1.3: SerialNumber:
575844314141305636323937
```

To set up the rule, create a file like this and reload udev:

```
$ sudo vi /etc/udev/rules.d/90-disk.rules
ACTION=="add", ENV{DEVNAME}=="/dev/
sd?", SUBSYSTEM=="block", ENV{ID_SERIAL_SHO
RT}=="575844314141305636323937", RUN+="/sbin/hdparm
-S 120 $env{DEVNAME}"
```

```
$ sudo udevadm control -R
```

If the `hdparm` cannot put your disk to sleep, then try other alternatives like `sdparm`, which can send a SCSI command to your disk, like ordering it to shut down in that instant:

```
$ sudo sdparm -C stop /dev/sda
```

There are tools like `hd-idle` (<http://bit.ly/2j3zWSk>) or periodic scripts you can run to put your disk to sleep. In my case they did not work, but make sure to try them manually before settling on a solution. Here is a manual script which checks a disk (identified by a partition's UUID) for activity in a 10s window, and if there was no disk activity (data transferred), it uses `sdparm` to stop the disk. You can run it via cron:

```
$ sudo wget -O /usr/local/bin/hdd-idle.sh http://bit.ly/2k6LK7Y
$ sudo chmod a+x /usr/local/bin/hdd-idle.sh
$ sudo /usr/local/bin/hdd-idle.sh "4283-E975"
```

You must be aware that there are tools and services which will wake up your disk periodically, even if no data is transferred. Such tools include `smartctl` (from `smartmontools`) and `smartd`. The `smartd` service periodically checks disk health, and if not correctly configured, it may keep your disk up needlessly. You can consult the thread at <http://bit.ly/2kh6b17> in case you do not know what is keeping your disk awake. You should be able to infer the disk's state by running this command:

```
$ sudo smartctl -d sat -n standby -a /dev/sda
```

If it exits with an error, your disk is still in standby and should have been spun-down.

## Flash disk performance

One more thing to keep in mind when using flash storage (eMMC or SSD) is that they need periodic trimming to maintain their speed. Basically, in order to write to a storage block, you need to erase it first and this takes longer than writing to it. Normal filesystems do not do this erase when deleting data, so after a while disk performance drops significantly. To "revive" the disk, the trim operation informs the disk controller to erase all empty blocks, thus restoring write speeds. The trim operation must be supported by the filesystem and the disk controller. Again, using cron once a week to run `fstrim` can save you from slowdowns in the long term:

```
$ sudo crontab -e
#trim the eMMC once a week
15 0 0 * * /sbin/fstrim / >/dev/null 2>&1
```

## Governor

Performance and heat are also directly dependent on what governor you are using for the CPU. Keeping "performance" on gets you top performance, but also generates a lot of heat. In my tests, the best combination was a modified "on-demand" governor based on the recommendations at <http://bit.ly/2jfaDjw>. To enable it, make sure you select governor = `ondemand` in `/media/boot/boot.ini`, and set the rest of the parameters inside `/etc/rc.local` (test out the commands before). The commands below work for a 4.9 kernel and may differ for the 3.10 kernel:

```
$ sudo vi /etc/rc.local
echo 1 > /sys/devices/system/cpu/cpufreq/ondemand/
io_is_busy
echo 10 > /sys/devices/system/cpu/cpufreq/ondemand/
sampling_down_factor
echo 80 > /sys/devices/system/cpu/cpufreq/ondemand/
up_threshold
```

With the setting above, the CPU will ramp up frequency sooner and will consider IO usage as CPU, making IO intensive tasks influence the CPU frequency. This allows you to have great performance when needed and low heat when idle. In my usage, the little cores idle around 300MHz while the big cores idle at 200MHz.

## Network performance - MTU

If you have a Gigabit network with proper cables, you can increase the MTU (Maximum Transmission Unit) on the ODROID-XU4's onboard network. This will allow it to send larger packets which have less overhead and generate fewer interrupts on the receiving end. However, to benefit from it, you will need to have network devices (switches/routers) and end devices which support Jumbo frames. Ideally, Jumbo frames would need to be enabled on all network devices in your LAN, otherwise you might see dropped traffic or even devices unable to send large traffic to each other. For example, SSH works because it uses small packets, but getting a web page or transferring a file stalls the connection. If you do decide to enable jumbo frames, the ODROID-XU4's magic MTU value is 6975 (<http://bit.ly/2jP9zDI>). You can enable it on the ODROID-XU4 inside `/etc/rc.local`:

```
$ sudo vi /etc/rc.local
#MTU
/sbin/ifconfig eth0 mtu 6975 up
```

## Network performance - interrupts

Another big cause of performance drops at a high traffic

rate is the fact that the interrupt handler on the ODROID runs on CPU0 by default, which is a little core. You can get another 150+Mbps in network speed if you move the network card's interrupt handler to a big core. I have setup a service that moves all USB interrupt handlers to the big cores. You can download it and install it with the following commands:

```
$ sudo wget -O /etc/systemd/system/affinity.service \
https://raw.githubusercontent.com/mad-ady/\
odroid-ODROID-XU4-optimizations/master/affinity.ser-
vice
$ sudo systemctl enable affinity
$ sudo systemctl start affinity
```

You can check that it worked by looking inside `/proc/interrupts` and seeing values in the 6th column (corresponding to the 4th CPU) for the USB bus. Also, `iperf` will report faster download speeds. There are other possible improvements you can do, but in my use case I saw no difference in performance. You can review and discuss them at <http://bit.ly/2k6JOMF>.

## Fast transfer with sshfs/scp/sftp

Since SSH is a very flexible protocol and supports tunneling and file transfer, it would be wise to use it at full speed. If you attempt a secure copy (scp) transfer on an ODROID-XU4 with the `sshd` process tied to the little cores, you will get about 15MB/s top speed. If you tie the `sshd` process to the big cores you get 40MB/s. If you are feeling adventurous and do not mind sacrificing some security, you can squeeze 50MB/s by lowering the encryption algorithm used. I did that by starting a different `sshd` instance (on port 2222) with different settings:

```
$ sudo wget -O /etc/systemd/system/ssh-big.service \
https://raw.githubusercontent.com/mad-ady/\
odroid-xu4-optimizations/master/ssh-big.service
$ sudo wget -O /etc/ssh/sshd_config_big \
https://raw.githubusercontent.com/mad-ady/\
odroid-xu4-optimizations/master/sshd_config_big
$ sudo systemctl enable ssh-big
$ sudo systemctl start ssh-big
```

To mount or transfer a file using this new ssh service you will need to specifically specify the cipher (since it is disabled by default because it is considered weak). You can do so in an entry in `~/.ssh/config` on the client:

```
Host odroid-big
  Hostname odroid-ip
  Port 2222
  Ciphers arcfour
  Compression no
```

To transfer files you can simply use the following command:

```
$ scp bigfile odroid-big:/remote/path
```

## Tune systemd timeouts

It can be irritating to wait around for `systemd` to finish waiting for something that will never finish. You can tweak `systemd`'s timeouts by modifying the global timeout settings in `/etc/systemd/system.conf`:

```
DefaultTimeoutStartSec=20s
DefaultTimeoutStopSec=10s
```

Note that some services (like networking) set explicit timeouts and you'll need to change those as well:

```
$ sudo vi /etc/systemd/system/network-online.target.
wants/\
networking.service
TimeoutStartSec=30sec
```

## Performance

Here are some performance metrics you can expect with the tweaks above and a Gigabit network. The client is an ODROID-C2 running Ubuntu 16.04, while the server is the ODROID-XU4. The download and upload directions are relative to the ODROID-XU4. The disk attached to the ODROID-XU4 has a write speed of 110MB/s. File transfers transferred an 8GB file filled with zeros (`dd if=/dev/zero of=zero bs=1M count=8000 conv=fsync`).

Test	Command	Download	Upload
iperf	<b>ODROID-XU4</b> <code>iperf -s -i 10 -N</code> <b>c2</b> <code>iperf -c 192.168.1.9 -t 30 -i 10 -r</code>	925Mbps (115MB/s)	820Mbps (102MB/s)
samba	<b>c2</b> <code>sudo mount -t cifs -o username=odroid,password=odroid //odroid-ip/Seagate-XFS /media/share</code> <b>c2</b> <code>cp /media/share/zero /dev/null</code> <b>c2</b> <code>cp zero /media/share/zero.1</code>	79MB/s (98MB/s with a PC client)	46MB/s
nfs	<b>c2</b> <code>sudo mount -t nfs -o rw odroid-ip:/nfs/Seagate-XFS /media/share</code> <b>c2</b> <code>cp /media/share/zero /dev/null</code> <b>c2</b> <code>cp zero /media/share/zero.1</code>	102MB/s (111MB/s with a PC client)	56MB/s
scp	<b>c2</b> <code>scp odroid-big:/media/seagate/xfs/zero /dev/null</code> <b>c2</b> <code>scp zero odroid-big:/media/seagate/xfs/zero.1</code>	38MB/s (52MB/s with a PC client)	31MB/s (52MB/s with a PC client)

Please note that part of the performance depends on your client as well. I was able to get better performance with a Linux PC than with the ODROID-C2 as a client.

Feel free to discuss possible optimizations on the support thread at <http://bit.ly/2j3Ddke>.

# THE AMAZING ODROID-VU8

A PORTABLE ALL-IN-ONE ODROID TOUCHSCREEN TABLET WITH 1024 X 768 RESOLUTION

edited by Rob Roy (@robroy)



Figure 1 - The new ODROID-VU8 delivers a crisp, clear image

The ODROID-VU8 is a new 8-inch multi-touch screen for the ODROID-C2 and C1+ that is available for purchase at <http://bit.ly/2k8bML5> for USD\$90. It gives users the ability to create all-in-one, integrated projects such as tablets, game consoles, infotainment systems and embedded systems. Just plug a 5V/4A DC adapter into the LCD Shell and power up the C2 and C1+ and display with a single cord. This high-quality wide viewing angle LCD touchscreen is specifically designed to work with both Android and Linux on the ODROID-C1+ and ODROID-C2. More detailed information may be found at <http://bit.ly/2iY960M>.

## Specifications

- 8-inch TFT-LCD
- Screen Resolution: 1024x768 pixels (4:3 ratio)
- 10 finger capacitive touch input (USB ID 18D1:4E12)
- Back-light brightness control with ODROID GPIO PWM
- Max Power consumption : 1.1A/5Volt (Only LCD and display controller)
- Viewing angle : Left 75, Right 75, Up 75, Down 75 degree
- Screen Dimensions : 189 x 149 x 29 mm
- Viewable screen size : 162 x121.5 mm (active area)

## Hardware components

- A. Assembled 8inch TFT LCD + 10 points multi touch screen
- B. Plastic bottom case
- C. Converter board
- D. HDMI dual gender board
- E. 8 x 3.5mm screws
- F. 2port jumper cable
- G. Micro-to-Micro USB Cable (approx. 8cm)
- H. Micro-to-TypeA USB Cable (approx. 20cm)
- I. 5V/4A Power supply

Figure 2 - ODROID-VU8 kit



Figure 3 - Ethernet port and USB host ports x 2



Figure 4 - Vent hole and cutting line for the 40pin GPIO port



Figure 5 - 5V/4A PSU input



## Assembly

Take care while assembling the ODROID-VU8.



Figure 6 - Plug the HDMI dual gender board to the converter board. It is easier to fit the HDMI dual gender board with this angle

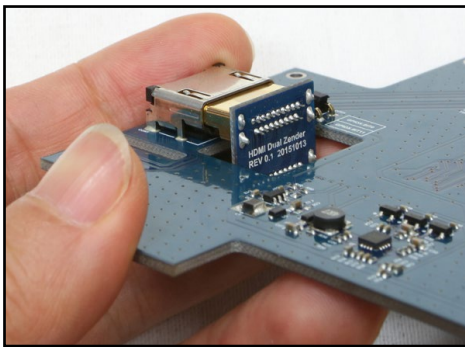


Figure 7 - Plug it tight and double-check the connector

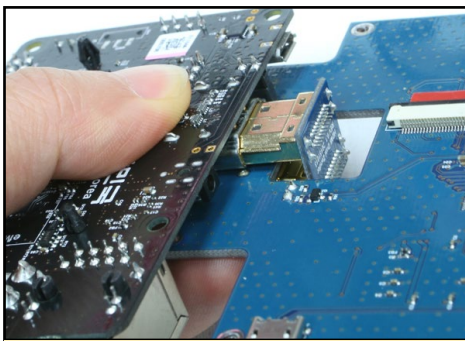


Figure 8 - Plug the ODROID-C2 / C1+ to the other connector on the HDMI dual gender board

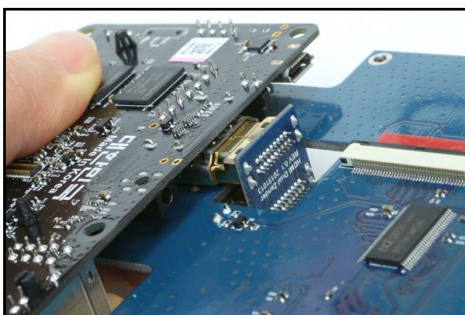


Figure 9 - Plug it tight for stable HDMI connectivity

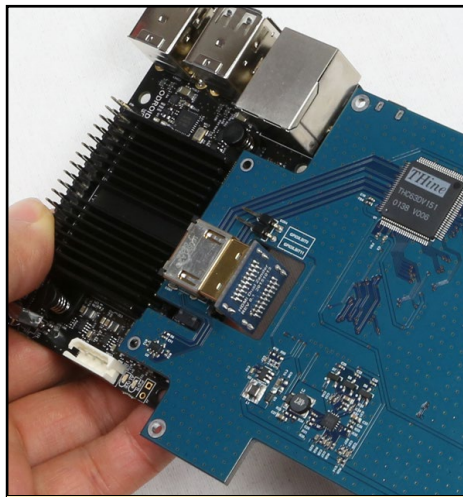


Figure 10 - The HDMI signal is connected

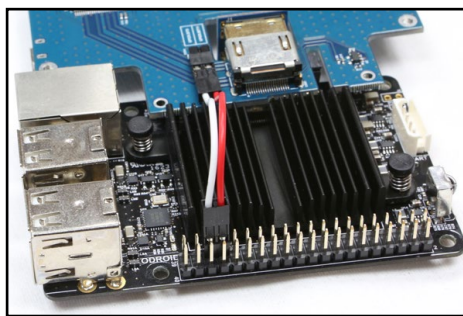


Figure 11 - Plug the jumper cable to the GPIO pin #33 and #35, and plug the other side of the cable to the converter board

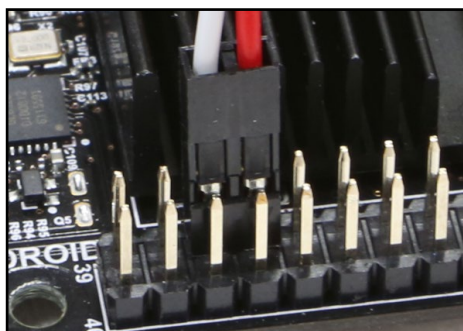


Figure 12 - Check the placement of the pins in relation to the jumper cable

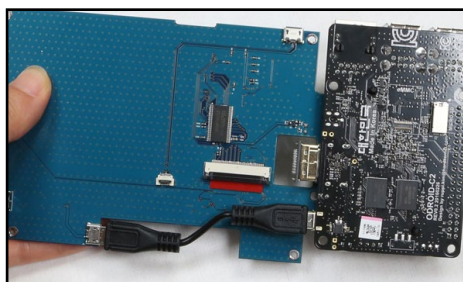


Figure 13 - Plug the micro to micro USB cable, which gives the C2 or C1+ the power via the converter board

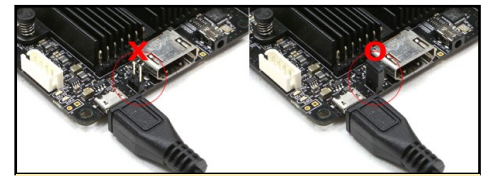


Figure 14 - The two jumper pins must be plugged to input the power via the microUSB connector

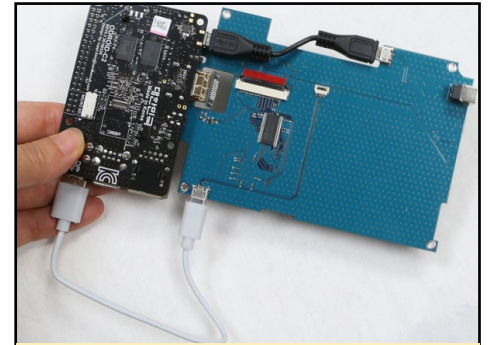


Figure 15 - Plug the standard to micro USB cable for touch signal between the C2 / C1+ and the converter board

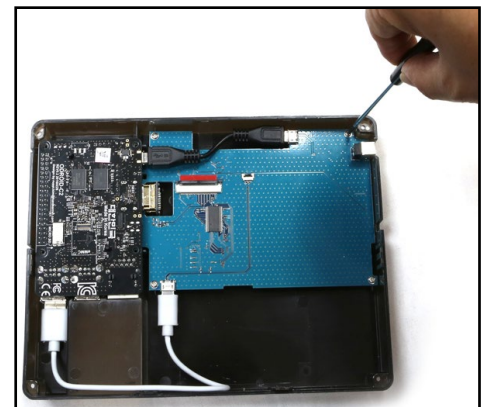


Figure 16 - Place the assembled board on the bottom case and put the screws

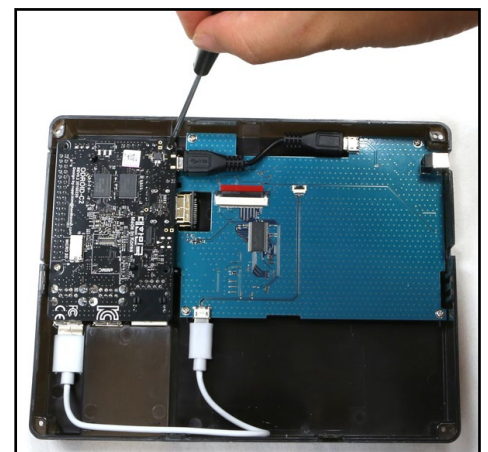
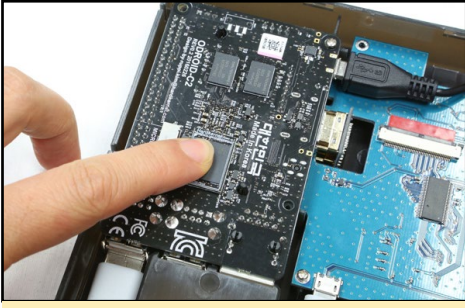
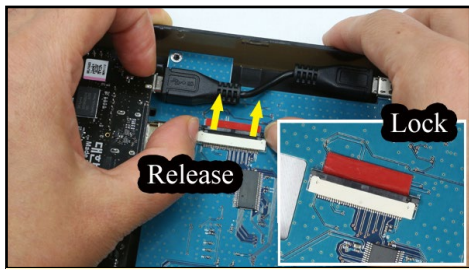


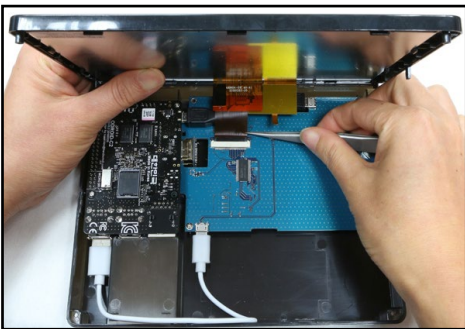
Figure 17 - Put screws on the screw hole of the C2 / C1+ board



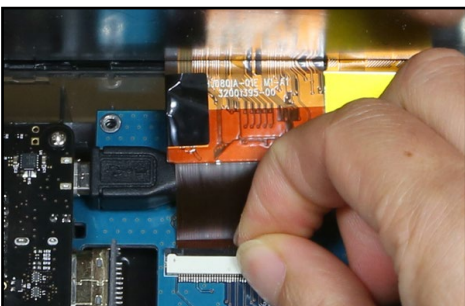
**Figure 18 - Install the eMMC Module / MicroSD card. It is not possible to change the eMMC module or microSD card after full assembly**



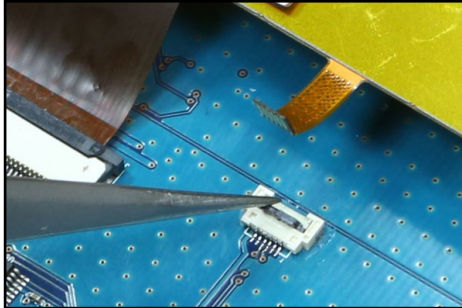
**Figure 19 - Slide out open the 40pin connector slot**



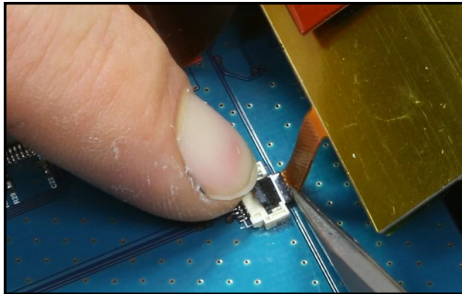
**Figure 20 - Insert the 40pin FPCB into the connector slot. The LCD signal goes to the converter board via this FPCB**



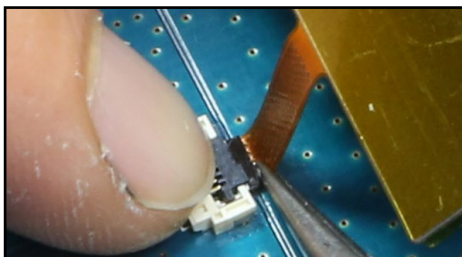
**Figure 21 - Lock the slot**



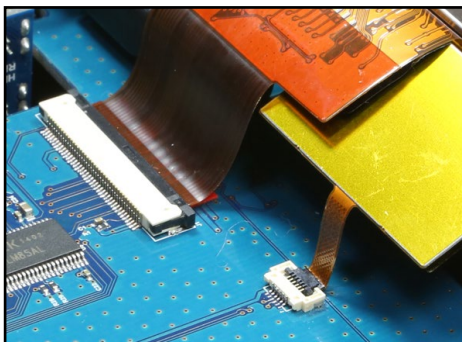
**Figure 22 - Open the 6pin connector. The door must open 90 degrees**



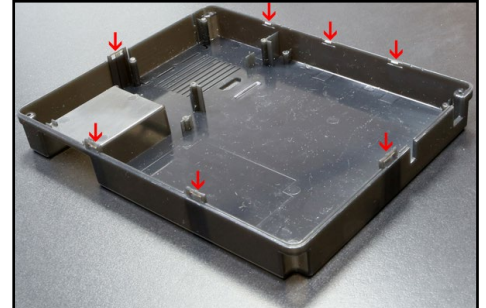
**Figure 23 - Insert the 6pin FPCB into the connector carefully with tweezers. The dark part must be inserted correctly. The touchscreen signal goes to the converter board via this 6pin FPCB. This is one of the most difficult steps. If the 6pin FPCB is not inserted correct, it will loosen and the touch signal will be disconnected**



**Figure 24 - Close the door**



**Figure 25 - Check the 40pin and 6pin FPCBs connections**



**Figure 26 - Check the hooks and locking holes on the bottom case before you close the front piece**



**Figure 27 - Close the case. Make sure not to damage the fragile FPCBs and system inside the case**



**Figure 28 - Press the side of the bottom case together reasonably hard to hook the locking parts**



**Figure 29 - Insert and tighten the screws on 4 corner side of the screw hole**

## Software

In Android, select “ODROID-VU8” in the ODROID Utility while attached to a normal HDMI monitor.

In Linux, open boot.ini with a text editor and uncomment these lines:

```
setenv m "1024x768p60hz"
setenv vout "dvi" # C2
setenv vout_mode "dvi" # C1/C0
```

# BUILDROOT

NOW AVAILABLE FOR  
ODROID-C0/CI/CI+

by @Brian.K

# BuildRoot

Making Embedded Linux Easy

**B**uildroot is a handy tool to make the process of building Linux simpler and more efficient. The tool is not only easy to use but, also supports a variety of advanced options such as cross-compilation and building in additional software with packages. Buildroot is heavily menu-driven, which allows you to quickly get things setup and compiling with 15 minutes! This guide will walk you through the simple steps needed to use Buildroot for the ODROID-C1+ and ODROID-C0.

Before you can use images generated by Buildroot on your ODROID-C1+ and ODROID-C0, you have to prepare the SDCard or eMMC. First, download the ODROID-C1+ / C0 buildroot source code from GitHub:

```
$ git clone --depth 1\
  https://github.com/hardkernel/
  buildroot.git
```

Additionally, we are going to need a cross-compiler so the compiled boot-loader and kernel can run on the ODROID:

```
$ wget\ http://releases.linaro.
  org/archive/14.09/components/\
  toolchain/binaries/gcc-linaro-
  arm-none-eabi-4.9-2014.09_linux.
  tar.xz
$ sudo tar\
```

```
Jxvf gcc-linaro-arm-none-ea-
bi-4.9-2014.09_linux.tar.xz \
-C /opt/toolchains/
$ export ARCH=arm
$ export CROSS_COMPILE=arm-none-
eabi-
$ export PATH=/opt/tool-
chains/gcc-linaro-arm-none-ea-
bi-4.9-2014.09_linux/bin/:$PATH
```

Next, build the BuildRoot executable:

```
$ cd buildroot
$ make odroidc1_defconfig
```

After a successful build, you can edit the build options using the following command:

```
$ make menuconfig
```

Finally, we will compile all the files and build the rootfs image. Note that you will need to have Internet access, since Buildroot will download any source code for additionally included packages:

```
$ make
```

After building, you should have the following file tree:

```
output/images/
+-- Image
```

```
+-- boot.ini [1]
+-- boot.vfat
+-- meson8b_odroidc.dtb
+-- rootfs.ext2
+-- rootfs.ext4
+-- rootfs.tar
+-- sdc card.img
+-- bl1.bin.hardkernel
+-- sd_fusing.sh
`-- u-boot.bin
```

The boot.ini file is the configuration file used for the ODROID-C1+ / C0 u-boot. Once the build process is finished, you will have an image called “sdc card.img” in the output/images/ directory. Flash the bootable “sdc card.img” onto an SD card or eMMC using “dd” on Linux/OSX, or the Win32 Disk Imager on Windows:

```
$ sudo dd if=output/images/sd-
card.img of=/dev/sdX conv=fsync
```

After flashing, insert the SD card or eMMC into your ODROID, and power it up. Your new system should be ready to use. For comments, questions, and suggestions, please visit the original post at <http://bit.ly/2jqGRsO>.

**SKY FORCE  
RELOADED  
WHAT DO WE LOVE  
ON A SHOOT 'EM UP?  
BULLET HELL!**

by Bruno Doiche

Sometimes we want all the action we can take at once. And for times like this, the Infinite Dreams team has created what I believe to be the most polished shoot 'em up to date for the Android platform. Sky Force Reloaded nears masterpiece status! If you like a challenging game, this one will certainly be promoted to your home page on your Android. You have been warned.



<https://play.google.com/store/apps/details?id=pl.idreams.SkyForceReloaded2016>

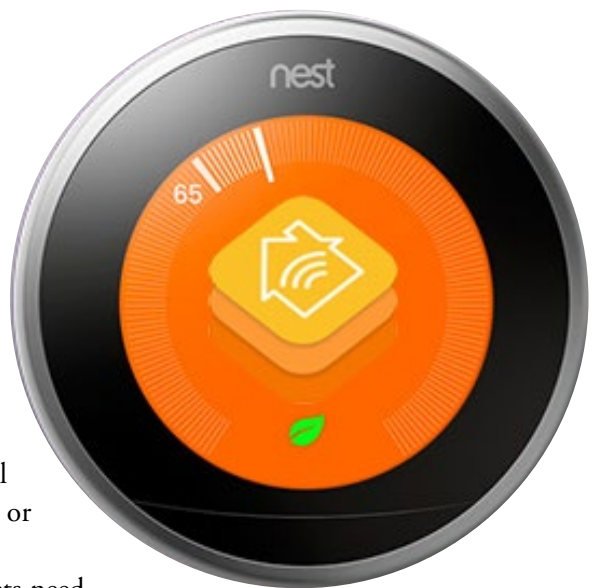


**This game has some amazing graphics, and is a good reason for you to buy a USB joystick if you don't have one already!**

**HOMEBRIDGE  
HOME  
AUTOMATION  
EASILY INTEGRATE YOUR HOME'S  
TECHNOLOGIES**

by @korinel

Home automation offers a great way to add all sorts of functionality to your home in ways both big and small. Whether it's for turning on the lights, adjusting the thermostat, or locking the doors, there are a host of gadgets you can buy to control this technology automatically or from your smartphone



However, a lot of these gadgets need their own apps or tools in order to run, making things complicate if you own a lot of different gadgets with different APIs. That's where HomeBridge comes in. This nifty NodeJS server helps unify your various home automation gadgets through Apple's HomeKit API, allowing your ODROID to help keep things connected between your iPhone and your various smart lights, smart locks, and other IoT devices, including those that Apple normally wouldn't let you connect to a HomeKit system.

I haven't seen instructions specific to getting HomeBridge working on an ODROID-C2, so I thought I'd research them, since it's a great platform for the open source technology. It took a while, but eventually managed to get a clean homebridge build. Most of the informa-

tion came from <http://bit.ly/2jZr2u3>, but some additional steps were necessary to get things running smoothly.

The stumbling block at first was in installing the nodes, as the version available on the arch64 repositories (since our ODROID-C2 is 64-bit) was severely outdated. To do this, create a temporary directory in your ODROID, then run the following commands to prepare your system:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt-get install libavahi-
compat-libdnssd-dev g++ git make
```

Next we build the most recent version of Node.js. Please note that the version number of Node.js may change, so

check for the latest version and modify these instructions as needed:

```
$ sudo curl -sL https://deb.nodesource.com/setup_6.x | sudo
-E bash -
apt-get source nodejs
$ dpkg-source -x nodejs_6.9.4-
1nodesource1~xenial11.dsc
$ cd nodejs-6.9.4
$ dpkg-buildpackage -us -uc
```

The above step will take a long time, so go have a few cups of tea while you wait. Once it has completed, install Node.js:

```
$ cd .. # Here are your local
.deb packages
$ sudo mv *.deb /var/cache/apt/
archives
$ sudo apt-get install nodejs
```

Next let's Install HomeBridge and its dependencies:

```
$ sudo npm install -g --unsafe-
perm homebridge hap-nodejs node-
gyp
$ cd /usr/lib/node_modules/home-
bridge/
$ sudo npm install --unsafe-perm
bignum
$ cd /usr/lib/node_modules/hap-
nodejs/node_modules/mdns
$ sudo node-gyp BUILDTYPE=Release
rebuild
```

Now we'll set up the HomeBridge service, with the respective user and support files:

```
$ sudo adduser --system home-
bridge
$ sudo mkdir /var/lib/homebridge
$ sudo chown homebridge /var/lib/
homebridge
```

To set up the service takes a little bit more work. Download the two files from this link: <http://bit.ly/2dRZSSY>.

Edit `homebridge.service` such that line 11 reads as follows:

```
ExecStart=/usr/bin/homebridge
$HOMEBRIDGE_OPTS
```

Place the file `homebridge` under `/etc/default`, and `homebridge.service` under `/etc/systemd/system`. Instructions for creating the `config.json` and installing additional homebridge modules can be found at <http://bit.ly/1QK07by>.

Next, place your `config.json` in `/var/lib/homebridge` and ensure that the above files are readable by the user "homebridge". Load and start the system service:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable home-
bridge
$ sudo systemctl start homebridge
$ sudo systemctl status home-
bridge # Just a status check
```

Now you're all set and ready to start making plugins for your various home automation devices. If someone could update one of the official repositories to include a more recent aarch64 version of NodeJS, that would simplify the process considerably. If you're using a 32-bit ODROID, such as the C1, then this process should be easier. For comments, questions and suggestions, please visit the original post on the ODROID Forums at <http://bit.ly/2iBbaIZ>.



## TAP 'N' SLASH UNCOMPLICATED SLASHING WITH FAST REFLEXES

by Bruno Doiche

**A**mong the best things to do when playing with your ODROID is to not have to deal with an overly complex game. There are times when you are just lazy enough not to want to open that drawer to get your USB controller, but you still want to play a fun game. So what do you do? Enjoy Tap 'N' Slash, a quick and fun little game that requires balancing quick thinking with your endless desire to slash monsters on dungeons. You won't believe how quickly you will become hooked on this amazing game!



<https://play.google.com/store/apps/details?id=com.invictus.tapnslash>



Don't get too confident and find yourself out of swords!



# LINUX GAMING

## PPSSPP SPEED COMPARISON

by Tobias Schaaf (@meveric)



Currently, it's hard to compare gaming performance between all of the ODROID devices. Games that make use of the boards' capabilities, such as dynamic recompiling, are very rare, especially when it comes to the ODROID-C2. None of the games for the PS1, N64, or NDS require dynamic recompiling for optimization and run really close to the hardware, and so work well on the ODROID-C2. However, there is one exception: PPSSPP, the Playstation Portable emulator. It actually implements an ARM64 dynamic recompiler which allows us to play PSP games on the C2 as well. So I thought it would be time to do a new comparison and see how the C2 fits into the world of emulation. Please note that this is only a comparison on Linux and performance under Android is very different from these tests here.

### PPSSPP settings

PPSSPP has a lot of different options to optimize performance or looks of games. I use a set of default options here to compare the different devices and went from there to see what options could be changed for improvement. These default settings are as follows:

Frameskipping 3  
Auto Frameskip: ON  
Mipmapping: OFF

Lazy texture caching: ON  
Spline/Bezier curves quality: Medium  
Texture coord speedhacks (speedup): ON  
Lower resolution for effects (reduces artifacts): Balanced

In my experience, these settings, combined with a 2xPSP resolution, normally work for most games. Originally, I wanted to compare every major ODROID board (C1, C2, U2/U3, XU3/XU4) but since I started with the ODROID-C1, I soon found that due to the alpha issue on the C1, most games weren't even working at all, and the games that were working had graphical issues. Switching from 32bpp to 24bpp or even 16bpp didn't help either. 16bpp caused an error on start, and the emulator wouldn't even boot up, and 24bpp was showing everything distorted. I haven't tried the PPSSPP Libretro cores on the C1 yet, because I wanted to compare the same emulator and settings over all boards. In the end, I gave up on the ODROID-C1, and only compared all the other boards with each other.

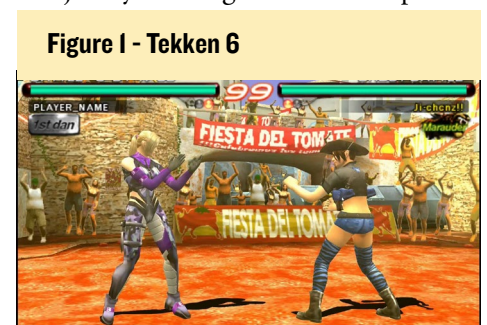
I primarily compared performance results on different resolutions, to check how high you can put the graphics before the game does not play well anymore. You will also often see the term FXAA in my results, which is a post

processing shader that I really like to use on the PPSSPP emulator. I collected some high resolution pictures of different resolutions with and without FXAA for you as a comparison. These pictures were taken on an ODROID-XU3 and show what games can look like in higher resolution and with FXAA shader. You can find the pictures at <http://bit.ly/2jZjIDQ>. The standard resolution of a PSP is 480x272, so a 2xPSP resolution would be 960x544, and so on.

### Tekken 6

For some reason, people really seem to like this game, although I really don't understand why. I find it rather dull and poorly programmed. While other fighting games can run at a stable 60 FPS, Tekken 6 is stuck at 10-15 FPS with lagging and skipping. Still, it is very popular.

The game is programmed in a way that makes it difficult to emulate. Frame rates are always low in this game, and, in my opinion, the graphics and effects do not justify that high demand on perfor-



mance needed to run this game properly. Let's see how this game holds up on the different platforms.

**C2**

Generally the C2 wasn't really able to handle this game well. Videos were a little laggy and the gameplay was even worse.

Video

2xPSP: 20-30 FPS, mostly between 20-25, somewhat laggy

2xPSP+FXAA: 17-23 FPS, mostly around 20 FPS

3xPSP: 18-25 FPS, mostly between 20-23 FPS, somewhat laggy

3xPSP+FXAA: 13-15 FPS, not as laggy as you might expect

In-Game

2xPSP: 1-14 FPS, laggy but playable

2xPSP+FXAA: 09-11 FPS, very laggy

3xPSP: 09-11 FPS, very laggy

3xPSP+FXAA: 08-09 FPS, very laggy

Once again, Tekken 6 proves to be a badly programmed game, which is too hard on the system.

Still, it surprised me that the C2 performance was so bad compared to the performance on the U3, which is 3-4 years older.

**U2/U3**

On the U3, I forgot to change the resolution for videos and only tested 2xPSP resolution for the introduction. The ODROID-U3 was handling this game surprisingly well, which I did not expect.

Video

2xPSP: 50-60 FPS, mostly 60

In-Game

2xPSP: 25-30 FPS, mostly 30 FPS

2xPSP+FXAA: 20-30 FPS, depending on the battlefield

3xPSP: 25-30 FPS, not as stable at 30 FPS

3xPSP+FXAA: 15-20 FPS, mostly 18 FPS which is surprisingly high

These results were much better than I expected. Considering that the ODROID-U3 hardware is from 2012 with only a Mali 400MP2 GPU, the game was running rather stable and fast, and gameplay was actually fluent.

**XU3/XU4**

I was a little bit surprised by the results of the XU3. If you only check the numbers, the XU3 seems even worse than the U3, although the gaming experience was quite nice. For this game, I experimented with disabling frameskipping due to the higher performance of the XU3/XU4. This helped a lot with movies which ran surprisingly well that way, but in-game, frameskipping was required anyway.

Video

2xPSP: full speed without frameskipping, otherwise 25-60 FPS

2xPSP+FXAA: full speed without frameskipping, otherwise 25-60 FPS

3xPSP: full speed without frameskipping, otherwise 25-60 FPS

3xPSP+FXAA: 25-60 FPS with frameskipping

In-Game

2xPSP: 15 FPS, constantly staying directly at 15 FPS with no lags

2xPSP+FXAA: 16-22 FPS, slightly blurry but still lag free

3xPSP: 15-20 FPS, still lag free, except for character close-ups at the start and end of fights

3xPSP+FXAA: 12-15 FPS, surprisingly fluent with only minor lags

I was surprised that even 3xPSP+FXAA is working rather well. It makes the game look very good, but for speed purposes, it's probably best to stick with 2xPSP.

Generally, Tekken 6 has proven that it's a hard game to get working fluently

on any system. Still, both the U2/U3 and XU3/XU4 can handle the game very well, and it shouldn't be hard to find a setting where the game is enjoyable for you.

**Hexyz Force**

I rather like this game, not only because I like RPG games in general, but also because this game is awesome when it comes to requirements. The game runs in only 30 FPS rather than 60 FPS like other games, which means you normally can put more resources in better graphics. Videos and walking around are so well implemented that you can increase the quality with nearly no performance issue at all. Only in fights you may experience slow downs, but since fights are only for a short period, you can probably live with some lags during fights for overall better graphics quality. The pictures at <http://bit.ly/2jZjlDQ> that demonstrate what FXAA and higher resolutions can do, were taken from this game. It also shows nicely how game graphics can be improved quite a bit over the original graphics of the PSP.



Figure 2 - Hexyz Force

**C2**

Video

2xPSP: 25-30 FPS, but mostly full speed

2xPSP+FXAA: 18-30 FPS, mostly around 23-25 FPS slightly laggy

3xPSP: 23-30 FPS, mostly in the higher 20s or even full speed

3xPSP+FXAA: 14-20 FPS, mostly 16-17 FPS, a little laggy

Walking around

2xPSP: 28-30 FPS, mostly 30 FPS

2xPSP+FXAA: 24-30 FPS, mostly 30

FPS

3xPSP: 27-30 FPS, mostly 30 FPS

3xPSP+FXAA: 16-18 FPS, does some skipping while walking

Fights

2xPSP: 15-30 FPS, mostly in the higher 20s

2xPSP+FXAA: 13-23 FPS, slightly laggy, often below but close to 20

3xPSP: 10-30 FPS, often in the lower 20s or below

3xPSP+FXAA: 8-17 FPS, mostly below 15

I was really surprised to see the FPS this low on the C2. This game runs rather well on other devices, and the C2 should have a very strong GPU to easily handle the effects. Still, with the 2xPSP setting, you can play the game in full speed, so there is hope.

**U2/U3**

I forgot to switch graphics for videos, but 2xPSP ran at full speed and I expect similar results on 2xPSP+FXAA and 3xPSP.

Walking around

2xPSP: 30 FPS

2xPSP+FXAA: 30 FPS, a little blurry

3xPSP: 30 FPS

3xPSP+FXAA: 23-24 FPS, looks nice, but skips

4xPSP: 30 FPS

4xPSP+FXAA: 15 FPS, lots of skipping

Fights

2xPSP: 25-30 FPS, can even drop down to 20 FPS at times

2xPSP+FXAA: 25-30 FPS, a little blurry

3xPSP: 25-30 FPS, may drop to 15 FPS when a lot is going on on the scene

3xPSP+FXAA: 15-18 FPS it gets very laggy here

The game runs rather well on the U2/U3. I was even able to test 4xPSP

resolutions while walking around, which looked great. 2xPSP+FXAA was working nicely, but the “low resolution” made the overall experience a little blurry. I found that setting frameskipping to 1x instead of 3x will reduce the skipping of graphics. Overall, any 30 FPS game should only be run with 1x Frameskipping instead of 3x, or else you might experience skipping graphics. The game probably runs best at 3xPSP on the U2/U3, at least in fights. You may want to try 2xPSP+FXAA to see if you get bothered by the blurriness or not. Still, this game is overall a very good experience on the U2/U3.

**XU3/XU4**

I set Frameskipping to 1 for testing on the ODROID-XU3/XU4.

Video

2xPSP: 26-30 FPS, mostly 30 FPS, seems slightly laggy

2xPSP+FXAA: same as 2xPSP

3xPSP: same as 2xPSP

3xPSP+FXAA: same as 2xPSP

4xPSP: same as 2xPSP

4xPSP+FXAA: 28-30 FPS, even better than 2xPSP, which is unusual

I found that on the XU3/XU4, videos being played under ffmpeg seem to be slightly laggy. This doesn't mean that the hardware can't handle it, but rather that something is not working correctly. For example, running a 1080p H264 movie in ffplay on the U3 works perfectly fine. The hardware is close to its limit, but in most cases, it can handle the movie perfectly fine. The same movie on the XU3/XU4, which has twice the performance of the U3, the video seems to skip here and there, but the CPU cores are only 30-50% in use. I can see similar effects in games. Disabling frameskipping may actually improve video quality, although it shouldn't be necessary.

Just for the record, starting at 5xPSP+FXAA, it starts dropping frames and gets laggy. Without FXAA, even

9xPSP resolution was running at 30 FPS. 10xPSP was mostly at 30FPS but could fall down to 22-24 FPS, which was laggy. It's quite amazing, if you think about it. 9xPSP is an internal rendering of 4320x244, and it's running fine on the XU3/XU4. That's a really nicely optimized game!

Walking around

2xPSP: full speed

2xPSP+FXAA: full speed

3xPSP: full speed

3xPSP+FXAA: full speed

4xPSP: full speed

4xPSP+FXAA: full speed

Similarly to videos, performance drops at 5xPSP+FXAA without FXAA, and even 9xPSP is playable. Frametimes can drop when it needs to load new textures into memory. Once data is in memory, there are no lags at all.

Fights

2xPSP: 30 FPS, but may drop to 15 FPS (frameskip) but stays full speed (no lags)

2xPSP+FXAA: 30 FPS, may drop to 18-19 at lowest, no lags

3xPSP: same as 2xPSP

3xPSP+FXAA: same as 2xPSP+FXAA

4xPSP: same as 2xPSP

4xPSP+FXAA: same as 2xPSP, can fall to 15 FPS, but still no lags

There's a lot of room for experimentation here. I improved Spline/Bezier curves quality, disabled lazy texture caching, and disabled lower resolution for effects. These kinds of adjustments made the game looked stunning, but still ran in full speed with frameskipping set to 1. I really like the game for its performance as well as gameplay, and shows what optimization on games can do. The game looks and plays like it was made for the ODROID XU3/XU4. I suggest using 3xPSP+FXAA, and seeing where you want to go from there. This game also very nicely shows what FXAA can do for



you, as seen in the comparison screenshots previously mentioned.

## Asphalt Urban GT 2

For this game, I changed the default settings:

Disable Frameskipping

Activate Disable slower effects (speedup)

Disable Texture coord speedhack (speedup), which surprisingly makes it slower in this game

This is a very interesting game, emulation-wise. I have used it for performance testing ever since I started using PPSSPP. It's kind of hard to emulate, and results varied between different versions a lot, from unplayable slow to full speed without issues. It still has some issues with the effects, but that makes it perfect as a benchmark for seeing the progress of the PPSSPP project.

The game has a very variable frame rate. It does not stay directly at 30 or 60 FPS, but jumps a lot. Even in the menu, you can see the FPS jump between 30 and over 50 FPS, which means that the game speed varies. In game, this happens as well, but normally at a smaller scale between 30-34 FPS, rather than staying at 30 FPS.

The game reacts very badly to frameskipping. It was actually the first game where I had to turn off frameskipping to increase game speed, and now I know that setting the frameskipping too high causes skipping in the game itself, which then makes it seem like the game is lagging, although it's supposed to run at full speed with frameskipping. Still, setting the Frameskipping to 1 can im-



Figure 3 - Asphalt Urban GT 2

prove speed without having visible skipping issues.

## C2

Menu

2xPSP: 30/45 FPS, varies, music laggy

3xPSP: 30/45 FPS, same as 2xPSP

Car Select

2xPSP: 19-23 FPS, extremely laggy

2xPSP+FXAA: 19-20 FPS, extremely laggy

3xPSP: 21-23 FPS, extremely laggy

3xPSP+FXAA: 14-15 FPS, extremely laggy

Racing

2xPSP: 18-21 FPS, very laggy

2xPSP+FXAA: 16-18 FPS, very laggy

3xPSP: 15-18 FPS, very laggy

3xPSP+FXAA: 11-14 FPS, extremely laggy

I haven't found a configuration where this game was playing lag free. Even 2xPSP with Frameskipping set to 1 does not give a lag free experience, even if it's playable at about 85% speed most of the time.

## U2/U3

Menu

2xPSP: full speed

3xPSP: full speed

Car Select

2xPSP: 56 FPS, mostly fine

2xPSP+FXAA: 49 FPS, music laggy

3xPSP: 56 FPS, mostly fine

3xPSP+FXAA: 28 FPS, music very laggy

Racing

2xPSP: Normal racing full speed most of the time, 1x Nitro 1-2 FPS lower, 2x and 3x Nitro around 19/33 FPS, speeder picture 10 FPS

2xPSP+FXAA: Normal racing 23-25 FPS, laggy, 2x and 3x Nitro around 17 FPS, speeder picture 10 FPS

3xPSP: Normal racing 17-19 FPS, very laggy, 2x and 3x Nitro around 11 FPS

3xPSP+FXAA: Normal racing 13-15 FPS extremely laggy, 2x and 3x Nitro around 8-10 FPS

Setting Frameskipping to 1 will improve speed, and 2xPSP+FXAA and 3xPSP are playable that way. Any value higher than 1 will make the game laggy again, since too much skipping makes the graphics jumpy. It is best to keep the game at 2xPSP. With Frameskipping set to 1, the game is fully playable and enjoyable on the U2/U3.

## XU3/XU4

Menu

2xPSP: full speed

2xPSP+FXAA: full speed

3xPSP: full speed

3xPSP+FXAA: 45-49 FPS, slightly too slow

4xPSP: full speed

Car Select

2xPSP: 50% speed, but is full speed with frameskipping 1

2xPSP+FXAA: 50% speed, but is full speed with frameskipping 1

3xPSP: 50% speed, but is full speed with frameskipping 1

3xPSP+FXAA: 50% speed, frameskipping 1 is slightly below full speed, and you hear some lagging

4xPSP: 50% speed, but is full speed with frameskipping 1

4xPSP+FXAA: 50% speed, frameskipping 1 won't help, and the speed is only around 67% which means it's rather laggy, frameskipping 2 would be full speed but also jumps a little

Racing

2xPSP: full speed with frameskipping 1

2xPSP+FXAA: full speed with frameskipping 1

3xPSP: full speed with frameskipping 1

3xPSP+FXAA: full speed with frameskipping 1

4xPSP: slightly below full speed 95%

4xPSP+FXAA: a little below full speed 80-95%

I was a little disappointed that I had to use frameskipping 1 even in 2xPSP, but activating it allows even for smooth 3xPSP+FXAA gameplay, which looks quite nice. It seems that the GPU part is pretty well utilized. Special effects, like when you drive with 2xNitro, don't bother the system at all, while on the U3 and C2, this caused serious drops in framerate. Other effects like the "speeder picture" only drop slightly in framerate, while on other devices there is a massive slowdown for a few seconds.

## Final Fantasy VII - Crisis Core

This game has a lot of different scenes, such as video cut-scenes, scenes where you walk around or talk to people, fighting scenes, and boss fights. As a result, the game may act differently in each situation, and finding a setting that satisfies all scenes equally well may be hard. So, in my tests, I looked for an overall good experience, and not one especially for a certain aspect of the game.



Figure 4 - Crisis Core

## C2

Videos

2xPSP: 16-30 FPS, mostly over 20

2xPSP+FXAA: 13-19 FPS, slightly laggy

3xPSP: 15-30 FPS, mostly over 20

3xPSP+FXAA: 13-17 FPS, mostly around 15

In Game/Fights

2xPSP: 13-30 FPS, mostly below 20 FPS and lagging

2xPSP+FXAA: 13-18 FPS, mostly around 15 FPS and lagging

3xPSP: 13-22FPS, mostly around 15 FPS

3xPSP+FXAA: 8-16 FPS, mostly around 13 FPS or below, quite laggy

This game is unfortunately not a very good experience on the C2, although it's probably playable in 2xPSP. Once again, the C2 shows rather bad results in playing PSP games under Linux.

## U2/U3

Videos

2xPSP: full speed

2xPSP+FXAA: full speed

3xPSP: full speed

3xPSP+FXAA: 25-30 FPS

The results are typical for the U2/U3, and show very good results on video playback via ffmpeg in PPSSPP. The experience of watching video cut-scenes is probably best on the U2/U3.

In Game/Fights

2xPSP: full speed

2xPSP+FXAA: Mostly full speed, some scenes can drop to 25 FPS

3xPSP: Mostly full speed, some scenes can drop to 25 FPS

3xPSP+FXAA: 17-27 FPS often skipping

Since this is also a 30 FPS game, running it with Frameskipping set to 1 is probably best. Playing the game in 2xPSP+FXAA looks very nice, and has really good performance. Unlike Hexyz Force, Final Fantasy VII - Crisis Core does not get blurry under 2xPSP+FXAA, which actually means that it's a good way to improve the graphics of the game, and it seems to run stably on the U2/U3.

## XU3/XU4

Video

2xPSP: 15-30 FPS, mostly around 30 FPS, but doesn't feel laggy

2xPSP+FXAA: same as 2xPSP

3xPSP: same as 2xPSP

3xPSP+FXAA: same as 2xPSP

4xPSP: same as 2xPSP

4xPSP+FXAA: similar to 2xPSP, but more often around 20 rather than 30 FPS

Once again, the bad handling of videos with ffmpeg is evident here. While there are real lags, it still skips frames every now and then, even on low resolutions, but it's nothing serious, and the movies in this game are not really affected by the skipping.

In Game/Fights

2xPSP: 15-30 FPS, mostly full speed at 30 FPS

2xPSP+FXAA: same as 2xPSP

3xPSP: same as 2xPSP

3xPSP+FXAA: same as 2xPSP

4xPSP: similar to 2xPSP, but often at around 20 FPS, can be laggy for a second or two

4xPSP+FXAA: same as 4xPSP

By now, it shouldn't be a surprise that the XU3/XU4 outperforms any other platform. I probably like this game best in 3xPSP+FXAA. However, I experienced a few graphical issues in this game. Some effects resulted in a kind of rainbow light. I'm not sure what's causing this, nor if it's just my image, or any other XU3/XU4 as well. I don't remember seeing this in the past, though.

## Soul Calibur

I rather like this fighting game, and

Figure 5 - Soul Calibur



would prefer it over Tekken anytime. I loved playing Soul Calibur on my Dreamcast, and the PSP version is really good as well.

## C2

### Video

- 2xPSP: 18-24 FPS, laggy
- 2xPSP+FXAA: 15-20 FPS
- 3xPSP: 16-22 FPS
- 3xPSP+FXAA: 12-15 FPS

### In Game/Fights

- 2xPSP: 13-20 FPS, mostly lagfree
- 2xPSP+FXAA: 13-16 FPS, slightly laggy
- 3xPSP: 13-16 FPS, better than 2xPSP+FXAA
- 3xPSP+FXAA: 10-11 FPS, very laggy

I was once again rather disappointed by the C2 here. Normally, the game runs rather well on other platforms, and I would have expected more here. 2xPSP should have at least been in the higher 20s or 30s, but thanks to the Frameskipping setting, 2xPSP is mostly lag free.

## U2/U3

### Video

- 2xPSP : 40-60 FPS, no visible lags, mostly over 50 FPS
- 2xPSP+FXAA: 35-50 FPS, can have some slight lags, mostly over 40 FPS, rarely close to 50
- 3xPSP: 38-60 FPS, no visible lags, mostly over 50 FPS
- 3xPSP+FXAA: 28-30 FPS, I saw 26 FPS once, but most of the time it's at 29 FPS

### In Game/Fights

- 2xPSP: 44-60 FPS, no lags, mostly above 50 or even at 60 FPS
- 2xPSP+FXAA: 36-45 FPS, no lags, mostly around 40 FPS
- 3xPSP: 36-60 FPS, no lags, mostly around 45-50 FPS, but also 60 FPS sometimes
- 3xPSP+FXAA: 23-26 FPS, slight skipping, especially when the camera is

performing a fly-by.

The game runs very well on the U2/U3, and fights are really fun. 3xPSP+FXAA would work well on this game if it had a few more FPS, but since it's slightly laggy, setting it to 3xPSP or 2xPSP+FXAA should be the best option on the U2/U3.

## XU3/XU4

For the XU3/XU4, I experimented with no Frameskipping in this game, but at it turned out, some scenes do require Frameskipping of 1x or 2x, but after that the game worked perfectly fine.

### Videos

- 2xPSP: 24-60 FPS, can be slightly laggy, disabling Frameskipping keeps FPS at 60
- 2xPSP+FXAA: can drop to 55 FPS without Frameskipping, otherwise it behaves the same as 2xPSP
- 3xPSP: same as 2xPSP
- 3xPSP+FXAA: requires Frameskipping, 23-57 FPS, sometimes you can see the skipping
- 4xPSP: same as 2xPSP
- 4xPSP+FXAA: 20-32 FPS with Frameskipping, video stars getting a little laggy here

No Frameskipping works best here, but since the game needs Frameskipping,

depending on the resolution you choose, 3xPSP+FXAA and Frameskipping set to 1 or 2 is probably the best choice.

### In Game/Fights

- 2xPSP: depending on the background, even no Frameskipping runs at 60 FPS, but it is best played with 1x Frameskipping which is full speed, even if it's just at 30 FPS
- 2xPSP+FXAA: requires Frameskipping, but with 1x Frameskipping it's also full speed
- 3xPSP: with 1x Frameskipping it runs at full speed at 30 FPS
- 3xPSP+FXAA: runs mostly fine on 1x Frameskipping, but may slow down for a certain move or final blow, using 2x frameskipping keeps the game between 20-30 FPS with no lags or visible skipings.
- 4xPSP: runs fine with 1x Frameskipping
- 4xPSP+FXAA: runs perfectly fine with 2x Frameskipping

I was actually surprised to see 4xPSP+FXAA working that well, but even 3xPSP+FXAA already looks gorgeous and runs without any issues.

## GTA - China Town Wars

This game is also rather hard to emulate, but is fun to play, and although ev-





# ODROID Magazine is on Reddit!



## ODROID Talk Subreddit

<http://www.reddit.com/r/odroid>



ery action you do is on the same “field”, the performance of these action are quite different. Walking through town, driving in a slow car, driving in a fast car (even driving slow in a fast car), and fighting enemies with different weapons all react differently, so I will base my results on an overall average, although driving in cars is generally what’s stressing the system the most.

### C2

In Game

2xPSP: 9-15 FPS, depending on the car you drive, it can drop even lower

This was a really bad experience that I did not expect. This game runs really poorly on the C2. You can probably play it in 2xPSP with Frameskipping set to 1, but it’s far from being free of lags. I didn’t even bother to try in any higher resolution or with FXAA activated, since both of those settings degade performance even more.

### U2/U3

In Game

2xPSP: 25-30 FPS mostly full speed, can drop to 20 FPS on high speed chases

2xPSP+FXAA: same as 2xPSP

3xPSP: 20-30/30 FPS, mostly around 22-25 FPS, can drop to 20 FPS on high speed chases.

3xPSP+FXAA: 15-20 FPS, mostly around 17-18 FPS, will drop to 13 FPS and below on high speed chases

Although 2xPSP and 2xPSP+FXAA react nearly the same, the game does not benefit from FXAA that much, so it’s probably better just to use the 2xPSP setting. Higher resolutions such as 3xPSP may work as well, but in some cases, you may need the little bit of extra speed in the game, so I’d suggest sticking with 2xPSP with a setting of Frameskipping equal to 1.

### XU3/XU4

In Game

2xPSP: mostly full speed, except if you drive really fast cars

2xPSP+FXAA: same as 2xPSP

3xPSP: same as 2xPSP

3xPSP+FXAA: same as 2xPSP

4xPSP: same as 2xPSP

4xPSP+FXAA: slightly laggy occasionally, especially in high speed cars

This game also run better than I expected. It is probably best played in 3xPSP+FXAA or 3xPSP mode, since FXAA doesn’t seem to improve the graphics on this game anyway.

### Conclusion

I was very disappointed with the ODROID-C2 performance while running PPSSPP, which leads me to the conclusion that the ARM64 dynamic re-compiler is either not working correctly, or is not fully implemented yet, since I expected similar results as the U2/U3 performance wise, especially since the C2 has the better GPU. The U2/U3, the X2, and probably the ODROID-X as well are still rather solid devices, and have no issues playing gorgeous looking Playstation Portable games. Considering that these ODROID devices are now almost five years old shows what this hardware is still capable of.

Not surprisingly, the XU3/XU4 outperforms every other ODROID board when it comes to PSP emulation. The CPU is fast enough to emulate most games in full speed, and paired with the very strong GPU of the XU3/XU4, you can easily increase render resolution and activate shaders like FXAA. I’ve played and finished a couple of different games on the U3 and XU3 already, and from personal experience, I can tell that PPSSPP runs very well on ODROIDS and the graphics look just awesome. I hope that the issues with the C2 can be solved soon, and that we eventually see PPSSPP performance that is at least similar to the older ODROID-U2/U3 device.

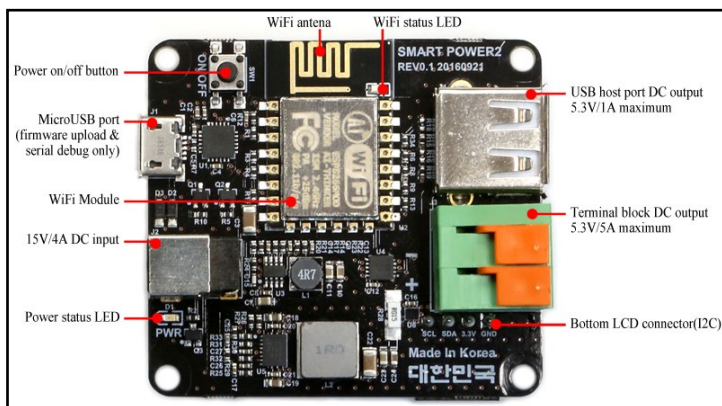
# SMART POWER2

YOUR ENERGY-SAVING BEST FRIEND

edited by Rob Roy (@robroy)



The SmartPower2 is a WiFi-enabled power supply that controls the voltage and switch on/off a connected device, available for purchase from the Hardkernel store for USD\$40 at <http://bit.ly/2j3hhcv>. You can also monitor the load current and power consumption remotely on your smartphone, tablet or PC by navigating to the built-in web page UI. The full guide, schematics, PCB layout file and detailed information are available on the Hardkernel Wiki at <http://bit.ly/2jVXvOC>.



Annotated layout of the SmartPower2 board

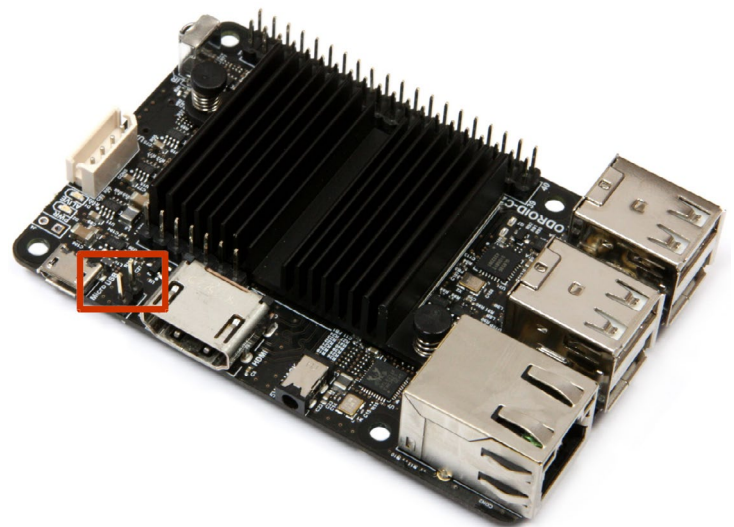


Remote monitoring of the SmartPower2 via smartphone

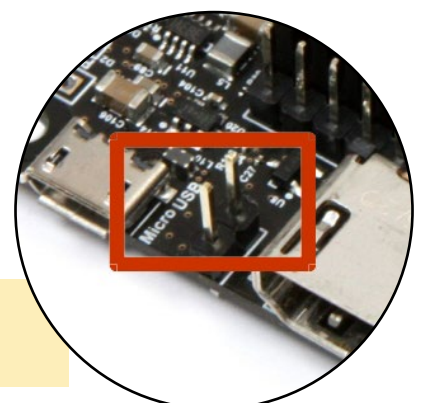
# ODROID-C2 POWER CONSUMPTION

A SIMPLE TWEAK THAT WILL WORK WONDERS

edited by Rob Roy (@robroy)



If you own an ODROID-C2, you can reduce the power consumption significantly by adjusting one of the jumpers. By default, the board ships with jumper J1 in the “on” position, which allows the board to be powered via the OTG port. However, if your ODROID-C2 is using the 5V DC plug instead, this can be an unnecessary use of power. According to measurements while allow the device to idle at a serial console, the power usage is around 1.7W. After removing the jumper, as shown in Figure 1, the usage drops to 1.2W, a savings of 0.5W. This modification will also reduce heat produced from the board. For comments, questions and suggestions, please visit the original post at <http://bit.ly/2jSI1sl>.



Jumper J1 removed

# MEET AN ODROIDIAN

RICHARD BOWN (@RICHARD-G8JVM)

edited by Rob Roy (@robroy)

*Please tell us a little about yourself.*

I have now retired, but I used to work as a RF engineer. I started in repair and calibration of RF test equipment, then moved on to design and development of mobile phone and base stations.

I have also been a ham radio amateur since 1974. I live in the rural county of Shropshire, England, not too far from the border with Wales. I studied industrial electronics at the university of the west of England, with a focus on digital electronics. I've been married to my wife, Sue, for the last 44 years, and we have two sons. The eldest has his own family and works for a major car manufacturer in the southeast area of England, globally troubleshooting manufacturing problems. My younger son has his own heating and plumbing business in Shropshire. You can learn more about me and my projects at <http://g8jvm.com>.

*How did you get started with computers?*

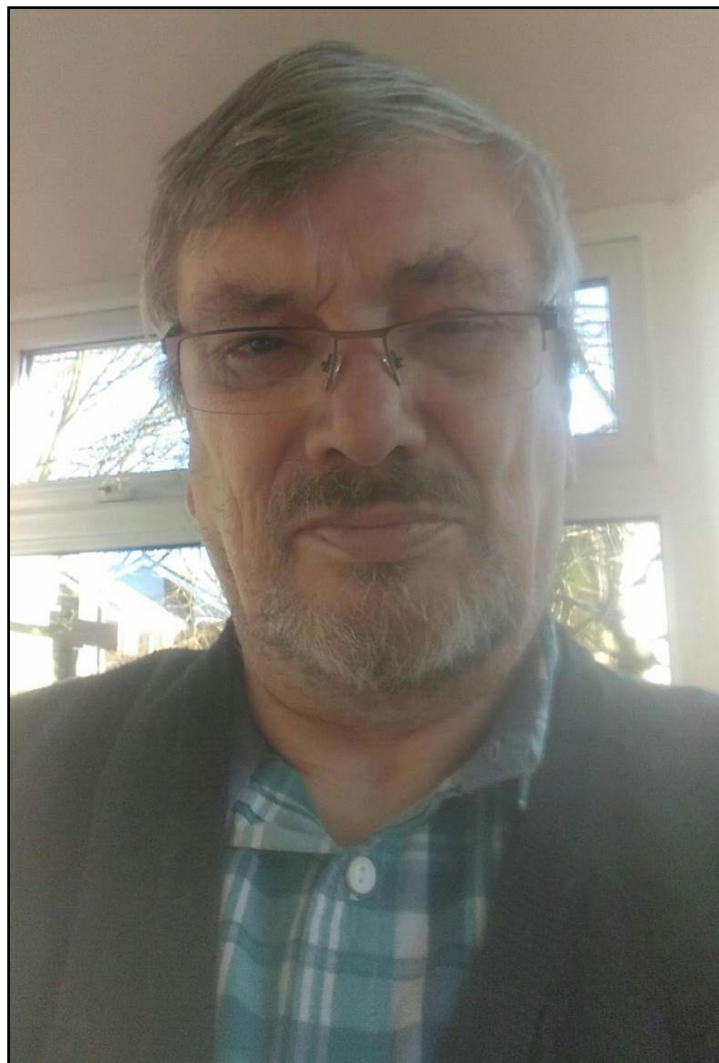
I've had some interest in computers since the 1980s, but I built my own Z80 based machine while in college. I learned a lot of instruction code programming, but as higher languages became more popular, those skills were soon forgotten. I started using Linux 22 years ago while getting involved in packet radio, not so much for chatting or mailboxes at 1200 baud, but more for the implementation of a high speed TCP/IP network in the UK, similar to that in Germany. However, I found that the technical desire in the UK was lacking, so I went back to my main interest in ham radio, which is weak signal working on VHF to microwave frequencies. This area now has some very useful software available to assist, but it uses very narrow bandwidths to extract signals from the noise floor.

*Whom do you admire in the world of technology?*

I admire Joe Taylor K1JT, for his work on weak signal software.

*What attracted you to the ODROID platform?*

Initially, I started with a C1+ to run motion for an IP camera. I then found it could also be used for Kodi as well. I did look at the Raspberry Pi, but the specifications of the C1+ were higher, albeit less used within the ham radio community.



**Richard is a long-time amateur ham radio operator**

*How do you use your ODROIDS?*

I got an ODROID-XU4 later to replace my main computer. I managed to get many of the applications that I use on a Linux platform to run on the XU4, such as the WSJTX suite. I also use Live MUF with a lot of help from Dave G7RAU, since this app needs Mono to run. The XU4 hosted other ham-related software, as well as Piklab which run on QT4, which has a nice easy GUI and is easy to use, unlike MPLAB. Unfortunately, I've recently retired the XU4 and returned to a small x86\_64 platform, since I needed to run several applications at the same time, which exceeded the capability of the XU4. I've upgraded



**Richard's radio shack, where he spends much of his free time**

my C1+ to a C2 running Ubuntu 16.04, which is a far more stable platform for supporting two IP cams and a USB camera on motions, as well as being a faster platform for Kodi.

*Which ODROID is your favorite and why?*

I like the XU4, because it's a more heavy-duty platform than the C1 or C2.

*What innovations would you like to see in future Hardkernel products?*

I'd like to see an infrared (IR) receiver built into all ODROID models so that a remote control can be used, and more onboard memory to handle graphics faster. It needs to be able to handle the startup of many programs which use a graphic display, especially where high-resolution maps are used.

*What hobbies and interests do you have apart from computers?*

I've heavily involved in ham radio. I've held the callsign G8JVM since 1974 (CEPT class 1). I also practice photography

as well as collecting model railways (4mm GWR circa 1935).

*What advice do you have for someone wanting to learn more about programming?*

Very simple: don't give up. We all have to start somewhere, and don't be put off by the occasional flames. Read a lot, and don't be afraid of breaking something, which is challenging.



**Richard's wife Susan in Monument Valley, Utah**