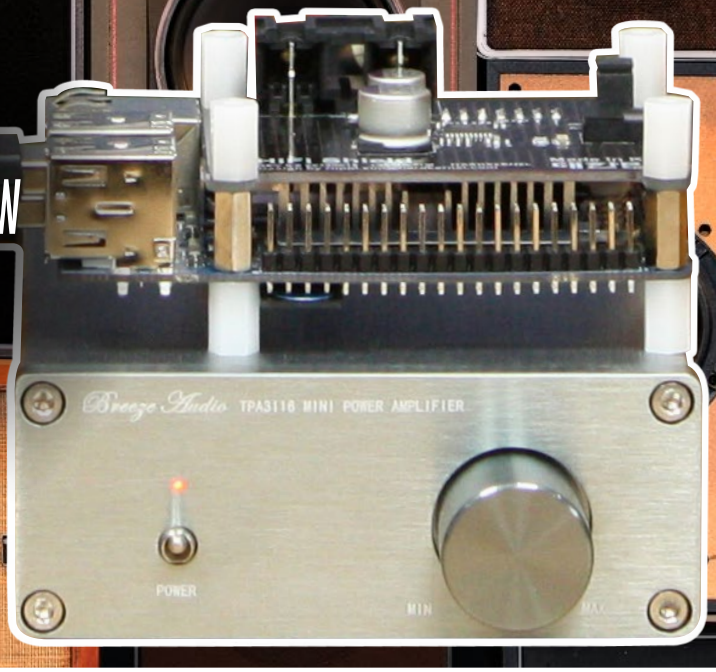
Manipulating VGs • Handbrake • Cloudshell • LFTP and Crontab Backup

ODROID Magazine

Year Two
Issue #23
Nov 2015

Hifi Shield

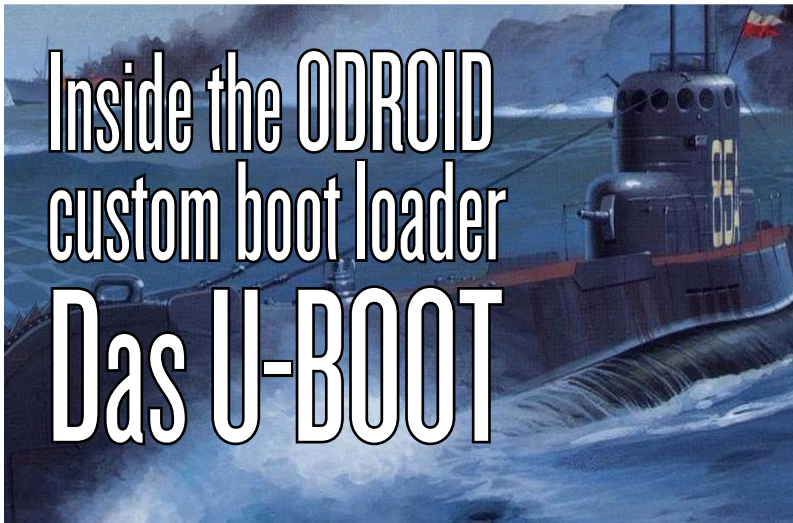
The world's most power efficient board now provides you with the best audio possible



Build your own RuneAudio media player

Inside the ODROID custom boot loader

Das U-BOOT



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

Pollin ELECTRONIC

We are now shipping the ODROID-U3 device 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

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

High fidelity audio reproduction has traditionally implied spending a lot of money, but Hardkernel's new HiFi-Shield (USD $39 at http://bit.ly/1M6UIXY) makes it very cost-effective to turn your ODROID-C1+ into a professional quality music player. When used with software such as RuneAudio or Volumio, it's perfect as the center of an integrated home audio system that can be managed from a single touchscreen device. A C1+, when paired with the 7" HDMI touchscreen (USD $55 at http://bit.ly/1NWxgDx) and the custom RuneAudio image becomes a complete, affordable home audio system.

This month, we concentrate on system and hardware administration. Uli gives an overview of the ARM bootloader called Das U-Boot, David continues his series on Logical Volume Management, Andrew covers techniques for using LFTP and CRON to automate backups, Pascal makes it easy to prevent spam from overwhelming your mailbox, Andy presents a Cloudshell personal server project, and Adrian teaches us how to transcode videos using Handbrake. For the fun side of ODROIDs, Tobias introduces us to GLShim, which allows a wide variety of games to run on the ODROID platform.

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, 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

ODROID Magazine

Rob Roy, Chief Editor

I'm a computer programmer living and working 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`.

Robert Cleere, Editor

I am a hardware and software designer currently living in Huntsville, Alabama. While semi-retired from a career in embedded systems design, including more than a decade working on the Space Shuttle program, I remain active with hardware and software product design work as well as dabbling in audio/video production and still artwork. My programming languages of choice are Java, C, and C++, and I have experience with a wide range of embedded Operating Systems. Currently, my primary projects are marine monitoring and control systems, environmental monitoring, and solar power. I am currently working with several ARM Cortex-class processors, but my ODROID-C1 is far and away the most powerful of the bunch!

Bruno Doiche, Senior Art Editor

Our always confident Art Editor had to manage some hard worked moonlighting for some crazy IT clients this month, but with good humour he turned all limes that were thrown at his direction to the most brazilian of caipirinhas!

Nicole Scott, Art Editor

I'm a Digital Strategist and Transmedia Producer specializing in online optimization and inbound marketing strategies, social media directing, and media production for print, web, video, and film. Managing multiple accounts with agencies and filmmakers, from Analytics and Adwords to video editing and DVD authoring. I own an ODROID-U3 which I use to run a sandbox web server, live in the California Bay Area, and enjoy hiking, camping and playing music. Visit my web page at http://www.nicolecscott.com.

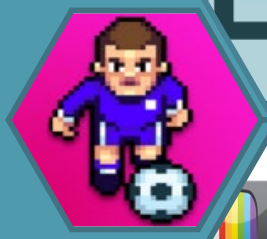
James LeFevour, Art Editor

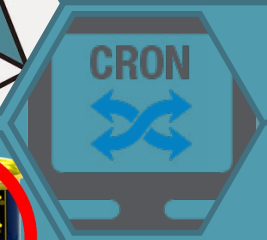
I am a Digital Media Specialist who is also enjoying freelance work in social network marketing and website administration. The more I learn about ODROID capabilities, the more excited I am to try new things I'm learning about. Being a transplant to San Diego from the Midwest, I am still quite enamored 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.

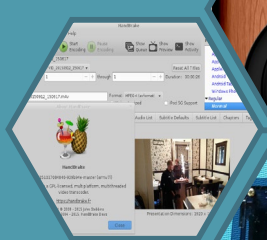
Manuel Adamuz, Spanish Editor

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.

INDEX



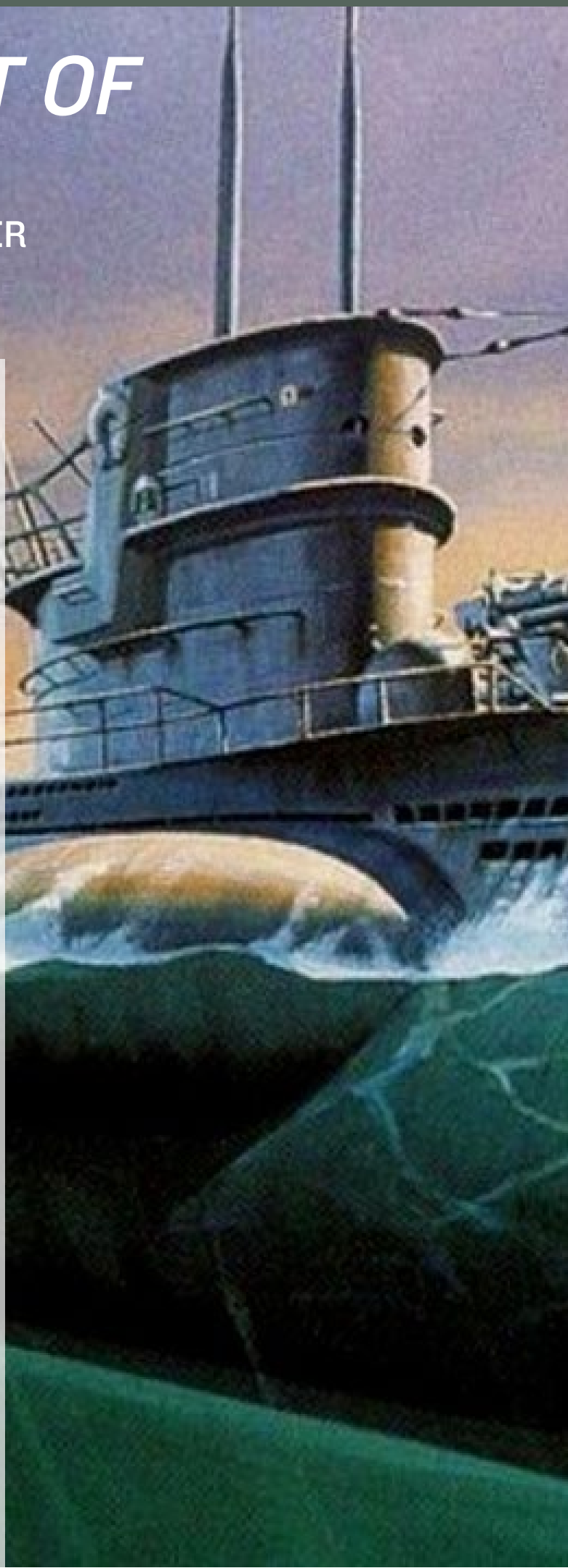




GET MORE OUT OF DAS U-BOOT

INSIDE THE ODROID BOOT LOADER

by Uli Middelberg



The majority of recent ARM based devices use "Das U-Boot" (u-boot) to load and start the Linux kernel. Many vendors focus on an end-user friendly u-boot configuration. If you need more flexibility, it is worth a deeper look into the u-boot internal structure.

What is u-boot?

U-boot is a boot loader similar to lilo (http://bit.ly/1W69Dg7) and grub (http://bit.ly/1PAFYY5), but is specifically designed for embedded devices. U-boot is maintained by @denx (http://bit.ly/1LFlOGm), and is published under the GNU General Public License v2 (GPL-2.0+). Compared to grub, u-boot doesn't offer a high level of self-configuration or end-user friendliness, but it does offer a much smaller footprint.

The most convenient way to interact with u-boot is via serial console. Many ARM devices offer serial access via USB, which makes it easy to manage console access. Just connect the serial console via USB to another Linux box, and you will have remote serial access to your ARM device. To do so, login to your Linux box via ssh and open the serial connection via minicom:

```
$ sudo minicom -b 115200 -D /dev/ttyUSB0
```

Press "Ctrl-A z" to open the interactive help/configuration screen, and press "Ctrl-A x" to exit your minicom session. Minicom isn't the only terminal emulator for serial access. Screen and picocom are popular alternatives:

```
$ sudo picocom -b 115200 /dev/ttyUSB0
```

When you are done using picocom, press "Ctrl-A Ctrl-X" to exit the session.

Boot process

When a device gets powered on, it has to go through various initialization stages before you see the operating system

login prompt or desktop. The first stage of the boot process is the Secondary Program Loader (SPL). This preliminary piece of code is responsible for board initialization, loading the u-boot binary ("secondary program") and handling the control flow over to the u-boot main program. It is device-specific, and is often provided as a closed source binary blob by the SoC vendor.

Secondary Program Loader

The secondary program loader (SPL) and the u-boot binary reside in a special on-board flash memory region or on the first sectors of the boot disk. ODROID devices use microSD or eMMC modules for storing the SPL and u-boot binary. The individual disk regions are detailed in the disk layout and partitioning section below.

U-Boot

At the second stage of the boot process, the u-boot main program is executed. U-boot first looks for a custom environment stored at a reserved space on the microSD or eMMC module, or falls back to the compile time default environment if needed. At this time, you can interrupt the automatic boot process by pressing a key on your serial console, which starts an interactive u-boot shell. The u-boot variable called bootdelay specifies the number of seconds to wait for a keypress before continuing automatic boot.

```
CPU : AMLogic S805
MEM : 1024MB (DDR3@792MHz)
BID : HKC13C0001
S/N : HKC1CC037EBCBFA4
0x0000009f
Loading U-boot...success.


U-boot-00000-geb22ea4-dirty(odroidc@eb22ea4b) (Jul 28
2015 - 22:16:46)

DRAM:  1 GiB
relocation Offset is: 2ff1c000
MMC:   eMMC: 0, SDCARD: 1
IR init is done!
vpu clk_level = 3
set vpu clk: 182150000Hz, readback:
182150000Hz(0x701)
mode = 6  vic = 4
set HDMI vic: 4
mode is: 6
viu chan = 1
config HPLL
config HPLL done
reconfig packet setting done
MMC read: dev # 0, block # 33984, count 12288 ...
12288 blocks read: OK
There is no valid bmp file at the given address
================================================
========
Vendor: Man 450100 Snr 01172c20 Rev: 4.7 Prod: SDW16
            Type: Removable Hard Disk
            Capacity: 15028.0 MB = 14.6 GB (30777344
x 512)
------------------------------------------------
--------
Partition     Start Sector     Num Sectors     Type
```

```
    1                3072          524288       b
    2              527360        14680064      83
    3            15207424        15569920      83
=================================================
========
Net:   Meson_Ethernet
init suspend firmware done.  (ret:0)
Hit Enter key to stop autoboot -- :  3 tstc enter

exit abortboot: 1
odroidc#
```

The automatic boot process executes a special u-boot macro, called bootcmd, which loads and executes the following procedures:

(optional) a custom u-boot environment, such as uEnv.txt
(optional) a precompiled u-boot macro, such as boot.scr
the kernel image, such as uImage
(optional) the device tree binary, such as meson8b_odroidc.dtb
(optional) the initial ramdisk, such as uInitrd

Linux kernel

The third stage is the loading of the Linux kernel. However, before the Linux kernel takes control, u-boot passes a command line to the kernel containing essential parameters. These parameters can be viewed after the operating system has booted by typing the following into a Terminal window:

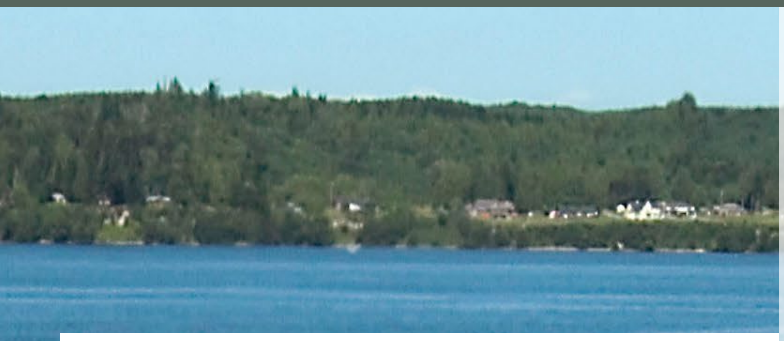
```
$ cat /proc/cmdline
root=/dev/mmcblk0p2 rootwait rw
console=ttyS0,115200n8 console=tty0 no_console_sus-
pend vdaccfg=0xa000 logo=osd1,loaded,0x7900000,720p,
full dmfc=3 cvbsmode=576cvbs hdmimode=1080p m_bpp=32
vout=hdmi disablehpd=true
```

The kernel initializes the hardware, mounts the root filesystem according to the "root=" kernel parameter, then passes the control flow to /sbin/init.

Hardkernel shortcut

Hardkernel makes use of a special u-boot command called cfgload, which bypasses the ordinary boot process and provides a simplified u-boot configuration facility in a single file called boot.ini. Configuration changes can be done easily by editing the file boot.ini, rather than modifying the u-boot environment), but this extension doesn't provide any access to the interactive u-boot shell.

You can boot only one configuration at a time by default. If a particular configuration change causes the system to hang during boot, you will need to remove the microSD or eMMC module from your device and revert the change by editing the boot.ini file on your notebook or PC.



Area	ODROID-C1+	ODROID-U3/XU3/XU4
SPL (BL1/BL2)	0 - 6	1 - 62
U-boot executable	64 - 1023	63 - 718
U-boot environment	1024 - 1087	1231 - 1262
1st partition	3072 - ...	3072 - ...

Figure 1 - Partition layouts differ between ODROID boards

device	label	filesystem	mount point
/dev/mmcblk0p1	boot	vfat	/media/boot
/dev/mmcblk0p2	rootfs	ext4	/

Figure 2 - Partition layout for vfat-based bootloader

Disk layout and partitioning

Many ARM boards, including ODROIDs, use a microSD or eMMC module device to store the different u-boot components: SPL, u-boot executable, and u-boot environment. In contrast to other devices which use flash based memory storage for this purpose, such as tablets and smartphones, you cannot render your device unbootable with an unsuccessful firmware update.

The u-boot components are stored on reserved areas of the disk before the first partition starts. As shown in Figure 1, there is no common canonical layout for the u-boot components across different boards. If you want to modify the partitioning with tools like fdisk, you should keep in mind that fdisk always tries to create new partitions starting from sector 2048 by default.

U-boot tries to load the kernel image and additional files from the first partition of the designated boot device. Earlier u-boot versions supported only the vfat file system, which resulted in the typical partition layout shown in Figure 2.

Recent u-boot versions support reading files from ext4 file-systems, so there is no technical need to use the vfat boot partition. Many installer images continue to use vfat for user convenience, since it allows editing of the boot.ini file on Windows and OSX, which don't support ext4 file systems by default.

U-boot commands

The interactive u-boot shell offers a set of executable commands, depending on the version and patch level. You can get a list of all supported commands by typing "help" or "help <command>". For further information please refer to the official documentation at http://bit.ly/1PH80Ai.

U-boot environment

The u-boot environment stores a set of variables in the form <variable>=<value>. If a saved environment is available on the disk, u-boots initializes the working environment with these values, or falls back to the build-in default environment. Variables are referenced by ${variable}:

printenv: prints out the whole set of variables

printenv variable: prints the value of a single variable

setenv variable value: sets a certain variable with the designated value

setenv variable: deletes a variable from the environment

env -a: resets the u-boot environment with the build-in defaults

saveenv: stores the current environment onto the microSD or eMMC module. You can also read and modify the u-boot environment when the system has started Linux.

U-boot macros

U-boot uses variables to store scripts, so the following command will define a variable called macro containing a list of commands, delimited by a semicolon (;):

```
setenv macro '...;'
```

Macros may be invoked by typing the "run" command:

```
run macro
```

The most prominent macro is bootcmd, which is run by default. Macros return the value of the last command executed, which can then be evaluated using an if/then/else clause. Additional information is returned by modifying specific variables. For example, the "load" command uses the global variable called "filesize" in order to return the number of bytes read from disk.

Typical boot sequence

The macro named "bootcmd" implements the boot sequence, which is executed in non-interactive mode. Prior to executing bootcmd, u-boot initializes the u-boot environment containing the configuration variables and macros. The typical steps inside the bootcmd macro are detailed below.

Load a custom environment from the boot partition

The bootcmd macro looks for a text file called uEnv.txt on the boot partition, load it, then merge its contents with the existing u-boot environment by overriding values of existing variables. This step is optional.

Load a custom environment from the boot partition

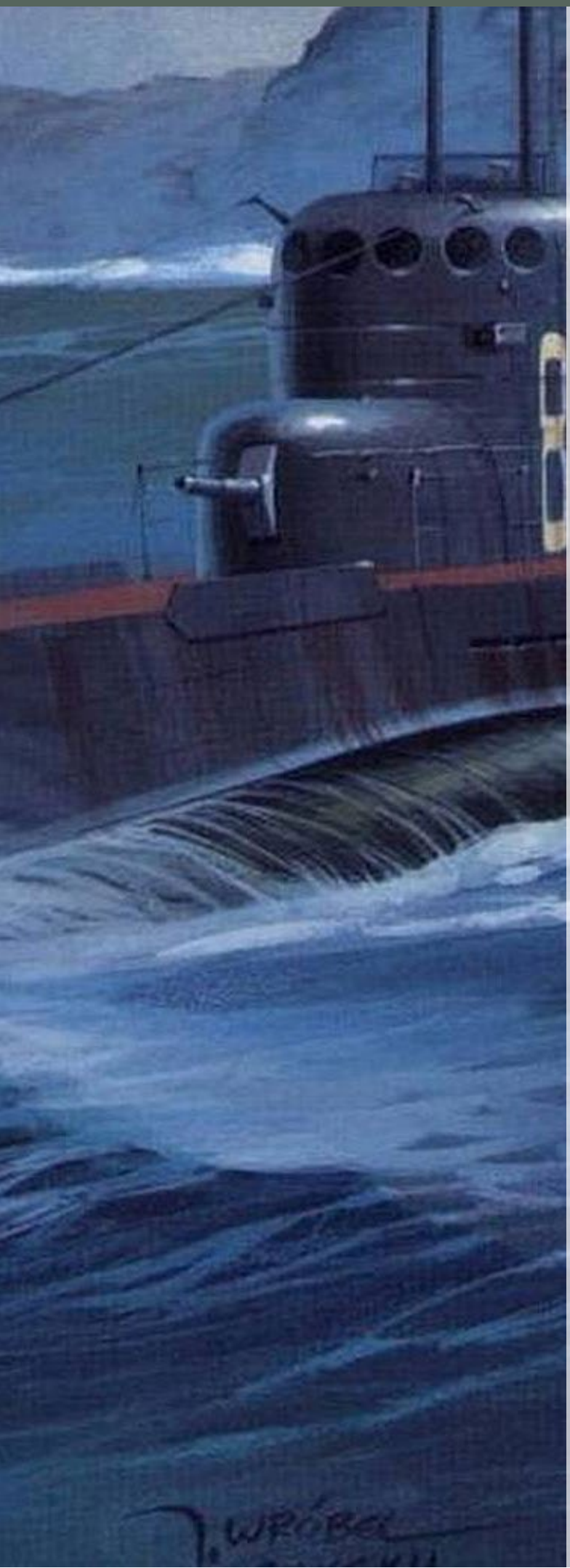
The bootcmd macro will then look for a file called boot.scr on the boot partition, load it, then execute its contents without returning back to bootcmd. The boot.scr file contains compiled u-boot commands in a binary format. This step is optional.

Load the kernel image from the boot partition

The bootcmd macro will subsequently look for the kernel image. The u-boot variable kernel contains the actual filename, usually called zImage. You can boot an alternate kernel image by changing the variable named "kernel".

Load the device tree binary from the boot partition

The Linux kernel an ARM needs a low level device description, or device tree, in binary format, either appended to the kernel image, or as a separate file. Many platforms prefer loading the device tree binary as a separate file, which offers more flexibility, allowing distribution of a single installation image for different platforms, or tweaking of the device tree for different use cases. Some vendors let bootcmd decide on



the actual device tree binary to load, depending on the board discovery performed by u-boot.

Load the initial ramdisk archive from the boot partition

The bootcmd macro will finally try to load the initial ramdisk archive from a file called uInitrd. This archive is created and updated by using the update-initramfs utility, which is usually done when a new kernel image has been installed. This step is optional.

Userland access to u-boot

After the system has booted, you can still modify the u-boot environment. The Ubuntu package called u-boot-tools contains two utilities: fw_printenv and fw_setenv, which are used for accessing the stored u-boot environment. First, install the package with the following command:

```
$ sudo apt-get -y install u-boot-tools
```

You then need to configure the storage device and address information in the file /etc/fw_env.config. As an example, this is the configuration for the ODROID-C1:

```
# <device>   <offset> <length>
/dev/mmcblk0 0x80000   0x8000
```

Test the configuration with the following command:

```
$ sudo fw_printenv bootdelay
```

The u-boot environment stored on disk contains a CRC checksum. If the offset and length doesn't match, you will receive a warning:

```
Warning: Bad CRC, using default environment
```

Typical use cases

By default, the ODROID-C1 is configured to boot via cfgload with a very short boot delay. Increasing the u-boot variable called bootdelay gives you the opportunity to interrupt the automatic boot process. You can use the fw_printenv command to check the current value, and the fw_setenv command to assign a new value to bootdelay:

```
$ sudo fw_printenv bootdelay
$ sudo fw_setenv bootdelay 3
```

Boot an alternative kernel

If you compile your own kernel from source, you may want to test the new kernel before overwriting the current one, or keep a known working kernel as a backup. Booting

a different kernel than the default kernel can be done by defining an u-boot macro, which refers to different files on the boot partition:

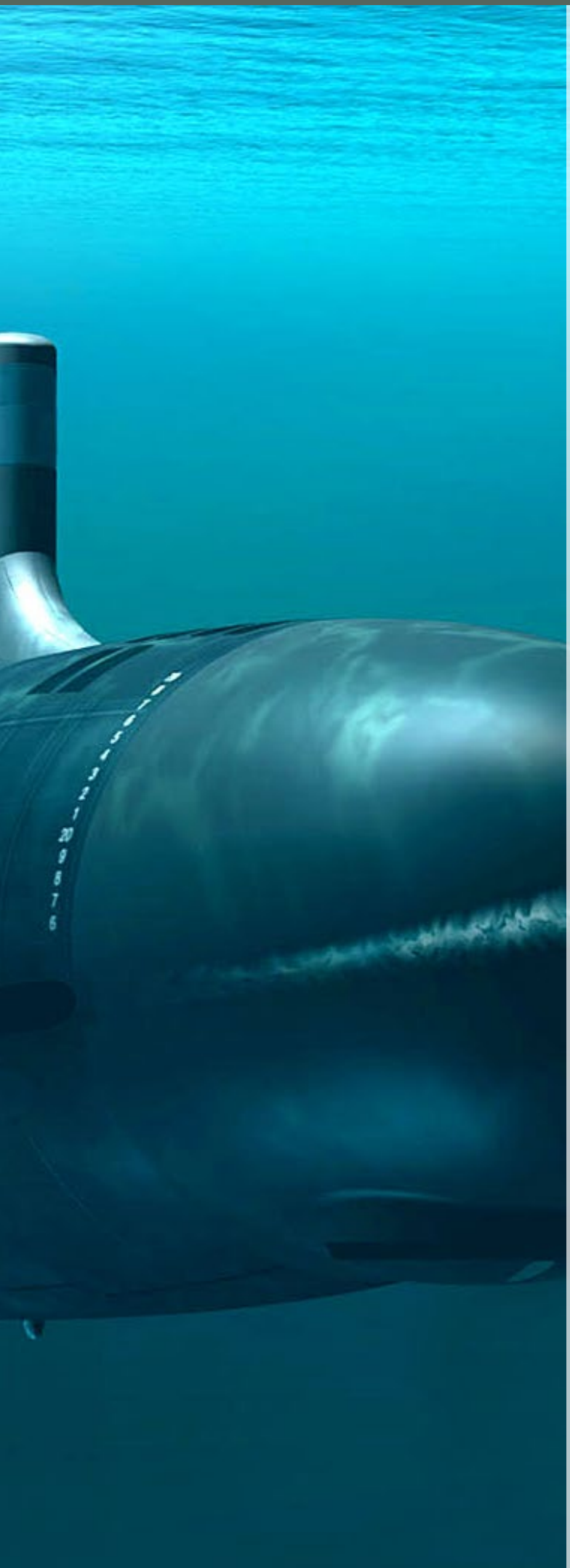
```
sudo setenv m_boot_ 'setenv bootargs "root=/dev/
mmcblk0p2 rootwait rw console=ttyS0,115200n8
console=tty0 no_console_suspend vdaccfg=0xa000
logo=osd1,loaded,0x7900000,720p,full dmfc=3
cvbsmode=576cvbs hdmimode=1080p m_bpp=32 vout=hdmi
disablehpd=true"; fatload mmc 0:1 0x21000000 _
uImage;fatload mmc 0:1 0x22000000 uInitrd; fatload
mmc 0:1 0x21800000 _meson8b_odroidc.dtb; fdt addr
21800000; fdt rm /mesonstream; fdt rm /vdec; fdt rm /
ppmgr; fdt rm /mesonfb; bootm 0x21000000 0x22000000
0x21800000'
```

This defines the u-boot macro "m_boot_", which references the kernel image "_uImage" and the device binary "_meson8b_odroidc.dtb". Unfortunately, u-boot doesn't allow line feeds inside macros, which makes them hard to read. If you insert a line feed after each semicolon (;), you will see the same command sequence as in boot.ini:

```
setenv bootargs "root=/dev/mmcblk0p2 root-
wait rw console=ttyS0,115200n8 no_console_suspend
vdaccfg=0xa000 logo=osd1,loaded,0x7900000,720p,fu
ll dmfc=3 cvbsmode=576cvbs hdmimode=1080p m_bpp=32
vout=hdmi disablehpd=true";
fatload mmc 0:1 0x21000000 _uImage;
fatload mmc 0:1 0x22000000 uInitrd;
fatload mmc 0:1 0x21800000 _meson8b_odroidc.dtb;
fdt addr 21800000;
fdt rm /mesonstream;
fdt rm /vdec;
fdt rm /ppmgr;
fdt rm /mesonfb;
bootm 0x21000000 0x22000000 0x21800000';
```

This is the effective command sequence during boot when configuring boot.ini to a headless configuration:

```
setenv vout_mode "hdmi"
setenv m_bpp "32"
setenv hpd "0"
setenv cec "0"
setenv vpu "0"
setenv hdmioutput "0"
```



Once you have defined the macro m_boot_, and copied the kernel image and device tree binary to the boot partition with a leading '_' in the filename, you can boot this kernel by interrupting u-boot and typing the following at the u-boot shell prompt:

```
run m_boot_
```

Boot rootfs from different partition or USB drive

You might have noticed the kernel parameter "root=", which u-boot passes to the Linux kernel. The kernel will try to mount the root filesystem (/) from there. The root filesystem can be addressed in different ways:

```
via device node: root=/dev/sda1
via UUID filesystem identifier:  root=UUID=e139ce78-
9841-40fe-8823-96a304a09859
```

If you have only one USB storage device connected to your board, you can safely reference the root filesystem using the device node. If you plan to dynamically connect or remove additional storage devices, you're better off by referencing the root filesystem using the UUID identifier, otherwise, the kernel might miss the root filesystem due to a "forgotten" USB stick during the next boot. You can read the available UUIDs using the blkid command:

```
$ sudo blkid
/dev/mmcblk0p1: SEC_TYPE="msdos" LABEL="boot"
UUID="E26F-2230" TYPE="vfat"
/dev/mmcblk0p2: LABEL="rootfs" UUID="e139ce78-9841-
40fe-8823-96a304a09859" TYPE="ext4"
/dev/mmcblk0p3: LABEL="rootfs2" UUID="e139ce78-9841-
40fe-8823-96a304a09860" TYPE="ext4"
/dev/sda1: LABEL="rootfs" UUID="e54a458d-6a66-4ed2-
9394-7b22d2943ec9" TYPE="ext4"
```

The UUID can be set while creating a filesystem with the mkfs command:

```
$ sudo mkfs.ext4 -O ^has_journal -b 4096 -L rootfs -U
e54a458d-6a66-4ed2-9394-7b22d2943ec9 /dev/sda1
```

If you omit the parameter -U, the UUID is chosen randomly. As shown the previous section, you can define an additional u-boot macro which will then pass an alternative root filesystem to the kernel:

```
sudo setenv m_boot_usb 'setenv bootargs "root=/dev/
sda1 rootwait rw console=ttyS0,115200n8 console=tty0
no_console_suspend vdaccfg=0xa000 logo=osd1,loa
ded,0x7900000,720p,full dmfc=3 cvbsmode=576cvbs
hdmimode=1080p m_bpp=32 vout=hdmi disablehpd=true";
fatload mmc 0:1 0x21000000 uImage;fatload mmc 0:1
0x22000000 uInitrd; fatload mmc 0:1 0x21800000 me-
son8b_odroidc.dtb; fdt addr 21800000; fdt rm /meson-
stream; fdt rm /vdec; fdt rm /ppmgr; fdt rm /mesonfb;
bootm 0x21000000 0x22000000 0x21800000'
```

Although it is not recommended, if you need to clone the contents of a mounted root filesystem you may perform a bind mount before copying:

```
$ sudo mount /dev/sda1 /media/usb
$ sudo mount -o bind / /mnt
$ cd /mnt
$ sudo find .  | sudo cpio -dump /media/usb
$ cd
$ sudo umount /mnt
$ sudo umount /media/usb
```

Which then allows you to boot from USB by typing the following at the u-boot shell prompt:

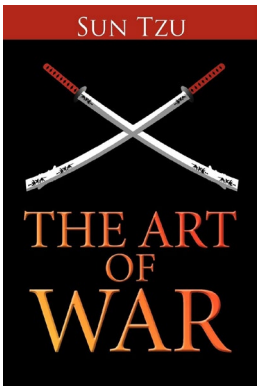
```
run m_boot_usb
```

For questions, comments, or suggestions, please visit the original post at http://bit.ly/1LMcDHM.

GIGALOMANIA

PRACTICE THE ART OF WAR

by Tobias Schaaf



Gigalomania is a remake of the old Mega lo Mania game originally available for Amiga and DOS. In this game you play some kind of god, and you have to manage your civilization, bringing them from the stone age to nuclear power. You start with your own fortress and have to use your resources to invent new weapons and defences and use them to defeat your enemies. This starts as simply picking up stones for use as throwing weapons, all the way up to jet fighters and nuclear missiles. Gigalomania is very similar to this, but with more modern graphics. The gameplay is still the same: research new technologies, harvest resources, and slay your enemies. The game is very fascinating and fun to play.

You can get Gigalomania from my repository either from the jessie/main or the wheezy/main package list, and it should work on any system with SDL2 support:

```
$ sudo apt-get install \
  gigalomania-odroid
```

If you want to install it directly, download the .deb file from http://bit.ly/1ZZaKOc and install it with these commands:

```
$ cd ~/Downloads
$ sudo apt-get install gdebi
$ sudo gdebi ./gigalomania*.deb
```

You can launch the game by typing the following in a Terminal window:

```
$ cd /opt/gigalomania/gigalomania
$ ./gigalomania gfx
```

Check out a gameplay video at http://bit.ly/1LbZran, or post comments, suggestions and questions on the original forum thread at http://bit.ly/1Z4b9yc.

Mega lo Mania original title screen

Gigalomania lets you build jet planes

Gigalomania gameplay (above and below)

USB CLEANING

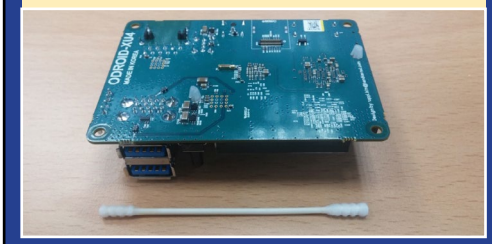
A QUICK AND EASY FIX FOR DISABLED USB PORTS

edited by Rob Roy

The USB host ports on any computer can sometimes become dirty, rendering them unable to make good contact with the USB peripheral, so it's a good idea to occasionally clean them with a cotton swab and some alcohol. The following pictorial details how to safely clean the USB ports on an ODROID-XU4.

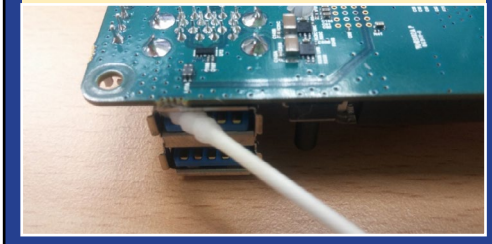
Be gentle, and make sure to allow the USB ports to dry thoroughly before using them. For comments, suggestions, and questions, please visit the original thread at http://bit.ly/1G8sgJe.

Prepare a cotton swab and alcohol and turn over the XU4 to expose the contacts



You will find 5 USB super-speed dedicated contact pads inside each port

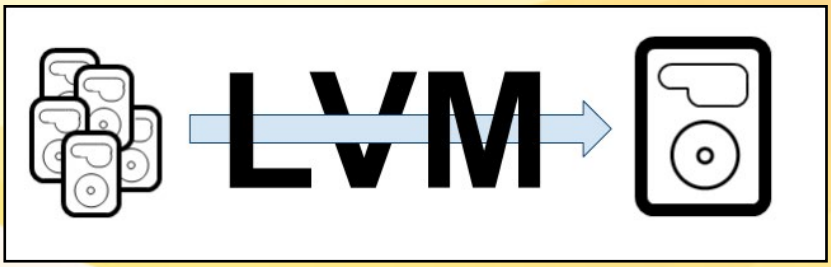
Clean the 5 pads with a sharp cotton swab and alcohol



LOGICAL VOLUME MANAGEMENT

MANIPULATING VOLUME GROUPS

by David Gabriel



After learning the basics from Logical Volume Management (LVM), and how to import and export your volume groups across different systems, I will now give you some tips on how to take advantage of the benefits of having LVM configured, whether you are a home user or a large business system administrator. Having LVM configured on your system is intended to make things not just easier, but also more reliable and resilient. And this becomes obvious when you have it on large environments with continuous growth.

Adding space

Let's say you already have your LVM created, and it is already full, so you need more space. However, you don't have any eMMC, SD card or external hard-drives available. But you do have flash drives, so why not use them? You can have as many devices you want on your volume group. And you don't even have to partition it if you don't want to. The following steps will prepare the disk to make it available to the LVM and add it to the volume group:

```
$ pvcreate /dev/sdb
$ vgextend rootvg /dev/sdb
```

Backing up

You have a very important database running on your ODROID, and losing all of its content would be a disaster. So, what do you do? You need a backup! There are many ways of keeping your data safe in case of any unexpected event, but I will show you how to backup using LVM snapshots. It is very simple, and you just need to run the following command:

```
$ lvcreate -L 20G -s -n snaplv /dev/rootvg/dblv
```

Once it completes, you can check your newly created logical volume (LV) with the following command, which will prepare the disk to make it available to the LVM and add it to the volume group:

```
$ lvs
```

The key here is the "-s" parameter, which will create a frozen image of the logical volume as it was at the time the snapshot was taken. The snapshot volume size does not have to be the same size as the original, but there should be enough space for the changes that will likely happen.

After that, you can mount the LV and copy it off the system anyway that you prefer. The snapshot may be taken with the filesystems mounted, so you don't have to worry about files being changed during the backup. It is very useful in production environments where you can not stop everything just for a backup, or need to deal with files that are not being backed up because they are in use. It is also a good idea to take a snapshot before major changes to the LVs, so that you will have a point in time from which to quickly recover in case anything goes wrong.

Removing disks

Imagine the following situation: You added a flash drive or an external drive to your LVM, but now, for some reason, you need it somewhere else. Or you just need to replace it, since it is showing signs of failure, or just want to change it for a better and bigger one. The problem is that, once you added it to a volume group, the system will coordinate how data is saved on it, so you can't just unplug it from the board or it will break your entire LVM.

To address this situation, you can move data between the disks on the system so that you can free up a specific one in order to remove it from the volume group. There are two options of doing this. First, if you have enough space available

on your volume group, use the following command:

```
$ pvmove /dev/sda2
```

Be patient, as it will take a while for this command to complete. It will move all contents from /dev/sda2 and spread them to other disks on the same volume group. After the command has completed, you can use the following command to remove the disk from the volume group:

```
$ vgreduce rootvg /dev/sda2
```

If you do not have space, you will have to add a new disk prior to moving things around. Use the commands shown earlier to make the new disk available to the LVM and add it to the volume group. Then, you can move the data from the old disk directly to the new disk with this command:

```
$ pvmove /dev/sda2 /dev/sdb1
```

Notice that, no matter which way you choose, always ensure that you have a backup of your files in case any problems occur that might cause you to lose your data, such as a power outage.. Also remember that you can do the moving while the volumes are online, so it is not necessary to unmount any file system, although I recommend doing it when the disks are not too busy, so you don't risk impacting system performance.

Splitting a volume group

It is always a good idea to have your volumes organized, and have different file systems for different purposes, so that a problem with one of them does not impact the others. The same applies to volume groups: it is better if you create separate volume groups for system files, different applications, and databases. However, if you didn't organize the volumes at first, and now everything is disorganized, there is a solution. We can take advantage of LVM one more time and split the volume groups.

You don't have to add any new disks to your LVM if there is already some space available. To check, use the "pvscan" command in order to show all of the physical volumes and their respective available space. Choose a disk to which you would like to move a new volume group, then use the "pvmove" command in order to free up space on it, which subsequently reorganizes the data across other disks. Finally, run the following command:

```
$ vgsplit rootvg appvg /dev/sdb1
```

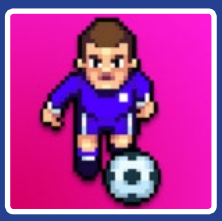
This will create a new volume group using /dev/sdb1. The effect is the same as if you had used the "vgreduce" command in order to extract it from the original volume group (VG), then used the "vgcreate" command in order to create the new VG. The "vgsplit" commands merges these two steps into a single one in order to make the process easier for the administrator to perform. After that, just create your logical volumes and file systems as you wish.

Logical Volume Management provides users with a high view of the system storage and how it is handled, giving more flexibility and friendlier tools. It is easier than it looks, and anyone can use it in order to keep their storage organized and allow more flexibility in managing disk space.

TOUCHSCREEN AT ITS BEST

TIKI TAKA SOCCER IS THE PERFECT GAME FOR EVERY TOUCHSCREEN USER

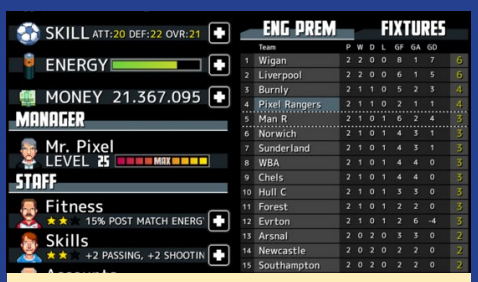
by Bruno Doiche



If you have attached your ODROID to a touchscreen in order to get Android up and running in all of its glory, don't miss the opportunity to download Tiki Taka Soccer. It is an amazing little game that uses the best of what a soccer game with touch commands has to offer, and will make you forget all those other games that are FIFA or PES derivatives. It is truly a soccer video game done well!

https://play.google.com/store/apps/details?id=com.timconstant.tikitakasoccer&hl=en

This is a wonderfully fun soccer game that will make real use of your touchscreen



As with all soccer games, managing your team to championship glory is a must.

LINUX GAMING

OPENGL COMPATIBILITY USING GLSHIM

by Tobias Schaaf

As we all know (or should know), ODROID devices do not support OpenGL but only OpenGL ES, which is an optimized subset of OpenGL designed for embedded systems. This means that quite a few programs and games cannot run on the ODROID, since they require the OpenGL library. This is quite frustrating, since ODROID devices and OpenGL ES could run these programs just fine, but many developers do not bother with porting programs to OpenGL ES, and migrate their games to Android or iOS instead. Some programs are mature enough that their development has already been completed, and OpenGL ES was never a topic for these programs. There are quite a few programs and games that we will miss because of this incompatibility. But don't worry, there is still a solution for it, and it's called GLshim.

What is GLshim?

GLshim is a OpenGL to OpenGL ES wrapper, which means it's possible to translate certain OpenGL functions into OpenGL ES function, which allows OpenGL to use OpenGL ES acceleration for doing the same (or similar) tasks. GLshim effectively allows to run OpenGL programs and games with OpenGL ES acceleration on the ODROID.

How GLshim works

OpenGL programs require a libGL.so library, which normally comes either from the graphics card vendor or from the Mesa project, if the programs are Mesa-compatible. Programs load this library and use functions declared in it in order to create certain graphical effects. The GLshim library is also packaged as a libGL.so file, which provides some of the functions that other libGL.so files provide.

Instead of loading the normal libGL.so, like the one from Mesa, the programs are told to load the GLshim library. This can be done by either pre-loading the library, or to telling the program where the library is installed. It can also be done when compiling a program.

Installation

If you're using my images or repositories, and have my libgl-odroid package installed, you will find the required library under /usr/local/lib/. Otherwise, you can install GLshim on your system by adding my repository to your image:

```
$ cd /etc/apt/sources.list.d/
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-testing.list
$ wget -O- http://oph.mdrjr.net/meveric/meveric.asc | apt-key add -
$ sudo apt-get update
$ sudo apt-get install libgl-odroid libglues-odroid
```

Or install the glshim packages manually with the following commands:

```
$ cd ~/Downloads
$ mkdir glshim
$ cd glshim
$ wget http://oph.mdrjr.net/meveric/other/freeorion/libgl-odroid_20150922-1_armhf.deb
$ wget http://oph.mdrjr.net/meveric/other/freeorion/libglues-odroid_20140903-1_armhf.deb
$ sudo apt-get install gdebi
$ sudo gdebi libglues-*.deb
$ sudo gdebi libgl-*.deb
```

Finally, link the Mali drivers (on the XU3 and XU4, use libmali.so instead of libMali.so):

```
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libEGL.so
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libGLESv1_CM.so
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/lib-
```

```
GLESv2.so
```

General usage

As an example, the following command runs the graphics demo called "glxgears" using GLshim:

```
$ LD_LIBRARY_PATH=/usr/local/lib
glxgears -fullscreen
```

By including the LD_LIBRARY_PATH variable, you can tell the system where to search for required libraries. This is normally the best method of using GLshim. It tells a program that it should first search for all required libraries in the /usr/local/lib folder, and only if it can't find a required library there, it will use other paths defined for libraries. That way, a program can load the GLshim library from /usr/local/lib and subsequently load the other libraries that it needs from the normal system paths.

Preloading GLshim

With the LD_PRELOAD variable, you can tell the system to load a specific library (or more than one) before starting an application. That way, if a program requires a library or certain functions, it uses the function provided by the preloaded library rather than a linked library. This is helpful if the program is hardcoded to use a certain library in a set path and won't load from any other path.

In the case that the previous on-demand method of using the LD_LIBRARY_PATH variable doesn't work, this is the best alternative for loading GLshim. The following command also runs the graphics demo called "glxgears" by preloading GLshim:

```
$ LD_PRELOAD=/usr/local/lib/lib-
GL.so glxgears -fullscreen
```

Compiling with GLshim

When you compile a program, it links itself to the required libraries. One of the libraries for OpenGL library is libGL.so, as previously mentioned. This is normally included by using the parameter -lGL, which will link against the libGL.so library installed on your system. In most cases on the ODROID platform, this is the MESA libGL.so library. There are a couple of ways to change this.

First, you can exchange the -lGL entry for the full path of the GLshim library (/usr/local/lib/libGL.so), which works most of the time. Another method is to change the path where the program is supposed to search for libraries. In your linking parameters, you might find something like this:

```
-Wl,-rpath,/usr/lib/arm-linux-
gnueabihf
```

These parameters tell the program that it should look for the following libraries in the standard folder of your Linux system, which is where the Mesa drivers are located. You can tell it to instead search for libraries in /usr/local/lib first before trying other paths, and therefore find the GLshim library before it finds the Mesa library, by changing the entry to the following:

```
-Wl,-rpath,/usr/local/lib
```

Another option is to try statically linking against libGL.a (located in /usr/local/lib) instead of dynamically linking against libGL.so, so that all functions needed to run your OpenGL program are already part of the program itself. The benefit of this is that if you know that a program is working with a certain version of GLshim, and you statically link against that version, you can be sure that your program will always work no matter what version of GLshim you have installed on the system.

However, this method also has some drawbacks. The size of your binary will increase, which might not be an issue, but the more libraries you link statically, the bigger the binary becomes. For some programs, this ends up creating binaries that are several hundreds of megabytes in size. Also, if you link GLshim statically, and a newer version of GLshim runs better with your program, it will still always use the older version, since it's embedded in the program, despite having a newer version of GLshim available.

Additionally, this method only works if all of the functions used for OpenGL are embedded in the code and are not invoked through other libraries such as SDL. Only the functions which are used will be taken from the library if you statically link against it, and if no function is declared in the program itself, but are dynamically generated from SDL, for example, the static link will not work.

Compatibility

You cannot run every single OpenGL program by using GLshim. Right now, it's limited to OpenGL 1.x calls, whereas OpenGL is up to 4.5 at the moment (http://bit.ly/1jUtj51). There are two major versions available with GLshim. One is from the original developer @lunixbochs, who started the work on GLshim. His work can be found in his git repository at http://bit.ly/1jcznWy. Another developer called @ptitSeb has improved the original version of GLshim in his fork at http://bit.ly/1KAOBur.

Both versions of GLshim work on ODROIDs, but are a little different from each other. @lunixbochs aims to be very accurate by following the OpenGL standard and MESA definition, and his work is meant to be the most stable version of GLshim. @ptitSeb tries to have as many features available as possible, and therefore has a wider range of game support, but while doing so, breaks support of some other games that work with the GLshim version by @lunixbochs.

To help in deciding which version to use, @lunixbochs has created a document showing his progress on GLshim development, including which functions are implemented at http://bit.

ly/1Gqof2M. This spreadsheet shows that his version is already very advanced when it comes to OpenGL 1.x support.

The limits of GLshim are obvious, since only OpenGL 1.x functions are supported right now, and not even all of them are available yet. This means that higher functions of OpenGL 2.x 3.x 4.x are not supported, and these versions are where the real "magic" happens like support of shaders and other "fancy" stuff. GLshim is still limited, and won't allow you to run every OpenGL program on your system. But if a program uses mostly OpenGL 1.x functions, there is a good chance it will run under GLshim and therefore on the ODROID.

Figures 1 and 2 - FreedroidRPG a isometric styled RPG game similar to Fallout or Baldurs Gate

Figures 3 and 4 - Secret Maryo Chronicles a very cute Super Mario Clone using OpenGL for many effects like falling leaves

Figures 5 and 6 - Hedgewars is a very nice Worms clone with unique weapons and extras. You can even play it on LAN or via Internet with your friends

Classic games

You can find a selection of titles that have already been published in the forums at http://bit.ly/1RWWX5n (you must be logged into the forums to view them). Some examples are OpenCPN, a Chartplotter and GPS navigation software package, as well as many games such as FreedroidRPG, Super Mario Chronicles, and Hedgewars, most of which are very impressive.

These games run very fast, and you can see the 3D acceleration working to produce beautiful graphics. The background effects, like falling leaves, snow, and the different lighting effects make these games very fun to play. Other games besides those made for native Linux work perfectly fine also. You can play remakes of classic DOS and Windows games that were ported to OpenGL on ODROIDs, such as Duke Nukem 3D, Shadow Warrior, and even Quake 2. While games like Quake 2 and Duke Nukem 3D push the device to its limits when rendering 3D polygons, many games use OpenGL for nice effects and smooth graphics.

Figures 7 and 8 - Duke Nukem 3D on ODROID in high resolution (top) and even fully 3D polygon models (bottom) work well, thanks to GLshim

Figure 9 - Even Quake 2 with high definition textures on an ODROID

Figures 10 and 11 - Games like Valyria Tear, although purely 2D have stunning graphics and special effects, made possible by OpenGL

Figures 12 and 13 - More effects in Valyria Tear, such as swirling particles on save spots (top) and lens flare effects of windows and other shiny objects (bottom) as well as transparent speech bubbles

Valyria Tear

Valyria Tear is a very nice RPG game, with wonderful music and graphics. It's still in development, but version 1.0 which is in my repository offers the fully playable first episode of the game. The title screen and short introduction of the game Valyria Tear show off the beautiful effects that you can expect from OpenGL. Floating clouds, lightning strikes and a spinning vortex are just a few of the amazing effects in Valyria Tear. Although many features are still being written, Valyria Tear offers a complete game experience, with fully working maps, menus and battles and all the things you can expect from a classic:

- ~50 maps done
- 18 32x32 Tilesets used
- 2 and half playable characters
- 19 different encounterable enemies
- A dozen NPCs, skills, spells, and different items
- Status effects can be applied on characters and foes
- Full joystick support
- The game is fully translated in 6 languages, and more are incoming
- Three UI themes
- Main and secondary quests, hidden treasures and traps
- All in all, something like 7-10 hours of gameplay
- ~140'000 lines of c++ code and lua script

It's a very nice game, and for version 1.1 the developers switched to SDL 2 and OpenGL 2.0, so it could be that the next versions of the game are no longer working on ODROIDs, since OpenGL 2.0 support is not yet a given in GLshim. Recently I was able to get even more great games to work, such as FreeOrion, a game greatly inspired by the Master of Orion series.

Figure 14 - The title screen of FreeOrion, which is a very complex strategy game now available on ODROIDs as well

Aquaria

Recently, I was able to get a very known commercial game running on the ODROID using GLshim called Aquaria, which was released in 2007, and is one of these so-called "indie" games. This game is really awesome. The music in this game is very deep, and you can just keep listening to it for hours. The graphics are beautiful with really cool effects. If you haven't heard of this game, you really should check out the trailer at http://bit.ly/1QUQOpw.

The game is very peaceful and rich with different surroundings that feel like new worlds. If you like action adventure games and do like a certain comic style of games, then this game is a definitely

Figure 15 - Aquaria title screen, where you can already hear and feel the magic and beauty of the game

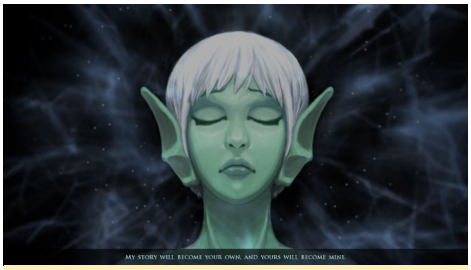
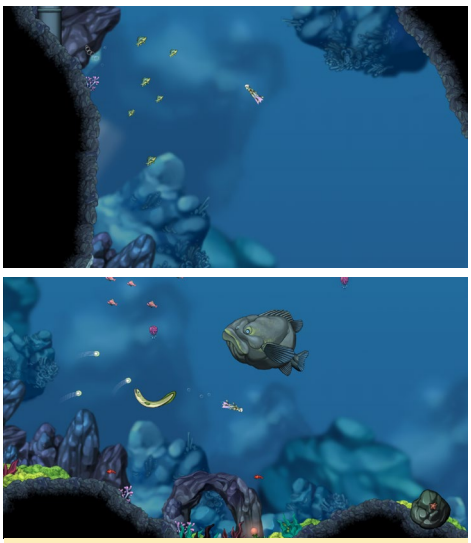


Figure 16 - Naija, the main character of Aquaria, with impressive voice acting from the lovely Jenna Sharpe



Figures 17 and 18 - Beautiful game scenes with fading backgrounds and lots of colors, different kinds of water effects, plants and animals

Figure 19 - At some points in the game you even can get to the surface seeing more than just water all around you. Just look at these wonderful colors!

must have for you, and now you can play it on your ODROID as well.

Aquaria also offers different kind of mods, which can be downloaded directly from within the game.

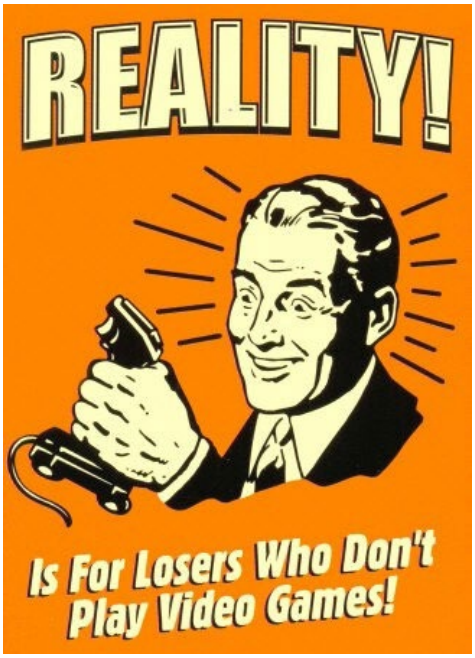
I've played this game on the U3 and XU4, and both run fine even in 1080p, although the XU4 feels more responsive than the U3. All of the pictures in this article are taken directly from a ODROID-XU4, so you can see how the game actually looks on the ODROID platform.

Conclusion

I hope the GLshim project continues to progress as well as it has so far. The main contributors, @lunixbochs and @ptitSeb, are doing awesome work with this project, and I hope we will be able to port many more games thanks to GLshim.

I anticipate that we will have good OpenGL ES 2.0 support soon, along with some OpenGL 2.x functions, so that we can use shaders and other fancy effects via GLshim on ARM boards.

I will keep looking for more games to port to the ODROID using GLshim, which I routinely post in the Games and Emulators section of the ODROID forums at http://bit.ly/1jUwR7u.



COMMUNITY WIKI

CONTRIBUTE TO THE EXPANDING ODROID KNOWLEDGE BASE

by Rob Roy

Hardkernel has recently set up a great resource for ODROIDians to contribute their knowledge to a community wiki, available at `http://wiki.odroid.in`. It is intended to complement the official Hardkernel wiki at `http://bit.ly/1R6DOgZ`, and is useful for posting your tips, community image links, projects, and anything else that might be beneficial to the Hardkernel community.

If you'd like to participate, click on the "Request Account" button in the top right, and include your ODROID forum username in the "Personal Biography" section. For comments, questions and suggestions related to the new wiki, please visit the original forum thread at `http://bit.ly/1QDMNoT`.

LFTP AND CRON

SERVER SYNCING MADE EASY

by Andrew Ruggeri

As with our LVM series, this article is a reminder that the best system strategy is to always have a good backup!

The ODROID and other dev boards based on high-end SoCs have been used in various types of servers (including web-servers, NAS, IoT Databases, media servers, etc.) for a while now. In all these usage examples, it is always a well needed feature to be able to have some form of backup or sync function to or from the server.

This guide will focus on two tools that can be used to help with the synchronization process.

LFTP: the first tool we will be looking at is LFTP (`http://lftp.yar.ru/`) which is a command line UNIX tool used for transferring files via FTP, SFTP, HTTP, HTTPS, and several other protocols. LFTP has the unique function of allowing segmented downloads which, depending on your use, can greatly increase transfer speeds.

CRON: is a useful UNIX program that can help schedule scripts and commands to run at certain periods.

As you might have guessed, we will be using cron to invoke a set of LFTP commands to help us sync with the server. In this guide we will be looking at 'lftp' for downloading several folders and files to/from a server. This functionality is useful in a situations such as pulling back-up tables off a database server to save locally, grabbing a new connect from some external server to pull locally for a HTPC, or just about any backup scenario you might imagine were new content needs to be backed up.

LFTP

LFTP on it's own is a capable tool, and very useful as well. For this guide we are going to assume that the remote server has a local SFTP server running as well. Note that LFTP supports many different protocols other than SFTP, and the steps below will be very similar for other protocols. The first thing we will need to do is to install LFTP via apt-get as follows:

```
sudo apt-get install lftp
```

Basics

To connect to an SFTP server with an user name of 'odroid', and an ip address of 192.168.0.101, in the terminal window you would type the following:

```
lftp sftp://odroid@192.168.0.101
```

Pretty simple right? If we examine the command, we are instructing lftp to make as SFTP connection, with a user name of 'odroid', to a server located at (@) 192.168.0.101

Once you connect, you will be prompted to provide a password. If all goes well you will then move on to something which looks similar to a normal command prompt. You will not have the full range of bash functionality, but you still have the major ones such as cd, ls, mkdir, rm, mv, etc.

To send commands in the local machine during your LFTP session, append the command with '!', so in order to change the directories locally the command would be '!cd', and to print the local working directory the command would be '!pwd'.

Downloading

To download a copy of a file from the server to the working local directory, we will use 'pget' followed by the filename as shown below in the command example and Figure 1 next page:

```
pget myFile.dat
```

Use mirror follow by a directory name in order to download over the directory. The following command will download the 'test' folder and all of its contents to a working directory.

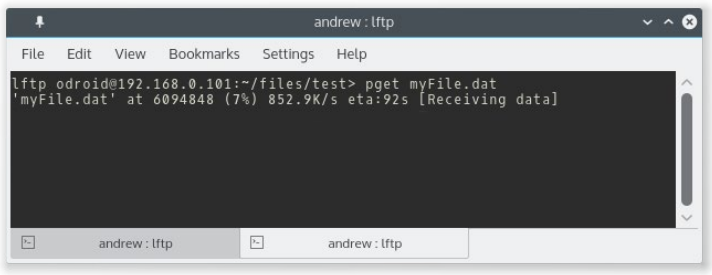


Figure 1 - pget command

```
mirror test
```

A download or upload can be stopped with ctrl+c, and resumed from where it left off with the '-c' argument as follows:

```
pget -c myFile.data
```

Uploading

To upload a single file to the server, use the 'put' command followed by the filename as shown below in the command example and Figure 2:

```
put myFile.dat
```

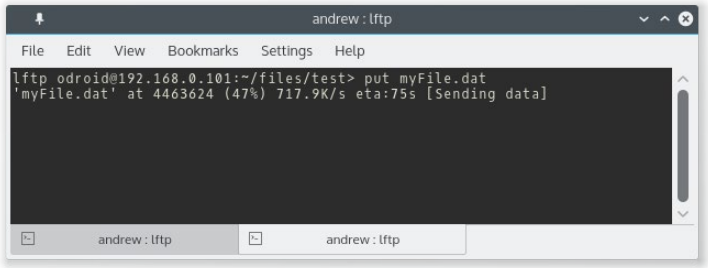


Figure 2 - put command

To upload a directory, the command would be 'mirror -R' followed by the local directory name as follows:

```
mirror -R test
```

Segments & Parallel Downloads

One major feature of LFTP is the ability to have segmented and parallel downloads which can greatly speed up transfer rates, especially when you are moving data to a server outside of your local network.

For a single file using the 'pget' command, the argument would be '-n' followed by the number of segments. For example, to have 5 segments you would use the command example shown below and in Figure 3:

```
pget -n 5 myFile.dat
```

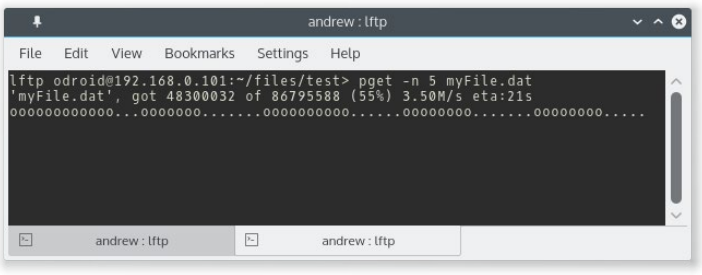


Figure 3 - pget command with multiple segments

For a directory transfer, use 'mirror', followed by '—use-pget-n=' and the number of segments for each file. For example, to transfer the 'test' directory in 7 segments you would use the following command:

```
mirror --use-pget-n=7 test
```

By setting the '-n' argument, we can download a single file in multiple parts. In addition to this, LFTP will also allow us to download multiple files at the same time. When we use the mirror command, it will download a single file within that directory at a time. With the parallel argument, we can have the mirror command download multiple files at a time.

Parallel downloads with mirror are invoked by adding '-P' and the number of files to download. For example, to download 3 files at the same time you would use the following command:

```
mirror -P3 test
```

To use parallel and segmented downloads at the same time using a mirror download, you would use the following command:

```
mirror -P3 --use-pget-n=7 test
```

Jobs

LFTP allows you to queue up jobs, as well as have jobs be run in the background, which allows you to perform commands to the session such as cd, ls, mkdir, etc.

To add a download or upload job to the queue, simply place 'queue' before the command. For example, to add segment 'pget' to the queue, you would use the following command:

```
queue pget -n 5 myFile.dat
```

Jobs in the queue can be turned on / off with a start and stop commands as follows.

```
queue stop
queue start
```

To see a list of all the jobs in the queue use the 'jobs' command as shown in Figure 4 below:

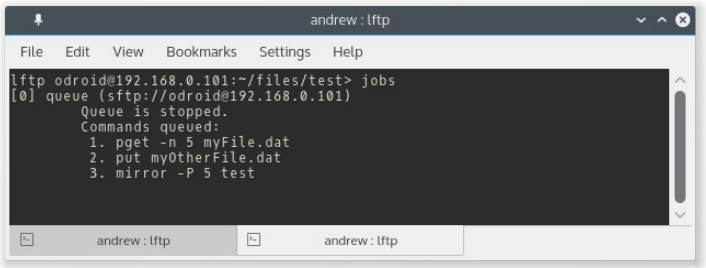


Figure 4 - Listing jobs under LFTP

The 'jobs' command will print out a lot of useful information including the queue status (stopped, in this case), a list of all jobs in the queue (3 listed in this case: pget, put, mirror), and the currently active jobs (queue is stopped, so for this case there are none).

The 'kill' command can be used to remove queued jobs. The 'kill' command is followed by the job number shown in the jobs info. For example to remove the 'put myOtherFile.dat' job, the command would be:

```
kill 2
```

Bookmarks

Bookmarks are a very easy way to save and recall a user name, password, remote server address, and remote server directory.

The first step, if you so choose, is to enable passwords to be saved in bookmarks. Without setting this you will be asked to enter you password each time, however attributes such as server address, user name, and remote directory will still be saved. To enable saving of passwords in a bookmark, use the following command:

```
set bmk:save-passwords true
```

Now you will need to create a new bookmark logon for the server you which to bookmark. You will also need to move the directory on the server you want to set. For example, with my bookmark I want to auto-logon to my sftp server at 192.168.0.101, with the username 'odroid', to the '~/files/test' directory. First I connect, then cd to 'files/test/'. Next I will give the bookmark 'add' command followed by the name of the bookmark, which in this example is 'odtest'. For this example, the commands would be as follows:

```
lftp odroid@192.168.0.101:~> cd files/test/
lftp odroid@192.168.0.101:~/files/test/> bookmark\
add odtest
```

To use the newly created bookmark from a local terminal session, simply type 'lftp' followed by the name of the bookmark as shown in Figure 5 below.

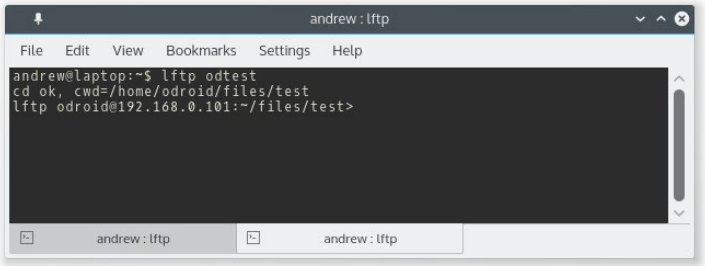


Figure 5 - Using bookmarks on LFTP

CRON

Cron is a great tool with a fairly easy learning curve, and comes preinstalled with most ODROID Lubuntu builds. If Cron is not available, it can be easily installed with 'apt-get'. Cron works by editing the crontab config file, which dictates when a command or script should run.

To start editing a crontab, we'll use the 'crontab -e' command. This will open up your system's default text editor (which for Kubuntu would be the 'nano' editor) as shown in Figure 6 below.

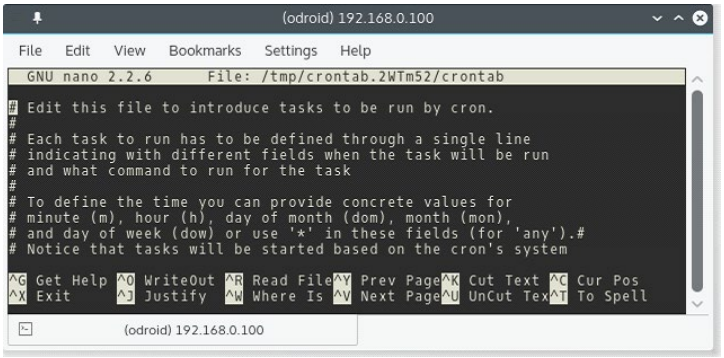


Figure 6 - The almighty CRONTAB

You will first need to scroll to the bottom of the file since this is where we will be begin adding our commands. To add a command to be executed, we will need to add a new row. Each row has 6 parts to it, from left to right as shown in the table below:

Each of part of the row is separated by a space. Let's look at an example for setting up a script '/home/odroid/foo.sh' to run every day at 10:42am. Refer to the command below and to Figure on the previous page for details.

```
42 10 * * * /home/odroid/foo.sh
```

In this command, the first part sets the minute to 42, the second part sets the hour 10, and the last part (part 6) sets the path to the script. Parts 3, 4, and 5 have been set to '*' which is one of the special symbols you can add. Please refer to the table and example commands below.

Symbol	Example	Definition
*	*	All values
*/<X>	*/5	Every <X>, or skip every X
<X>-<Y>	5-7	A range from X to Y
<X>,<Y>,<Z>	2,5,6	Used list multiple values

```
# run every 10 min
*/10 * * * * /some/command.sh

# run 3 time each hour: X:00, X:15, and X:30
0,15,30 * * * * /some/command.sh

# run at the start of every hour from 6 - 8am
0 6-8 * * * /some/command.sh
```

Another useful command you might wish to use is 'crontab -l', which will display the complete crontab config file.

Batch Script

The following example is a batch script which uses LFTP to connect to a bookmarked server. This script will cycle through each folder in the 'sync_folders' list, and download any new files found. The script will also allow only one instance of the batch script to be run at time, which is selected by creating a 'lock_file' in /temp. This is not a one-size-fits-all script however, and will need to be customized to fit your needs.

```
#!/bin/bash

# bookmark set to server with username and password
bookmarkName="odroid"

# directories to synced
remote_dir='files/data/'
local_dir="/home/andrew/backup/data/"

echo "Starting $bookmarkName Sync"

# ------------------------------------------------
--------------------
# Create a list of only the directories to be synced
# ------------------------------------------------
--------------------
declare -a sync_folders=(
"Eagle Projects"
"Images/Diagrams"
"Images/Graphs"
"MongoDB Backups"
"Thesis"
"Test Logs"
"Test Data")

# create lock file
base_name="$(basename "$0")"
lock_file="/tmp/$base_name.lock"
trap "rm -f $lock_file" SIGINT SIGTERM
if [ -e "$lock_file" ]
then
    # Only allow one instance of this script to run
at a time
    echo "$base_name is running already."
    exit
else
    touch "$lock_file"
    for current_folder in "${sync_folders[@]}"; do
        cd "$local_dir$current_folder"
        lftp $bookmarkName << EOF
        echo "Connected to $bookmarkName"
        cd "$remote_dir"
        set mirror:use-pget-n 9
        mirror -c -P5 "$current_folder"/.
        echo done
        quit
        EOF
    done
    rm -f "$lock_file"
    trap - SIGINT SIGTERM
    exit
fi
```

BUILD A CUSTOMIZED SPAM FILTER

REGAIN CONTROL OVER YOUR INBOX

by Pascal Pucholt

Not having to deal with SPAM is a tech dream, right?

Spam or "Unsolicited Bulk Email" can outnumber the real "ham" emails send to your Email address by several orders of magnitude. If you have your own domain name and also use it for your emails, you might be stuck with a mail server that does not do a very good job of handling this flood of spam. To solve this, you could forward all incoming mails to google or the like and let them do the filtering (and use your mails to present you with the "relevant" advertisement and also store them in their database so that NSA and others have central access to them). However you could take the techie approach and let your ODROID handle the spam problem to keep your Emails under your own control.

In this article I will present a method for filtering out the spam mail from your personal mails on a mail server. The mail server itself is not run on your ODROID computer. This method is based on the IMAP protocol and utilizes the widely used spam detection software SpamAssassin. So it remains universally applicable as long as you have access to your email server through IMAP, and the usage of SpamAssassin will ensure that proven technology is used to separate between the good "ham" and bad "spam" mails.

In my own test, the filter drastically reduced the amount of spam Emails in my inbox from several hundred per day to only a few, and I hope that the detection will get even better with more training.

Figure I: Schematic view of the spam filter

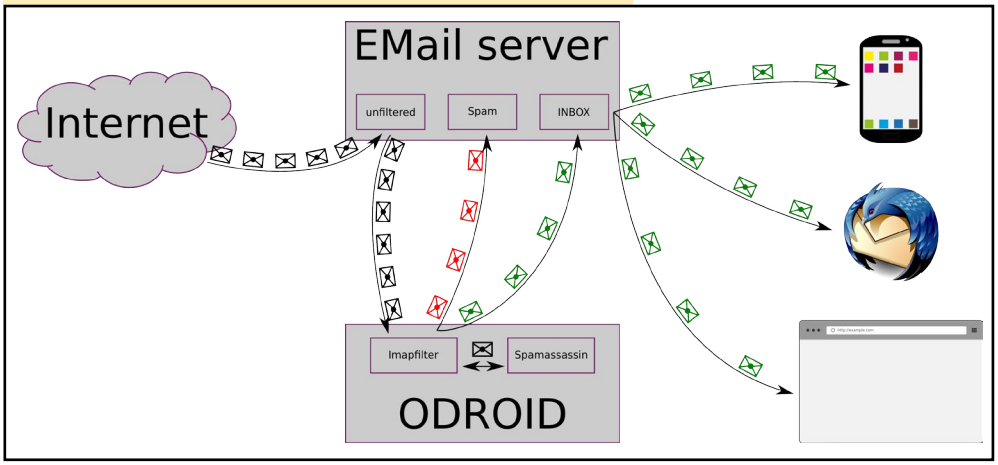


Figure 1 illustrates the spam filter preventing a non-classified email from reaching your server from the internet, to your reading of only the real mails (green) on any device connected to the mail server.

Requirements

An ODROID computer (I use a C1) running an Ubuntu Linux image
An Email server that you can access through IMAP

Preparing your Mail server

You will need to instruct your mail server to send all your incoming mails into a dedicated folder for unfiltered mails. It is easiest to just name it "Unfiltered". First you will need to create this folder on the server, either by an Email client like Thunderbird, or through a web interface that you might have. Next you will need set up a 'rule' on the mail server to forward all incoming mail to that folder. The method required to set up the rule will depend on the mail server software being used, and you need to consult the documentation for your own server. On most systems, the setting you need to look for will be called 'filtering' or something similar.

Installing software

This spam filter relies on a software package called IMAPfilter in order to communicate with your mail server, and on SpamAssassin for the actual filtering. These packages can be installed through the Ubuntu package manager with the command:

```
sudo apt-get install imapfilter spamassassin lua-posix
```

The configuration files provided with this article are intended to be used in an environment with a specific user account under which the programs are run. To create this account, run the command:

```
sudo adduser spamfilter
```

and provide the script with the requested details. This is also where you will set the password for the new user account.

You can then login into your ODROID computer using this account, and download the configuration files from github with the following commands:

```
sudo su spamfilter
git clone --recursive https://github.com/tux2000/
imapfilter-tools.git ~/.imapfilter
```

To add the details of your mail server, you will need to create a new version of the account configuration template file. With the last command in the list below, you make this file readable only by that user account, and protect your credentials from being read by other users:

```
cd ~/.imapfilter
cp accounts.lua.sample accounts.lua
nano accounts.lua
chmod 600 accounts.lua
```

In this file you need to add the address of your mail server, your username and your password.

The template looks like this:

```
account1 = IMAP {
        server = 'your.mail.server.here',
        username = 'user',
        password = 'password',
        ssl = 'ssl3',
}
```

At this point, spamfilter should already be working. You can test it by executing 'imapfilter -v' from within /home/spamfilter. If there are problems with connecting to your mail server, you will get error messages and the program will crash. When you have fixed all of the problems, the program should go into daemon mode and periodically let you know that everything is good by printing "* OK Still here" on the screen.

To start spamfilter automatically at machine boot time (and to restart it if it crashes due to communication errors with the server), you will need to copy the file 'imapfilter.conf' to the init folder with the command below:

```
sudo cp /home/spamfilter/.imapfilter/imapfilter.conf /
etc/init/
```

After this, you will be able to start the spamfilter daemon through the upstart system with this command:

```
sudo service imapfilter start
```

The daemon will log its output to /var/log/upstart/imapfilter.log so you can have an eye on it with the command:

```
sudo tail -f /var/log/upstart/imapfilter.log
```

Training the spam filter and tweak settings

SpamAssassin needs at least 200 ham and 200 spam messages to enable Bayesian filtering. Place 200+ spam messages in the Spam/False Negatives folder and 200+ ham messages in the Spam/False Positives folder on your email server and IMAPfilter will do the rest.

To tweak the behavior of SpamAssassin, you can change the parameters in /home/spamfilter/.spamassassin/user_prefs. Add your own customized scores for some tests to change the weight they get in the decision whether a mail is spam or ham, or to change the threshold for the classification as spam by changing the 'required_score' value. In my own configuration I reduced the 'required_score' value from the default of 5 to 4.2 so that the bayes filter can almost on its own classify a spam email when it is very sure of the classification. I have yet to receive a false positive for the classifying of an email as spam even with this setting. Documentation on which parameters are available to set in the user_prefs file can be found at http://bit.ly/1iVLeIv

Hash sharing: You are not alone

Spammers usually want to distribute their message as widely as possible, and often just follow a very simple approach by sending the same message to millions of addresses. So the chances are that both you and I will get the exact same spam mail. This characteristic can be utilized for detecting spam even if it has managed to pass through all of the pattern-based filters implemented in SpamAssassin. The basic idea is to use

the reports of spam mails from many users which have been stored in a central database, with SpamAssassin checking this database for every email. Two such databases which are easy to install on the ODROID system are the 'pyzor' and the 'razor' databases.

As an administrative user, you will install the programs to check the database with the following command:

```
sudo apt-get install pyzor razor
```

You will then need to change back to your spamfilter user as follows:

```
sudo su spamfilter
```

To set up the databases for Pyzor, you need to run the command:

```
pyzor discover
```

which will set up a list of servers to connect to.

To check that SpamAssassin is actually using pyzor, you should run the command:

```
echo "test" | spamassassin -D pyzor 2>&1 | less
```

and the output should contain lines similar to this:

```
Oct  4 12:15:33.018 [8603] dbg: pyzor: network tests
on, attempting Pyzor
Oct  4 12:15:40.866 [8603] dbg: pyzor: pyzor is
available: /usr/bin/pyzor
Oct  4 12:15:40.869 [8603] dbg: pyzor: opening pipe:
/usr/bin/pyzor check < /tmp/.spamassassin8603t9Q0e9t-
mp
Oct  4 12:15:43.099 [8603] dbg: pyzor: [8657] fin-
ished: exit 1
```

Similarly, you will set up the razor program with the commands:

```
razor-admin -create
razor-admin -register
```

and test it with the command:

```
echo "test" | spamassassin -D razor2 2>&1 | less
```

and the output should contain:

```
Oct  4 12:17:11.738 [8662] dbg: razor2: razor2 is
available, version 2.84
Razor-Log: Computed razorhome from env: /home/spamfil-
ter/.razor
Razor-Log: Found razorhome: /home/spamfilter/.razor
Razor-Log: No /home/spamfilter/.razor/razor-agent.conf
found, skipping.
Razor-Log: read_file: 2 items read from /etc/razor/
razor-agent.conf
```

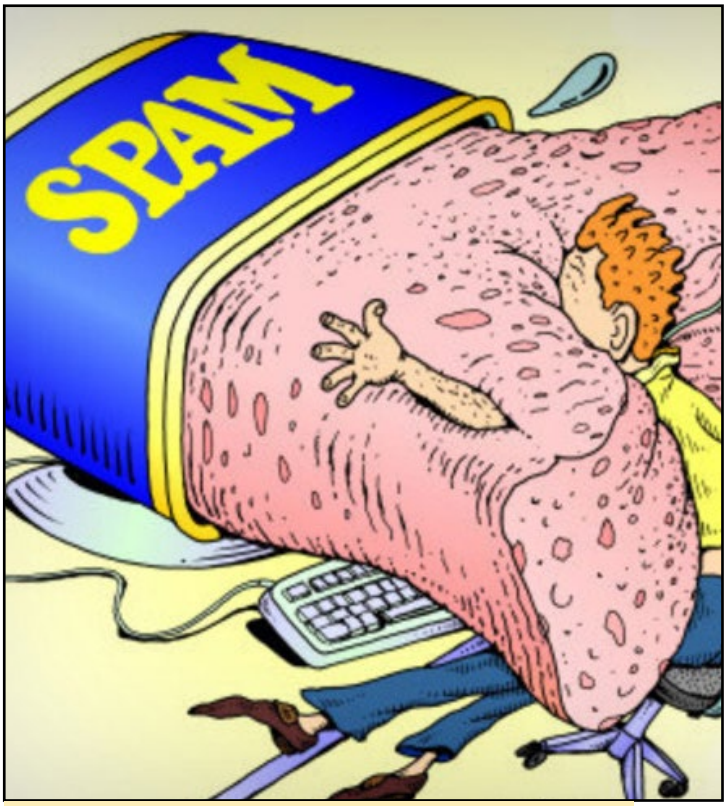
When you check the newly detected spam in your spam folder on the server, you will see that some of the mails were also found in either the pyzor and/or razor databases.

Conclusion

With the system presented above, you can use your own email server with all the advantages that it provides. Your ODROID will not contain the mail server itself, which places lower requirements on your internet connection and on hardware performance. The spam filtering is done separately on your ODROID system, which will allow you to tweak all settings to the values you need in order to prevent your emails from being detected as spam, while removing everything you don't want to see from your inbox.

Acknowledgments

The configuration of the IMAPfilter I provide within this article is largely based on work by Kyle Manna: (ref: https://github.com/kylemanna)



Now this won't happen to you anymore!

HIFI-SHIELD FOR THE ODROID-C1+

MAKE YOUR ODROID SOUND GREAT

by Justin Lee

We have developed a new product, specifically for audiophiles, called the HiFi-Shield. It is an advanced audio component that increases the signal-to-noise ratio and reduces noise level. It is available for purchase from the Hardkernel store at http://bit.ly/1M6UIXY for USD$39. We offer it as an add-on component to the ODROID-C1+ because, until now, when we wanted to enjoy high-resolution digital music on our ODROID, we needed to connect an external USB DAC like this:

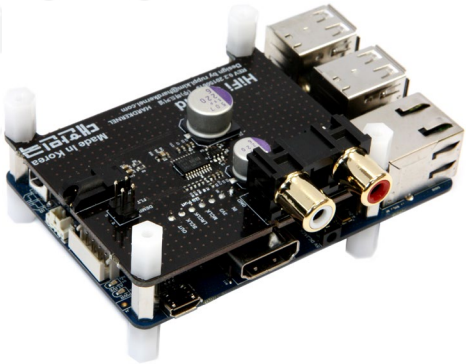
CPU -> USB Bus -> USB Cable -> USB to I2S converter -> I2S -> DAC -> Amplifier -> Speakers

That is quite a long way, isn't it? Add to it all the intrinsic issues affecting connection interfaces, and you will have a good idea of the issues that may arise. Ground noise, connection noise, cascaded clock jitters and complicated power rails can all affect the sound quality. Consider this scenario instead:

CPU -> I2S -> DAC -> Amplifier -> Speakers

This setup is much simpler and sounds better. However, we can make this chain even shorter by using an ODROID-C1+ I2S DAC setup. The advantages of this setup are:

- Ideal signal path which avoids extra interfaces such as USB and S/PDIF
- We're not using the troublesome USB bus of the single board computer, which can cause intermittent pop and clicks.
- Inexpensive and great sounding setup



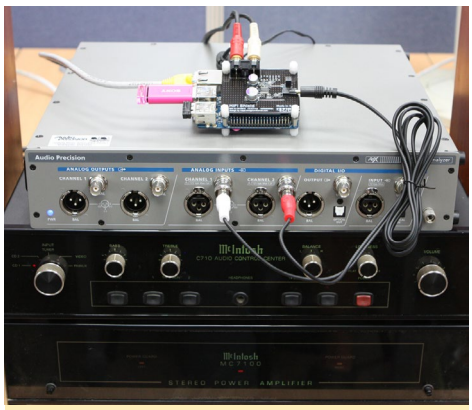
The HiFi-Shield for the ODROID-C1+ is designed for modern audiophiles

Technical Specifications

- The DAC chip uses a high-end PCM5102 chip from Burr-Brown company (now part of TI) which utilizes the I2S interface. It supports 16, 24, 32 bit audio formats with minimal distortion (-93dB) and ideal dynamics (110dB+), plus an amazing sampling rates of 384kHz.
- The output ports include gold-plated stereo RCA terminals and a 3.5mm audio jack.
- An ultra-low noise dropout regulator is coupled with two solid capacitors for the power supply, significantly reducing power supply noise and greatly increasing the signal to noise ratio.
- The I2S interface allows for direct decoding of the digital input to analog output using master clock synchronization.
- The PCB surface is comprised of gold-plating on top of 2 oz. of copper, ensuring signal continuity and reducing signal reflection and refraction.

Audio Quality

Some high-end USB DACs cost over USD$1,000, which have lower dynamic range and more signal noise than the C1+ HiFi-Shield system. To investigate the audio quality of the HiFi-Shield, we used an audio analyzer, and invested nearly USD$10,000 to purchase the industry standard "Audio Precision" equipment. WIth this equipment, we proved the HiFi-Shield's signal quality and innovative PCB design to the audiophile world.



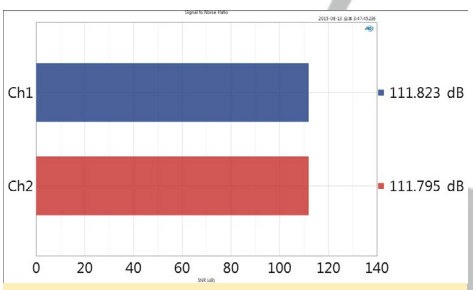
Hardkernel's in-house audio analyzer

The signal to noise ratio (SNR) is higher than 110dB and noise level is lower than -107dB as below two measured graph, so the dynamic range is also near 110dB, which is an amazing value.

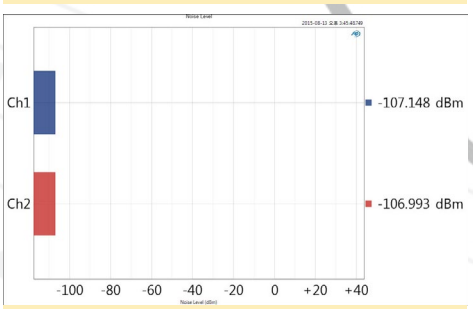
Hardware

There are two technologies responsible for the implementation of high quality audio on the HiFi-Shield:

The output swing level must be higher than 2Volt-RMS to achieve 110dB of dynamic range. 2Volt-RMS ground centered outputs is implemented with the integrated Negative-Charge-Pump.



Hi-Fi Shield's Signal-To-Noise Ratio (higher is better)



Hi-Fi- Shield Noise Output Level (lower is better)

TI calls it "DirectPath™" technology.

An ultra-low noise dropout regulator is coupled with two solid capacitors for the power supply, significantly reducing power supply noise and greatly increasing the signal to noise ratio. The I2S interface allows for direct decoding of the digital input to analog output using master clock synchronization.

It is ready to play with a high-fidelity sound reproduction system simply by adding an amplifier with speakers. Creative Gigaworks T20 is a good entry level speaker system to enjoy the high resolution audio with relatively affordable price range. A small form factor D-Class amplifier with bookshelf speaker system is also recommended.

A Creative Gigaworks T20 speaker system sounds great with the HiFi-Shield

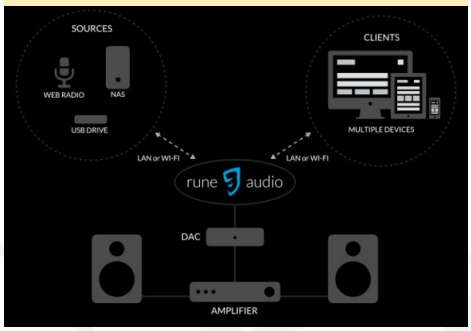
You can also use a D-Class amplifier with bookshelf speaker system

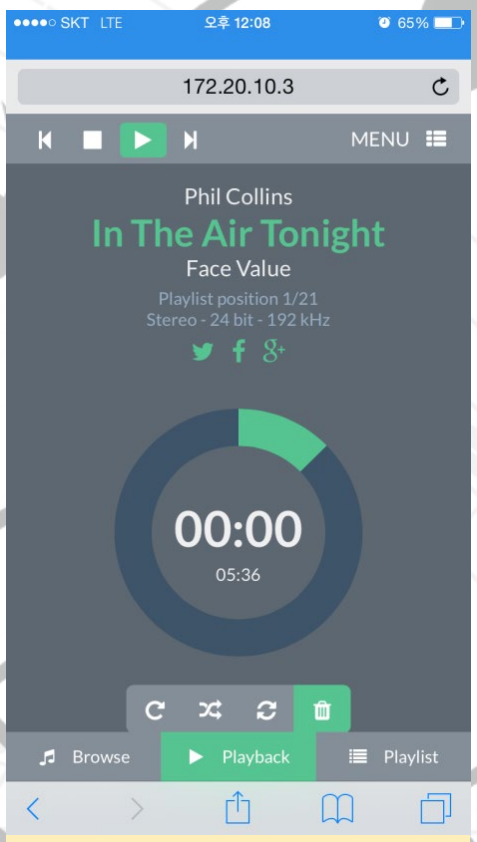
Software

There are two famous high-fidelity sound dedicated operating systems: Volumio and RuneAudio. You can use them in your living room, in your car, or at work. They can reproduce your digital music library from local USB drives or network mounts (NAS), and also plays any Internet stream, such as web radio. You can control the music playback via your smartphone, tablet or PC with very nice web UI interface. Please note that the HiFi-Shield works under Linux only, and drivers for Android are not available at this time.

Due to improvements made in the ODROID-C1+ in order to support high-fidelity audio, the HiFi-Shield is not compatible with the original ODROID-C1. However, if you want to use an ODROID-C1 with a 5.1 surround sound system, a USB-SPDIF adapter is available at http://bit.ly/1NUsxCa.

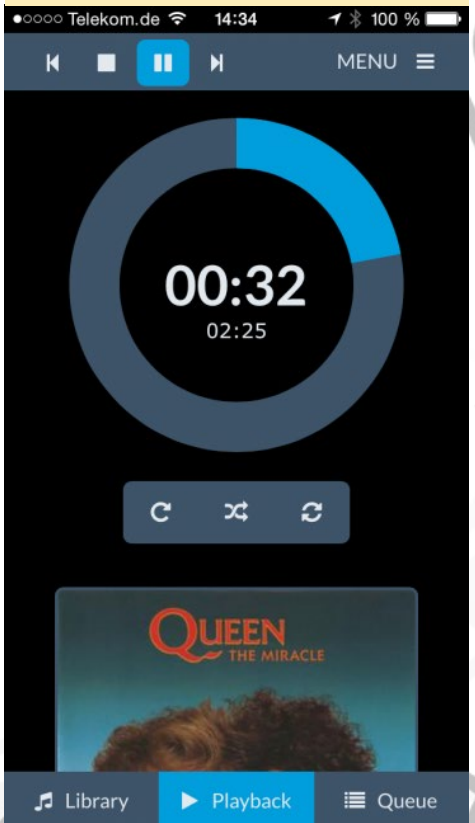
Typical RuneAudio audiophile setup





Volumio offers apps for iPhone and Android and recently announced that they have added the ODROID-C1+ to their list of supported boards (http://bit.ly/1Oec783)

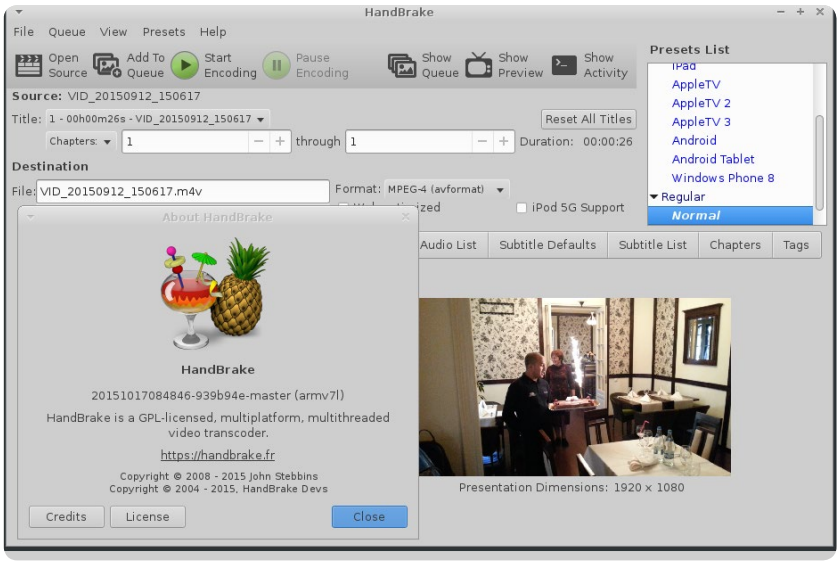
RuneAudio has partnered with Hardkernel to ensure that the ODROID-C1+ is 100% compatible with the latest version of the RuneAudio client



GENTLY RELEASE THE HANDBRAKE

TRANSCODE VIDEOS TO ANY FORMAT

by Adrian Popa



HandBrake is a tool for converting video from nearly any format to a selection of modern, widely supported codecs. The output formats include MP4 and MKV as containers with H264, H265, MPEG 2, Theora and V8 as video codecs and AAC/AC3/MP3/Vorbis/FLAAC as audio codecs. It comes with a set of presets to convert to formats supported by specific devices such as Apple and Android, but you can create your own presets to fit your needs. Although the ODROIDs are not designed for video transcoding, they can do the job if you're not in a hurry. Unfortunately, there is no prepackaged version of Handbrake for ARM platforms, so it is necessary to compile it from source.

When I first started writing this article, it was meant to be an overview of common configuration options and compilation problems, and what to do to fix all of them. However, it seems that all of the problems that I encountered had just two causes. First, I had failed to review the readme file, and kept stumbling on missing dependencies. Also, with the help of HandBrake's main developer, I was able to find a quick and easy way to go around all the problems by installing a missing package on my Ubuntu system, which made all the problems go away.

To compile HandBrake from source, you'll first need to get the latest development version from GitHub, which can be done as a regular non-root user in a Ubuntu environment:

```
$ git clone https://github.com/HandBrake/HandBrake.git
$ cd HandBrake
```

At this point, it is very helpful to read the compilation instructions to understand what you need to do, which is available in doc/BUILD-Linux:

```
$ less doc/BUILD-Linux
```

You can installed most of the required packages and build environment by running this command:

```
$ sudo apt-get install subversion cmake yasm build-essential autoconf libtool \
      zlib1g-dev libbz2-dev libogg-dev libtheora-dev libvorbis-dev \
      libsamplerate-dev libxml2-dev libfribidi-dev libfreetype6-dev \
      libfontconfig1-dev libass-dev libmp3lame-dev libx264-dev libjansson-dev \
      intltool libglib2.0-dev libdbus-glib-1-dev libgtk-3-dev libgudev-1.0-dev \
      libwebkitgtk-3.0-dev libnotify-dev libgstreamer1.0-dev \
      libgstreamer-plugins-base1.0-dev libappindicator-dev
```

In addition to these, you'll also need libtool-bin, which is not explicitly specified in the instructions but causes massive headaches if it's not installed, as described at http://bit.ly/1RUrmkW.

```
$ sudo apt-get install libtool-bin
```

Now, you're ready to compile. First run the configure script to select what options you want compiled in:

```
$ ./configure --disable-gtk-update-checks --enable-x265 \
  --enable-fdk-aac --enable-libav-aac --launch-jobs=6
```

This line will configure HandBrake with the GTK interface and the x265 codec, and two implementations of AAC. Other codecs are already built-in. If you want only the CLI version, you can also add --disable-gtk to the options. Now, it's time to run make (and get a cup of coffee):

	Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz x8, Fujitsu-Siemens S792 laptop	ODROID-XU3 with interactive governor, maximum frequency on all cores	ODROID-XU3 with powersave governor, maximum frequency of 600MHz
Average FPS during first pass	36	7.5	1.5
Maximum temperature	N/A	94C	65C
Total encoding time for a 26s video	1m5.236s	4m15.786s	29m1.483s
Price	1600€	158€	158€

Table I - comparison of encoding benchmarks

```
$ cd build
$ make
```

When make is done (hopefully without errors), you have two options:

- sudo make install - will copy the relevant files to /usr/local/bin and install the software
- sudo checkinstall - will ask a few questions and then create and install a deb package for you. We'll go with this option because it's "the Debian way."

```
$ sudo checkinstall -D
--install=yes --pkgname=handbrake
\
  --provides=handbrake --requires
=libtheora,libogg,libvorbis,libsa
mplerate,libxml2,\
  libfribidi,libfreetype6,libfont
config1,libass,libmp3lame,libx264,
libjansson,\
  libglib2.0,libgtk-
3,libgudev-1.0,libwebkitgtk-
3.0,libnotify,libgstreamer1.0,\
  libgstreamer-plugins-
base1.0,libappindicator
--showinstall=yes --strip=yes
```

Now, the package handbrake should be installed in your system, providing /usr/local/bin/HandBrakeCLI for your scripting needs and /usr/local/bin/ghb for the GTK frontend. If everything is OK, you're free to delete the HandBrake directory and reclaim your disk space and you're done! If compiling from source is not your preference, you can try installing the already compiled package from http://bit.ly/1M5SDSm, which should work on Ubuntu 15.04. If you get stuck you can ask for help on the support thread at http://bit.ly/1MC4Atx.

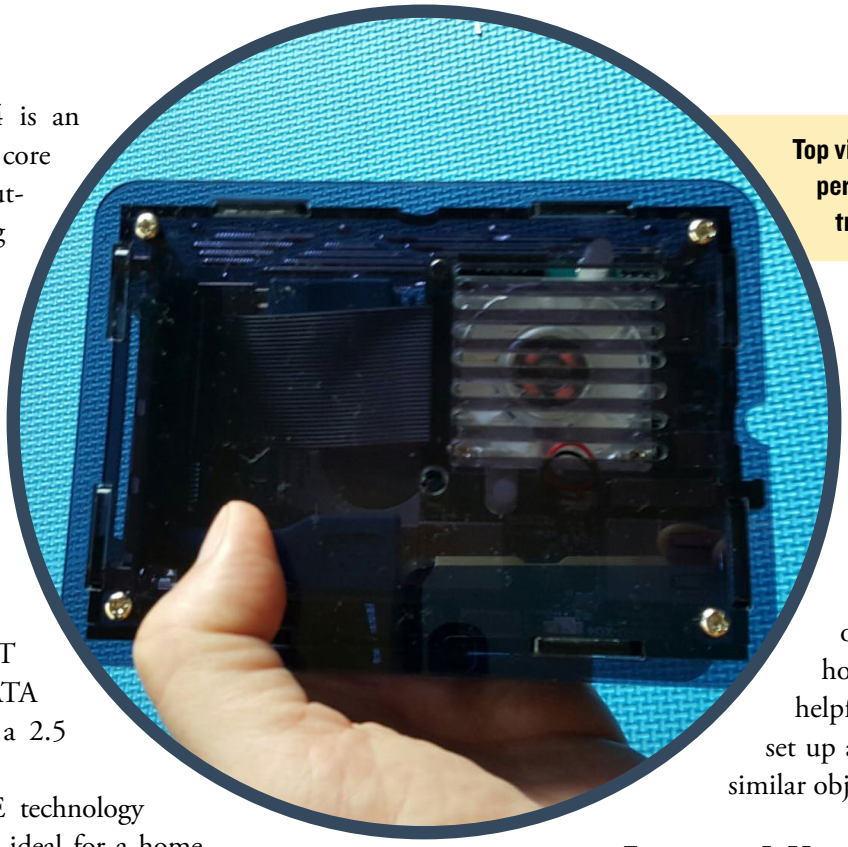
In terms of performance, HandBrake will generally use just the CPU for video encoding (it even uses NEON instructions if available). Compared with other systems, Table 1 shows what to expect when encoding a 1080p mp4 with x264, two passes, AAC sound, x264-preset=slow, and x264-profile=high. If you're doing frequent video encoding, you should ensure that your board doesn't operate for too long at high temperatures. Either improve cooling or reduce the maximum frequency so that you don't overheat it.

As expected, the ODROID doesn't match its bigger brother (Intel) in terms of performance, but it's 10 times cheaper while being just 4 times slower, so it's a good tradeoff if you are not in a hurry while transcoding. The way I'm using HandBrake with the ODROID is to convert high bitrate videos (20Mbps) taken with my phone to a more manageable bitrate (6Mbps) while keeping the same quality, all done with an automated cron job. Have fun transcoding!

CLOUDSHELL WITH AN ODROID-XU4 AS A HOME SERVER

AN ALL-IN-ONE PERSONAL CLOUD DEVICE

by Andy Yuen



Top view of the CloudShell, the perfect accessory for your trusty XU4

The ODROID-XU4 is an ARM-based Octa core single board computer based on the Samsung Exynos 5422, consisting of 2Ghz Cortex-A15 and 1.4 Ghz Cortex-A7 cores, and which comes with 2GB of memory. The XU4 replaces the original XU3 and XU3 Lite computing devices. CloudShell is a compact case for the ODROID-XU4 with a 2.2 inch TFT LCD and a USB 3 to SATA bridge for connection to a 2.5 inch SATA hard disk.

The ARM big.LITTLE technology used in the Exynos5422 is ideal for a home server due to the fact that when the cpu load is low, it uses the power efficient A7 core(s) and when the load is high, it switches to use the high performance A15 core(s). Depending on the load it may use 1 core, more than 1 core, or all cores at the same time. This also results in energy savings in the long run.

When people talk about home servers, they are usually referring to Kodi acting as a home theatre server. For this purpose I can use a smart TV or a set-top box. My home server requirements are quite specific, it has to allow access from the Internet to the following services running on it:

- **WordPress server for running my blog**
- **SSH server for remote access**
- **A rules engine demo application for users to interact with**
- **Run a NFS server**
- **Serve as the head node for my home computer cluster (not exposed to the Internet)**

The following sections describe how I set up my ODROID-XU4 as a home server, gives links to useful information regarding the setup procedure, and provides details of the finer points of installation and usage. I hope that this article will be helpful for anyone planning to set up a CloudShell and who has similar objectives to my own.

Assembling the CloudShell

Assembling the CloudShell is not difficult. However, if you are new to Linux, you will probably be wondering where to locate your hard disk after booting up the machine since the disk does not show up after you issue the df command. For the drive to appear, you will have to partition the drive, add an entry in the /etc/fstab file, and either issue a 'mountall' command or reboot the machine. You should test to see if you have any errors in your new /etc/fstab entries using 'mountall'. If there is an error in the entry that you are unaware of, the machine will not boot up after shutdown. If this occurs you will need to remove the SD card, insert it into your Linux machine, mount it, and correct the fstab file before you will be able to boot up your ODROID again. Here are some useful information nuggets, and links to help you set up your CloudShell.

The physical assembly is straight-forward, just follow the pictorial procedure here (http://bit.ly/1N3xNm7)

Information on how to set up the LCD and IR receiver can be found in the ODROID Wiki (http://bit.ly/1R6DOgZ) by searching on "CloudShell"

The Hard Disk setup procedure can be found here (http://bit.ly/1Gu1smv)

You may also want to set up a swap partition by following the instructions here (http://bit.ly/1Xnn48x)

Note that the UUID needed for /etc/fstab will not be generated until you issue the "mkswap" command.

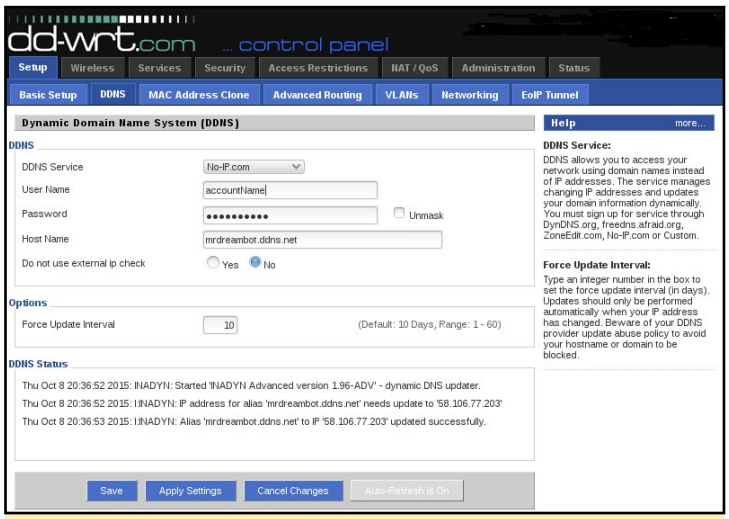
Networking and Dynamic DNS

Before you expose your home server to the Internet, you should make sure that the proper security is in place. Here is an article (http://bit.ly/1kxJuWw) from my blog on how I set up my home network security using a DMZ.

If your ISP is like mine, it can provide you with a dynamic IP address. For this case you will need to set up Dynamic DNS in order to access your server from the Internet using the same domain name despite the fact that your IP address changes regularly. To do this, you need to:

Sign up for a free Dynamic IP address account here (http://www.noip.com/free). Of course you can use your own preferred providers instead.

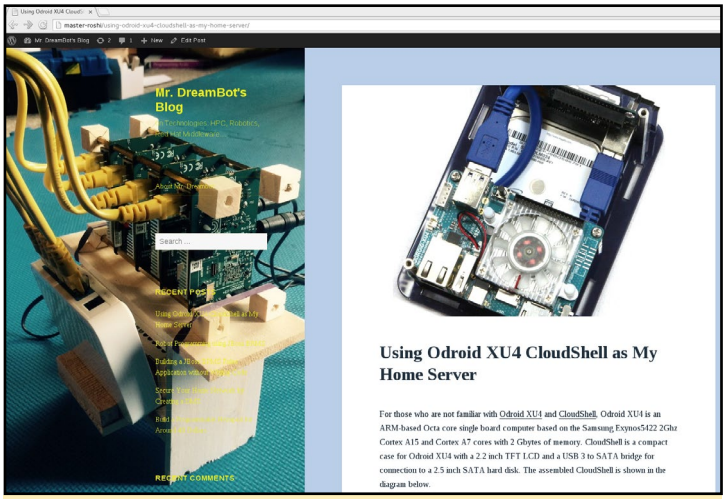
Configure DDNS on your Internet-facing router using your router's administration tool. You will need your account name, password, and host name (the domain name you have chosen for your home server registered with the provider when you signed up). If you are using DD-WRT on your router, the screen may look like this:



Router DDNS Setup

WordPress Server for Blogging

Setting up WordPress is not trivial. You have to set up the LAMP stack. LAMP is an an acronym of the names of its four open-source components: the Linux operating system, the Apache HTTP Server, the MySQL relational database management system (RDBMS), and the PHP programming language. This means that you will need to create a database in MySQL, configure the Apache Web Server and WordPress which includes setting the proper ownership of files and directories. There are times when I say to myself 'it may be easier if I just signed up with a free blog site'. But just have a little patience and you can reap the rewards of running your own blog on your home server. And as the saying goes, 'It's all about the journey, not the destination'.



My blog (http://MrDreamBot.ddns.net) powered by Cloudshell

Here are some hints to help you along:

Install the LAMP stack by following these instructions (http://do.co/1sLDg7T)

Install WordPress as described here (http://do.co/1Auy0sz). Installing WordPress is quite different from installing other packages. You cannot use the installed package that is in place after doing an 'apt-get install'. You will have to copy all that content to your Apache Web Server document root as described in the link.

If clicking on the "Add Media" button does not show any dialog box, add 'define('CONCATENATE_SCRIPTS', false);' to the bottom of your wp-config.php file.

Replace all symbolic links under the /yourPath/www/html directory and sub-directories with a copy of the file that it points to

If you get a message when accessing WordPress that states that you don't have access to a certain directory, it is most likely caused by /youPath/www/html/.htaccess. Remove or rename it and see if it helps.

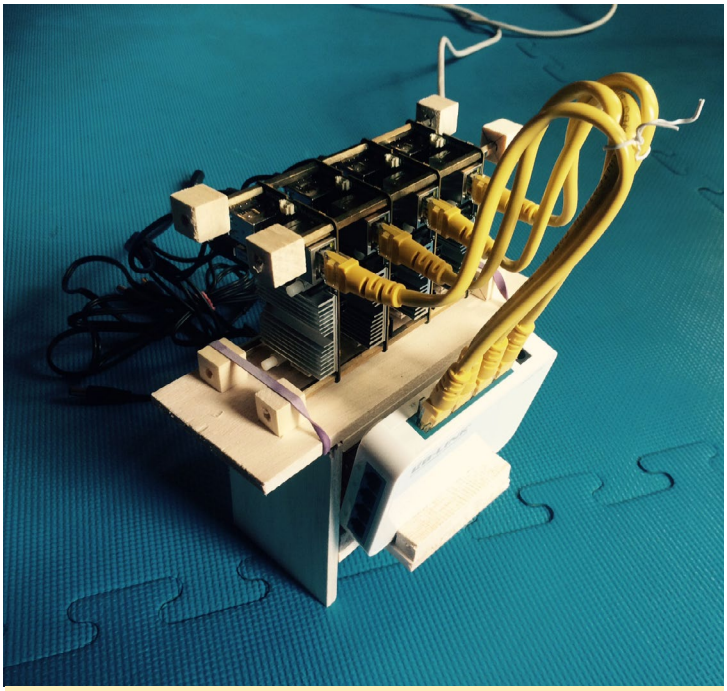
'+FollowSymLinks' and 'AllowOverride All' in /etc/apache2/sites-available/default.conf are important if you customise the WordPress permalinks that require URL rewriting. Make sure that you issue the commands 'sudo a2ensite default.conf' and 'sudo service apache2 restart' after making changes to the default.conf file, as shown in the example below:

```
ServerAdmin webmaster@localhost
             DocumentRoot /yourpath/www/html
             <Directory /yourPath/www/>
                          Options -Indexes +Fol-
lowSymLinks
                          AllowOverride All
                          Require all granted
             </Directory>
```

Configure your Internet-facing router to perform port forwarding of port 80 to your CloudShell IP address and port 80

NFS Server and Head Node for my Home Compute Cluster

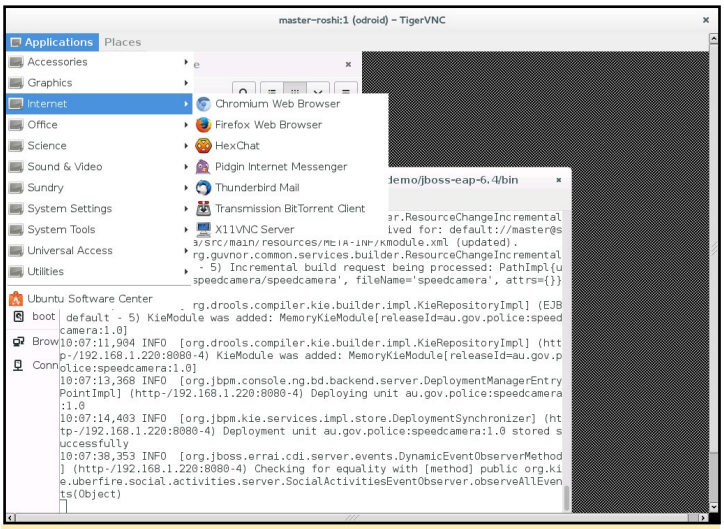
I built a Home Compute Cluster using 4 ODROID-U3 development boards as compute nodes for a total of 16 cores of computing power some time ago. It is now time for a revamp. I am adding the CloudShell as the head node of the cluster, and using the CloudShell as the NFS server, so that when I want to run a MPJ Express (http://mpj-express.org/) parallel program on the cluster I do not have to copy that program to each of the compute nodes. By using a NFS server, all I need to do is copy the program on the server, and the compute nodes can access it. No more copying. I own a large variety of Linux-capable single board computers including a Raspberry Pi, BeagleBone original, BeagleBone Black, CubieBoard V1, ODROID-XU3 Lite, and a ODROID-U3. Some of the OSes do not have the NFS module included in the kernel for use as a NFS server. I am pleasantly surprised that the ODROID-XU4 has no such problems.



My Home Compute Cluster using 4 X ODROID-U3

SSH Server for Remote Access

You can also set up a VPN (Virtual Private Network) server in order to access your home network safely from anywhere. For me, setting up a VPN is overkill, although I plan to play around with it for fun when I have time. All I really need is the ability to access my CloudShell from the Internet using SSH. Once I am able to access my CloudShell from the Internet, I can then use SSH port forwarding to create an encrypted tunnel for accessing my CloudShell's desktop using VNC.



VNC Desktop Connection to My CloudShell Via SSH

Here is how I set up my SSH server for Internet access:

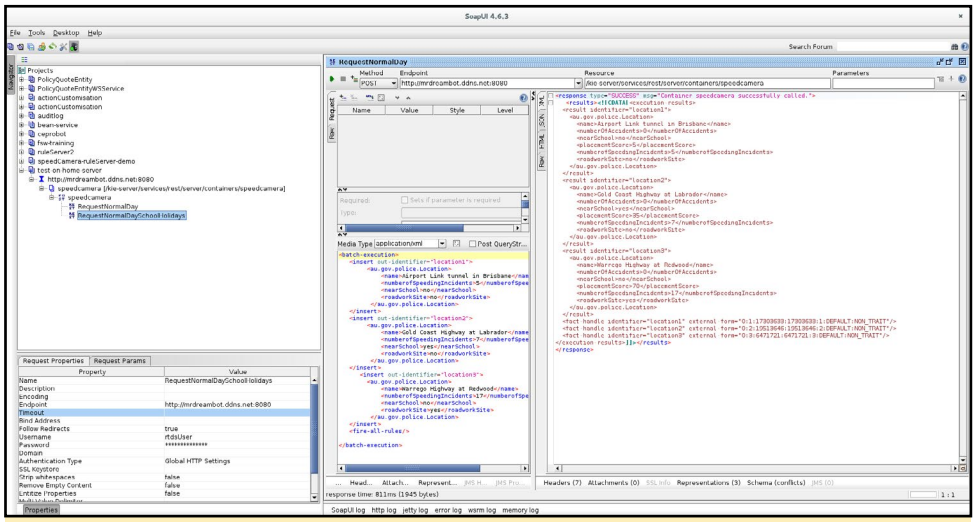
Set up the server as described here (http://do.co/1Gu1Mlj)

Configure your Internet-facing router to perform port forwarding from a port number of your choosing to your CloudShell IP address and port 22. When you SSH to the CloudShell from the Internet, you use the port number that you have chosen instead of the default port 22 to provide added Security through obscurity

Set up public key authentication (http://bit.ly/1LfGhF6) on your CloudShell and on the machines from which you will SSH from the Internet. Then force SSH login via public key authentication by following these instructions (http://bit.ly/1hYEgBg). With this configuration you will not be prompted for a password. The session will simply terminate if the connecting machine does not have the correct public key.

Business Rules Engine Demo

I am a middleware solution architect by profession. So it should not surprise you that I experiment and build small applications for demo purposes all the time. One application I built is a web service that uses business rules to determine the best location, among many, to place a speed camera. All of this is achieved without writing a single line of code. You can inter-



Using SoapUI to Interact with My Speed Camera Business Rules Demo Application

act with this web service using SoapUI, an open-source web service testing application, as described here (http://bit.ly/1NWYw4F).

This application requires the installation of a Business Rules Management System (BRMS) on the CloudShell. BRMS runs on a JEE (Java Enterprise Edition) application server, so installing Java is a prerequisite. Unlike the ODROID-XU3 Lite and ODROID-U3, Java is not installed on the Linux MicroSD card that I purchased together with the XU4. I download OpenJdk 1.8 using 'apt-get install'. I ran into some problems installing BPMS using its Java Installer. Most of the time the installer failed to install, complaining about missing Java classes, and occasionally it finished installing with errors and the installed application could not be started. After several tries, I decided to try the Oracle Jdk 1.8 for ARM instead. There was no problem with either the BRMS install, or with running BRMS using the Oracle Jdk. This came as a surprise to me because I used OpenJdk at work running BRMS on Fedora and other favours of Linux without any issues. It is rock-solid on the Intel platform. My ODROID-U3-based Compute Cluster has been running Java 1.8 without any issues. Interestingly enough, when I examined which version of Java has been installed on the U3, I found both OpenJdk 1.7 and Oracle Jdk 1.8. The latter is the default Java version meaning that I have been using Oracle Jdk 1.8 on my U3 machines all along!

To make the application accessible from the Internet, I have to do the following:

Configure my Internet-facing router to perform port forwarding from port 8080 to my CloudShell IP address and port 8080

When starting my BRMS on the JEE (Java Enterprise Edition) Application Server using the standalone.sh script, I have to add the option parameter '-b ipaddressOfMyCloudShell' (eg, ./standalone.sh -b 192.169.1.100) as by default standalone.sh binds only the localhost or loopback address.

Conclusion

The ODROID-XU4/CloudShell combination constitute a powerful and energy-efficient (compared to a regular PC) little device that appears to be tailor-made for use as a home server. With all the mentioned applications running on it including the enterprise grade BRMS middleware running on a JEE application server, I just can't help being amazed by how well it handles the load.

CUTE BUT CHALLENGING

DUNGEON BOSS IS A SUPERB FIT FOR THE CASUAL PLAYER

by Bruno Doiche

Sometimes all we ever want is a game that makes us feel the reward of a good turn-based adventure but time constraints work against us in finishing that 600+ hours RPG. If you tend to suffer from this problem, look no further! Dungeon Boss is a great dungeon game that has a great balance of being fun and cute while not being exceptionally time consuming, which nowadays is everything we ask for a game! As a bonus, the graphics are superb for such a small game. Enjoy!

https://play.google.com/store/apps/details?id=com.bigfishgames.dungeonbossf2pgoogle&hl=en

Begin your quest against some peasy skeletons.

And get ready to face a challenging but super fun to beat Boss.

RUNEAUDIO MUSIC PLAYER

BUILD YOUR OWN PROFESSIONAL QUALITY HOME AUDIO KIOSK

by Justin Lee

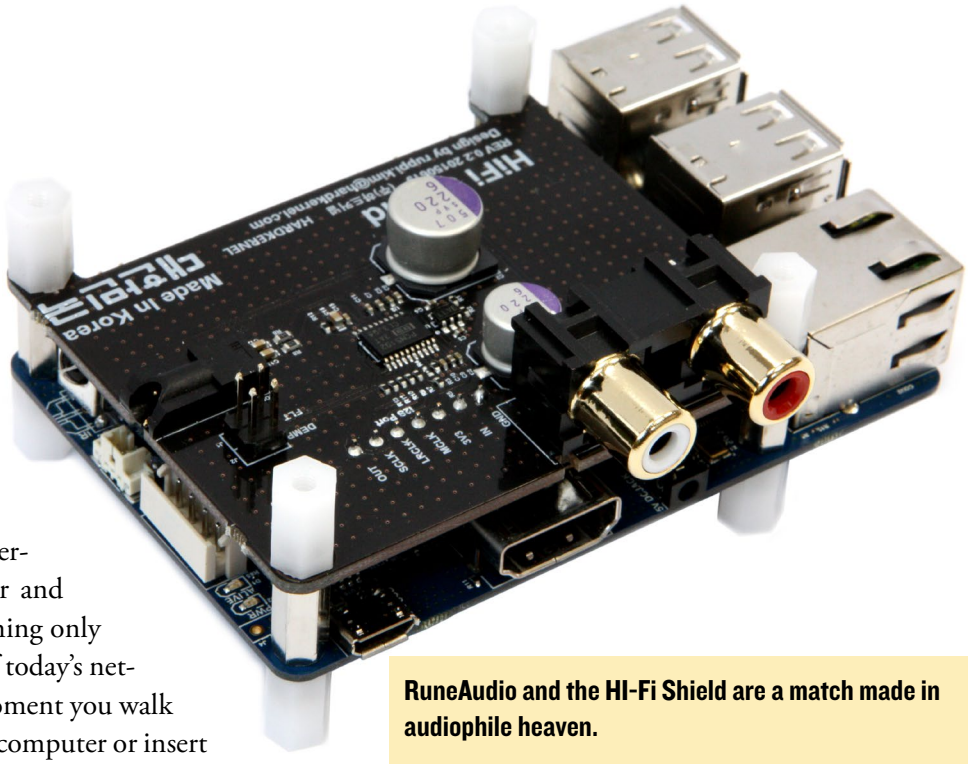
A network music player is a device that connects you to a virtually limitless assortment of music. What you can play varies from model to model, but most let you play music from:

Your NAS or USB storage library or other types of music files stored on your computer, smartphone, or tablet

Thousands of free Internet radio stations

A variety of online music subscription services that offer intriguing ways to discover and mix songs. Do you think that this is something only a geek could love? Think again! The best of today's network music players are ready to rock the moment you walk in the door. You don't have to turn on your computer or insert a CD. Simply fire up an app on your phone or tablet and tap the screen a few times, and there you have it: sweet-sounding music to soothe or inspire you. You can also add a nice color touch screen to display the artist and track information as well as organize your favorite music files.

This article will help you to build a network audio player, as well as to understand how to make a standalone embedded system with an ODROID. If you turn on the device, it will show you a beautiful UI and play the music automatically.



RuneAudio and the HI-Fi Shield are a match made in audiophile heaven.

The software setup has three major stages, with an optional fourth stage:

- **Install the RuneAudio OS image and set up the basic environment**
- **Enable the 7-inch HDMI LCD driver and install X11 and web browser**
- **Learn how to do auto-login, auto-start X-windows and applications like a professional consumer product**
- **Configure the IR remote controller settings**

Requirements

ODROID-C1+ (http://bit.ly/1Upx5yI)
HiFi-Shield (http://bit.ly/1M6UIXY)
7-inch HDMI LCD (http://bit.ly/1NWxgDx)
Speaker system with amplifier
Optional Infrared remote control (http://bit.ly/1M6UGiR)

Hardware

Plug the HiFi-Shield on your ODROID-C1+ and assemble the screws, then connect the 7-inch HDMI LCD to the C1+ board with a HDMI cable and a micro-USB cable.

Once you complete the full software setup, you will have a standalone music player system.

Follow this long process, and you can show a music web UI on an LCD screen using the Chromium web browser in the kiosk mode.

Install RuneAudio OS

Install the RuneAudio OS image for C1+ on your eMMC or SD. You can do this by downloading the OS image called RuneAudio_odroid-c1_0.3-dev_20150824_2GB.img from http://bit.ly/1GhrkCw. Use the serial console (with ODROID USB-UART kit), SSH to the IP address given to the board by your router, or use the HDMI screen with a keyboard. The username is "root", and the password is "rune".

Optimize startup

In order to speed up the boot process if using a direct Ethernet connection (no WiFi), enter the following command:

```
$ systemctl disable netctl-auto@wlan0
```

Maximize storage

The official OS image is made for cards around 2GB. In most cases your card has more storage capacity, so you can use it for local storage or other packages you like to install. Type the following command to enter the fdisk utility:

```
$ fdisk /dev/mmcblk0
```

- **Get the current size ("p + ENTER" on your keyboard)**
- **Delete the first partition ("d + ENTER")**
- **Add a new partition with default parameter ("n + ENTER + ENTER + ENTER + ENTER + ENTER")**
- **Write the partition table to your card, ignoring the error message ("w + ENTER")**
- **Restart system ("reboot + ENTER")**
- **Login again**
- **Resize the root file system:**

```
$ resize2fs /dev/mmcblk0p1
```

Check the new size:

```
$ df -h
```

Maximize RAM

Remove the reserved memory for the video play driver by editing the boot configuration file:

```
$ nano /boot/boot.ini
```

Scroll down until reaching this part:

```
# Disable VPU (Video decoding en-
gine, Saves RAM!!!)
# 0 = disabled
# 1 = enabled
setenv vpu "1"
```

Change the value to "0" from "1":

```
# Disable VPU (Video decoding en-
gine, Saves RAM!!!)
# 0 = disabled
# 1 = enabled
setenv vpu "0"
```

Save your changes with "Ctrl-x + y + ENTER" and reboot. The total accessible memory will be changed to 948MB from 836MB. We have gained over 100MB!

Enable I2S DAC driver

Enable the requirements for the ODROID HiFi-Shield by editing the boot configuration file again:

```
$ nano /boot/boot.ini
```

Scroll down until reaching this part:

```
# PCM5102 audio DAC Enable/Disable
# Uncomment the line below to __ENABLE__ Audio-
DAC(PCM5102)
#setenv enabledac "enabledac"
```

Remove the hash like this:

```
# PCM5102 audio DAC Enable/Disable
# Uncomment the line below to __ENABLE__ Audio-
DAC(PCM5102)
setenv enabledac "enabledac"
```

Save your changes with "Ctrl-x + y + ENTER" and reboot.

Set HDMI resolution

The default HDMI resolution is 720p (1280x720). If you have a problem with booting from UHC-1 SDCARD, try this approach. Find this section in the /boot/boot.ini and add a hash on the last line:

```
# UHS Card Configuration
# Uncomment the line below to __DISABLE__ UHS-1 Mi-
croSD support
# This might break boot for some brand models of
# setenv disableuhs "disableuhs"
```

Install desktop

Let's install a window-manager and web-browser with related

packages. First of all, you must synchronize the package database with the Arch-ARM package server:

```
$ pacman -Syy
```

Install video driver files (one for x11 and the other one for framebuffer):

```
$ pacman -S xf86-video-odroid-c1 xf86-video-fbdev
```

Install xorg server and xinit (startx):

```
$ pacman -S xorg-server xorg-xinit
```

Install the LXDE desktop GUI, making sure to select all optional packages:

```
$ pacman -S lxde
```

Install an extra font if desired. I installed Korean font because I want to display the song information in Korean.

```
$ pacman -S ttf-baekmuk
```

Next you need to Install the Chromium web browser. However the chromium package in the Arch-ARM server doesn't work. The solution is to download a working Chromium browser package from an alternative source and install it manually. I used the package available at http://bit.ly/1W4AFo5.

```
$ wget http://odroidxu.leeharris.me.uk/chromium-
45.0.2454.101-1-armv7h.pkg.tar.xz
$ sudo pacman -U chromium-45.0.2454.101-1-armv7h.pkg.
tar.xz
```

Next, run the LXDE window-manager:

```
$ lxdm
```

Test the RuneAudio UI on the HDMI screen. It should work with the following command:

```
$ chromium --user-data-dir --kiosk 127.0.0.1
```

Setup kiosk

Let's make the system run fully automatically, so that powering on will start everything. First, enable the auto-login on the HDMI console:

```
$ systemctl edit getty@tty1
```

Copy and paste below 3 lines and save it with "Ctrl-x + y + ENTER" and reboot:

```
[Service]
ExecStart=
ExecStart=-/usr/bin/agetty --autologin root --noclear
%I 38400 linux
```

You can bypass the login process on the HDMI screen, but you still need to login on the SSH and Serial consoles.

Auto-start LXDE

Run the following command to create a .bash_profile:

```
$ nano /root/.bash_profile
```

Copy and paste below lines and save it ("Ctrl-x + y + ENTER"):

```
#
# ~/.bash_profile
#
echo Run me first
if [[ -z $DISPLAY ]] && [[ $(tty) = /dev/tty1 ]];
then
exec lxdm
fi
```

Enable LXDE auto-login

Edit the file /etc/lxdm/lxdm.conf, edit the third line, save it, then reboot the system to verify that the system boots to the LXDE desktop:

```
autologin=root
```

Auto-start Chromium

Edit the file /etc/xdg/lxsession/LXDE/autostart, and add this line at the end:

```
@chromium --user-data-dir --kiosk 127.0.0.1
```

You can run the RuneAudio Web UI on the HDMI within 20 seconds after power-on when booting from eMMC module.

Remote control

You can optionally use the IR remote controller to control RuneAudio, which is available for purchase for USD$4 from the Hardkernel store at http://bit.ly/1M6UGiR.

The following functions are included:

- **volume up = "+"**
- **volume down = "-"**
- **next song = "arrow down" (so it matches to Queue)**
- **prev song = "arrow up"**
- **seek forward = "arrow right"**

The best way to interact with your songs while chilling out: A classic IR remote control

- **seek backward = "arrow left"**
- **toggle play/pause = "mute"**
- **power down = "power" (press and hole for 3 seconds) + "OK" (press and hold for I second)**
- **reboot = "power" (press and hold for 3 seconds) + "arrow up" (press and hold for I second)**
- **if player is stopped next or prev song (arrow down or up) starts the player**

First, edit the "/etc/lirc/lircd.conf.d/lircd.conf" file:

```
# this config file was automatically generated
# using lirc-0.9.0(default) on Mon May 25 19:47:48
2015
#
# contributed by
#
# brand: ODROID                        lircd.conf.conf
# model no.  of remote control:
# devices being controlled by this remote:
#


    begin remote

      name  lircd.conf
      bits           16
      flags SPACE_ENC|CONST_LENGTH
      eps            30
      aeps          100

      header       8964  4507
      one           544  1692
      zero          544   561
      ptrail        544
      repeat       8965  2285
      pre_data_bits   16
      pre_data       0x4DB2
      gap          107872
      toggle_bit_mask 0x0


          begin codes
              KEY_PREVIOUS             0x53AC
              KEY_NEXT                 0x4BB4
              KEY_FORWARD              0x837C
              KEY_BACK                 0x9966
              KEY_OK                   0x738C
              KEY_HOME                 0x41BE
              KEY_MUTE                 0x11EE
              KEY_MENU                 0xA35C
              KEY_VOLUMEDOWN           0x817E
              KEY_VOLUMEUP             0x01FE
              KEY_POWER                0x3BC4
          end codes


    end remote
```

Also, edit /etc/lirc/lircrc:

```
## irexec
    begin irexec
      begin
         prog = irexec
         button = KEY_POWER
         ignore_first_events = 20
         config = php /srv/http/command/ui_notify.
php Power "Power down = press OK\nReboot = press UP"
simplemessage
         mode = powerdown
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_PREVIOUS
         config = mpc prev && mpc play
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_NEXT
         config =  mpc next && mpc play
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_VOLUMEUP
         config = mpc volume +2
         repeat = 1
      end

      begin
         prog = irexec
         button = KEY_VOLUMEDOWN
         config = mpc volume -2
         repeat = 1
      end

      begin
         prog = irexec
         button = KEY_FORWARD
```

```
         config = mpc seek +00:00:02
         repeat = 1
      end

      begin
         prog = irexec
         button = KEY_BACK
         config = mpc seek -00:00:02
         repeat = 1
      end

      begin
         prog = irexec
         button = KEY_MUTE
         config = mpc toggle
         repeat = 0
      end
    end irexec

    begin powerdown
      begin
         prog = irexec
         button = KEY_OK
         ignore_first_events = 5
         config = /srv/http/command/rune_shutdown;
shutdown now -h
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_PREVIOUS
         ignore_first_events = 5
         config = /srv/http/command/rune_shutdown;
reboot
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_VOLUMEUP
         mode = irexec
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_VOLUMEDOWN
         mode = irexec
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_FORWARD
         mode = irexec
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_BACK
         mode = irexec
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_RIGHT
         mode = irexec
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_LEFT
         mode = irexec
         repeat = 0
      end

      begin
         prog = irexec
         button = KEY_MUTE
         mode = irexec
         repeat = 0
      end
    end powerdown
```

Execute the following commands to enable the infrared daemons:

```
$ systemctl enable lircd
$ systemctl start lircd
$ systemctl enable irexec
$ systemctl start irexec
```

To use the 7-inch touchscreen monitor, you need to select the 800x480 resolution and vout mode must be "dvi" instead of "hdmi". If the color of icon or cursor is strange, it can be fixed if you change the system color depth to 24bit.

```
# setenv m "vga"            # 640x480
# setenv m "480p"           # 720x480
# setenv m "576p"           # 720x576
setenv m "800x480p60hz"     # 800x480
# setenv m "800x600p60hz"   # 800x600
# setenv m "1024x600p60hz"  # 1024x600
# setenv m "1024x768p60hz"  # 1024x768
# setenv m "1360x768p60hz"  # 1360x768
# setenv m "1366x768p60hz"  # 1366x768
# setenv m "1440x900p60hz"  # 1440x900
# setenv m "1600x900p60hz"  # 1600x900
# setenv m "1680x1050p60hz" # 1680x1050
# setenv m "720p"           # 720p 1280x720
# setenv m "800p"           # 1280x800
# setenv m "sxga"           # 1280x1024
# setenv m "1080i50hz"      # 1080I@50Hz
# setenv m "1080p24hz"      # 1080P@24Hz
# setenv m "1080p50hz"      # 1080P@50Hz
# setenv m "1080p"          # 1080P@60Hz
# setenv m "1920x1200"      # 1920x1200

# HDMI DVI Mode Configuration
# setenv vout_mode "hdmi"
setenv vout_mode "dvi"

# HDMI BPP Mode
# setenv m_bpp "32"
setenv m_bpp "24"
# setenv m_bpp "16"
```

Don't forget to install the latest Kernel to enable the capacitive multi-touch driver. It might take a few hours to install and configure the software packages to make a standalone system. There is also a pre-built image available with all of the above steps included, which may be downloaded at http://bit.ly/1RXCLAs.

Credits

Thanks to Frank Friedmann (@hondagx35) who built and maintains the RuneAudio OS image for ODROID platform. He gave me a lot of ideas and improvements. You can find more helpful information at http://bit.ly/1MSStwB and http://bit.ly/1jSB2AL.

ODROID-VU7: 7-INCH HDMI MULTI-TOUCH SCREEN FOR THE ODROID-C1+

AN AFFORDABLE INTERACTIVE PORTABLE MONITOR

by Justin Lee

The 7-inch multi-touch screen for ODROID-C1+, known as the ODROID-VU7, gives users the ability to create all-in-one, integrated projects such as tablets, game consoles, infotainment systems and embedded systems. The 800x480 display connects via an HDMI link board and a micro-USB link board which handles power and signal. Just connect a DC plug in to the DC-jack on C1+, and you are ready to play, once you install the latest OS update. The price is only USD$55, and is available for purchase at http://bit.ly/1NWxgDx.

This 7-inch HDMI touchscreen is incredible as the perfect companion to your ODROID-C1+

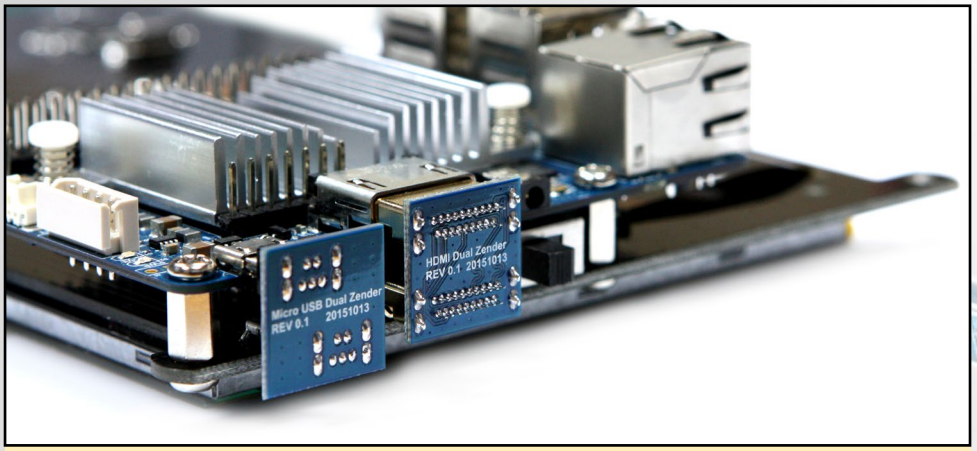
This high-quality touchscreen is specifically designed to work with both Android and Linux on the ODROID-C1+. You can easily attach the C1+ board onto the backside of the LCD screen in order to create a fully integrated system. It also works with the original C1 board if you use a micro-HDMI cable and a micro-USB cable. The 7-inch HDMI LCD comes with a HDMI link board, a micro-USB link board, three 8mm PCB supporters, and 6 screws.

Touchscreen drivers are available with support for 5 touch points, and proper drivers are integrated in the latest Ubuntu and Android OS images. Screen resolution must be configured to WVGA (800x400) and the DVI option should be enabled on the "vout" parameter by editing the boot.ini file, as shown in the Configuration section below.

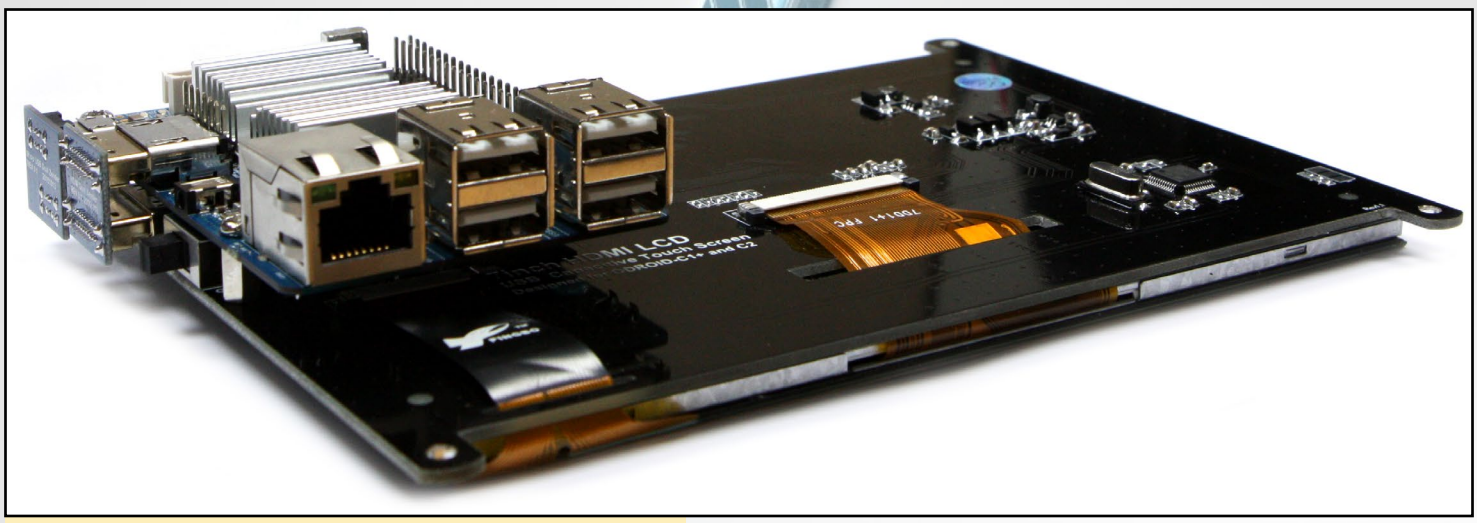
Assembly guide

Option I: Mount the C1+ board on the LCD unit with 3pcs of 8mm PCB supporters. The HDMI link board and micro-USB link board are used for interface and power supply.

Option 2: A Micro-to-micro USB cable and flexible flat HDMI cable can be used as well. The 35cm cable kit is sold separately.



The USB adapters that bridge the multi-touch screen to the CI+ in detail



The assembled direct mounted touch screen and CI board

A cable set is also available as an option to assemble your multi-touch screen

Specifications

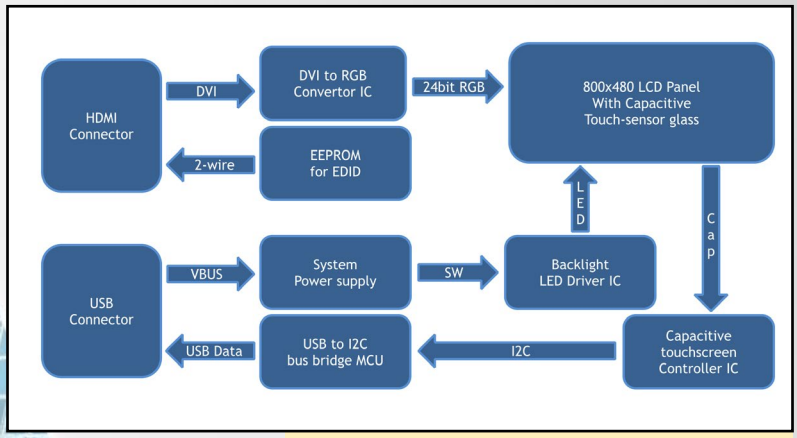
7-inch TFT-LCD
Screen Resolution: 800x480 pixels
5 finger capacitive touch input
Power consumption : 600mA /5Volt
Backlight on/off slide switch
Viewing angle (in degree): Left 70, Right 70, Up 50, Down 70
Screen Dimensions: 172.9 mm x 124.3 mm x 15 mm (including switch and connectors)
Viewable screen size: 153.6 mm x 86.64 mm (active area)

Configuration

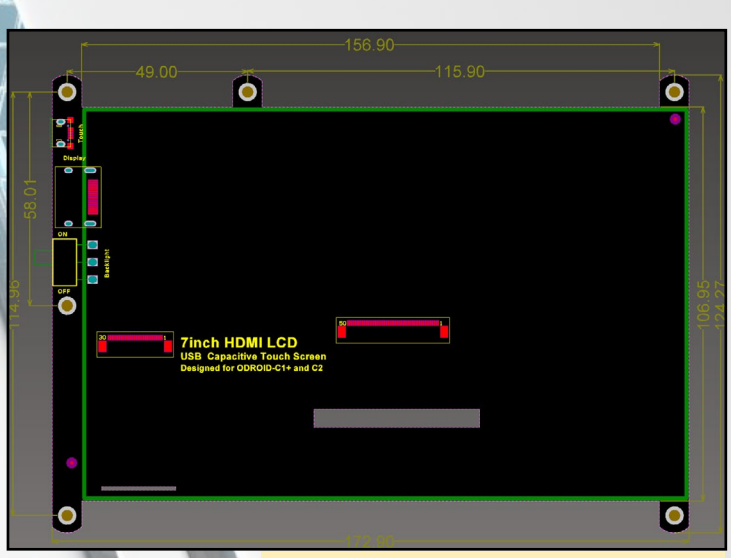
To use the touchscreen on Linux system, the Kernel version must be 3.10.80-128 or higher. Also, don't forget to configure the boot.ini file like this for 800x480 resolution and DVI output mode:

```
# Possible screen resolutions
# Uncomment only a single Line! The line with
setenv written.
# At least one mode must be selected.

# setenv m "vga"                  # 640x480
# setenv m "480p"                 # 720x480
# setenv m "576p"                 # 720x576
setenv m "800x480p60hz"           # 800x480
# setenv m "800x600p60hz"         # 800x600
# setenv m "1024x600p60hz"        # 1024x600
# setenv m "1024x768p60hz"        # 1024x768
# setenv m "1360x768p60hz"        # 1360x768
# setenv m "1440x900p60hz"        # 1440x900
# setenv m "1600x900p60hz"        # 1600x900
# setenv m "1680x1050p60hz"       # 1680x1050
# setenv m "720p"                 # 720p 1280x720
# setenv m "800p"                 # 1280x800
# setenv m "sxga"                 # 1280x1024
# setenv m "1080i50hz"            # 1080I@50Hz
# setenv m "1080p24hz"            # 1080P@24Hz
# setenv m "1080p50hz"            # 1080P@50Hz
# setenv m "1080p"                # 1080P@60Hz
# setenv m "1920x1200"            # 1920x1200

# HDMI DVI Mode Configuration
# setenv vout_mode "hdmi"
setenv vout_mode "dvi"
# setenv vout_mode "vga"
```

The Chromium browser supports the multi-touch pinch-zoom gesture, so you can enjoy the zoom in and out like a smartphone. The touchscreen also works with Android OS for the C1+. However, you need to use release version 2.0 or higher in order to use the touchscreen features.



Multi-touch screen block diagram



Rear side view and physical dimensions

Pinch and zoom demonstration

MEET AN ODROIDIAN

MANUEL ADAMUZ, ODROID MAGAZINE SPANISH EDITOR

edited by Rob Roy

Please tell us a little about yourself.

I am 33 years old and live in Seville, Spain, although I was born in Granada where most of my family lives. I am happily married with a 2 year old son, who is really bright and clever! A few years ago, I worked as a computer technician and programmer, but my current job is related to Quality management and IT: ISO 9001, ISO 27001, and ISO 20000. Computers are my passion, specially the single board computers such as the ODROID and Raspberry Pi.

How did you get started with computers?

While translating the ODROID Magazine into Spanish, I realized that many ODROIDians started with computers when they were kids. In my case, I got my first computer in high school when I was 17, thanks to my mother. It was a Pentium II, 233MHz with 64GB of RAM, a Hard Disk of 3GB, a graphic card of 4MB and a 64-bit Sound Blaster for the sound card. That was one of the first PCs with Windows 98. Everyone always remembers the first computer in their life! However, I didn't keep that computer, since my father eventually bought a more powerful computer and decided to trash it. I remember many very good moments from those days. I perfectly recall the day that the seller delivered the computer to my home. I hardly slept that night, and the only thing I could do was look through the window of the living room to see if there was a car arriving to deliver the PC. I also remember that it took me a week to crash Windows 98, after installing tons of games and applications. Windows did not start, and I had to check everything possible to get it working again.

What attracted you to the ODROID platform?

Thanks to my friend who lived in England at that time, I discovered the well-known Raspberry Pi, which I still own and use for different applications. I have 2 models: Raspberry Pi B and Raspberry Pi 2. After trying these boards, I realized that I needed something faster and more powerful, so I started looking for other alternatives on the Internet. I then bought a CubieBoard2 (1GHz CPU and 1GB RAM), mainly because of its price.

After a while, I confirmed that I had made a mistake by buying the CubieBoard, and now it is used as decoration in my office. Then, by chance, I found the ODROID boards, after reading an Internet blog about the comparison of several Single Board Computer, one of which was the ODROID-U3. I thought that with a 1.7 Ghz processor and 2GB of RAM, it sounded great! From that moment, it started my odyssey with ODROID boards, the ODROID forums and becoming the Spanish translator of ODROID Magazine. I have to admit that ODROIDs are perfect in many aspects, very powerful and versatile. I also want to mention that the ODROID Community is very impressive and, thanks to them, I managed to solve many of the problems during the installation and configuration of the different operating systems that I have installed, such as Ubuntu, Android, and OpenElec.

Manuel and his son David - we all hope he follows his father's footsteps as a cycling enthusiast!

How do you use your ODROIDs?

I have an ODROID-XU3 powered on permanently at home, which I use for surfing on the Internet, participating in social networks, and downloading software. It is very impressive how very little power consumption it requires. My Windows PC consumes 10 more times power than the ODROID board. I have another ODROID-U3 which I use as a Media Center running under OpenElec. My son is the one who mainly enjoys it by watching Disney movies. Recently, I re-

Participating in a semi-professional bike race

ceived an ODROID-C1 which I am currently evaluating, and so far it has provided very good performance. I plan to use it as a NAS system.

Which ODROID is your favorite?

That is a difficult question, since I like them all. Each ODROID board has its unique features. Most of the people like the ODROID-U3, and I really think it is a fantastic board. If I had to choose one, I would simply think about the application and then select the right board for that particular project.

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

I have read many opinions about this topic when I translate ODROID Magazine. I have seen during the years how Hardkernel has improved their board in many aspects: CPU, GPU, USB ports, eMMC, and GPIOs. But there is one aspect which has not improved: the RAM. Hardkernel has not manufactured any board over 2GB. Why? Is this not possible, or does Hardkernel think it is not necessary? I would like to see in the future ODROID boards with 3GB or 4GB of RAM. With more RAM, an ODROID board will replace the PC covering most of the applications.

What hobbies and interests do you have apart from computers?

I have several hobbies, but undoubtedly my other passion is the bicycle. As one friend says: "I like climbing hills with high gears...". From time to time, I participate in semi-professional races. Although I never win, at least I don't finish in last place. It is very difficult to win, as there are plenty of people obsessed

In 2012, Manuel visited Machu Picchu in Peru

with bicycles. I also enjoy travelling, although in the last years I haven't been able to travel much. I've been to Peru, Morocco, England, Scotland and Prague.

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

I worked as a programmer for a while, but did not really enjoy it. I think a good programmer must have first of all patience, as it is required to rewrite code and debug it many times in order to get the desired results. It is true that the Internet helps a lot, and there are plenty of online manuals and tutorials for almost everything. Also, if you find communities like the one behind the ODROID forums, I can guarantee you will save time and effort.

How did you become the translator of ODROID magazine?

When I discovered the ODROID boards, I also found that there was a magazine in electronic format about these boards. I took a look at the first articles and manuals published in 2014, and they were very interesting. In fact, I had to read the magazine on several occasions in order to solve some installation and configuration problems with the boards. I had some difficulties since the magazine was not available in Spanish. So, I took a risk and sent an email to Hardkernel proposing the translation into Spanish. I admit that it was difficult at the beginning, since some articles are very technical, but I really enjoy reading and translating the contributions of users who also enjoy these tiny computers.

Manuel's beloved ODROID-XU3 with external hard drive stays running 24 hours a day, 7 days a week