Retro gaming with Exagear • Docker swarm mode • Game Controllers

ODROID Magazine

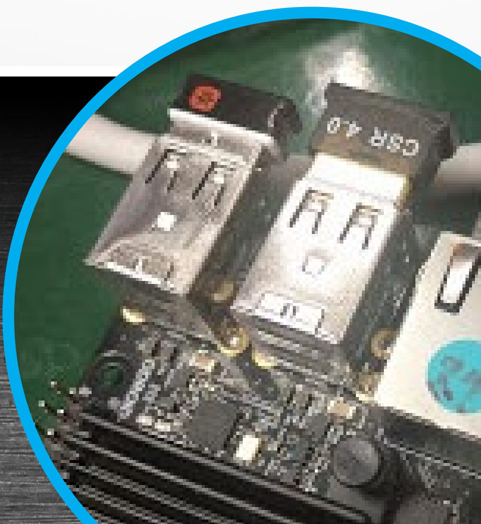
Year Three
Issue #36
Dec 2016

Water Cooled ODROID

Create a modern setup to get the absolute maximum performance from your computer

• Using the Hardware encoder functions of the ODROID-XU4

• How to Use an ODROID like a Bluetooth Speaker



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-C2 and ODROID-XU4 devices to EU countries! Come and visit our online store to shop!

Address: Max-Pollin-Straße 1
85104 Pförring Germany

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

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

Suche – Pollin Electronic

www.pollin.de/shop/suchergebnis.html?S_TEXT=odroid&log=internal

Elektronik, Technik und Sonderposten zu Superpreisen!

mein Konto | Kontakt & Service | Versand & Bezahlen | Katalog | Newsletter | Gutscheine

Suchen nach: Suchbegriff oder Bestellnummer Los

Merkliste | Neu registrieren | Anmelden

Direktbestellung

0 Artikel: 0,00 € Einkaufswagen

Sie sind hier: Startseite > Suche

Suchergebnis Ihre Suche nach "odroid" ergab 18 Treffer

sortieren nach: Relevanz (gewählt)

Seite: 1 von

1 - 18 von 18 Ergebnissen

Kategorie wählen

Filtereinstellungen

[-] alle Filter entfernen

Kategorie
[+] Bausätze / Module (17)
[+] Stromversorgung (1)

Preis 1 € 80 €

Ihre Vorteile
✓ Innovative Produkte
✓ Optimales Preis-/Leistungsverhältnis
✓ Kompetente Beratung
✓ Schnelle Lieferung

Noch Fragen?
Wir sind gerne auch persönlich für Sie da!
+49 (0) 8403 / 920-920

Sicher Einkaufen

ODROID-U3 BACKUP BATTERY
RTC Lithium-Backupbatterie für den ODROID XU, XU+E, XU Lite und U3. Wird direkt auf den RTC-Port des ODROID gesteckt.
2,95 €*
Best.Nr. 810 319

ODROID-U3 USB-UART MODULE, Schnittstellenkarte
USB/UART System-Interface (IC: CP für die ODROID Einplatinencomput X2, U2, U3, XU, XU+E, XU Lite). I Entwicklung und Debugging.
15,95 €*
Best.Nr. 810 326

ODROID-U3 Einplatinen-Computer, Cortex-A9 QuadCore, 2 GB
Kleiner als eine Scheckkarte aber riesig in der Leistung! Der ODROID-U3 verfügt über eine leistungsstarke SAMSUNG QuadCore-CPU, 2 GB Arbeitsspeicher und
69,95 €*
Best.Nr. 810 264

ODROID-U3 CASE Kunstst
Passgenaues Gehäuse mit Lüftungsschlitzen für den Maße (LxBxH): 87x52x29
4,95 €*
Best.Nr. 810 317

ODROID-U3 ODROID-UPS, USV, LiIon, 3 Ah
Unterbrechungsfreie Stromversorgung mit zwei LiIon-Akkus. Das Board besitzt die gleichen Außenmaße wie der ODROID-U3 Einplatinencomputer und kann direkt per
31,95 €*
Best.Nr. 810 328

ODROID-U3 eMMC Android
Superschnelles N den ODROID-U3 Betriebssystem aufgespielt un
39,95 €*
Best.Nr. 810

ODROID-U3 eMMC Modul, 64 GB, mit Linux
Superschnelles NAND-Speichermodul für den ODROID-U3 Einplatinencomputer. Das Linux-Betriebssystem XUBUNTU ist fertig aufgespielt und kann direkt

ODROID-Linux
Supersch den OD Linux- berei
39

Hardkernel attended TechCon 2016 in late October, and showed off some of the capabilities of the new **ODROID-C2**. There were several demos of **DIY** projects set up, including an **Ambilight** display, a **CloudShell 2** server running the latest mainline kernel, and a touchscreen media player using a **HiFi Shield**. The liquid cooling setup featured on the cover this month is another example of an amazing **DIY** project that creates the ultimate system for maximum overclocked performance. It's a work of functional art that demonstrates the technical expertise of the **ODROID** community.

Several other projects that community members have created include using an **ODROID** as a **Bluetooth** speaker, building a real-time kernel for use in a single-threaded operating system, and designing a combination seedbox and Networked Access Storage unit from an **ODROID-XU4** using the **CloudShell**. Andy concludes his Docker tutorial with an overview of swarm mode, our resident mad scientist Bo details his latest discoveries, Tobias helps us select the best gaming controller with an in-depth review, and Bruno brings us the latest in Android fun with his favorite games of the month.

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/1ypImXs.
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

ameriDroid.com
High-Performance Embedded Computers

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

ODROID-XU4

ODROID-C1+

ODROID-C0

OWEN ROBOT KIT

ODROID-C2

VU7 TABLET KIT

Hardkernel's EXCLUSIVE North American Distributor

OUR AMAZING ODROIDIAN STAFF:

Rob Roy, Chief Editor

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

Bruno Doiche, Senior Art Editor

Some wonder why Bruno talks to himself so much. He answered: "Of course I talk to myself. Sometimes I need expert advice, man!" Yea, he has done a lot of crazy things on the magazine over the years. After all, acting normal has never been one of his greatest strengths.

Manuel Adamuz, Spanish Editor

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

Nicole Scott, Art Editor

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

James LeFevour, Art Editor

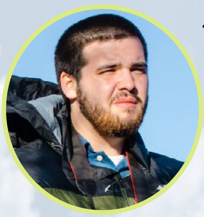
I'm a Digital Media Specialist who is also enjoying freelance work in social network marketing and website administration. The more I learn about ODROID capabilities, the more excited I am to try new things I'm learning about. Being a transplant to San Diego from the Midwest, I am still quite enamored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

Andrew Ruggeri, Assistant Editor

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

Venkat Bommakanti, Assistant Editor

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



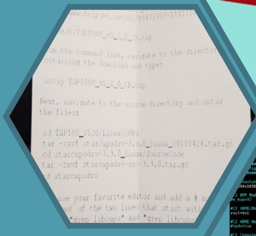
Josh Sherman, Assistant Editor

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

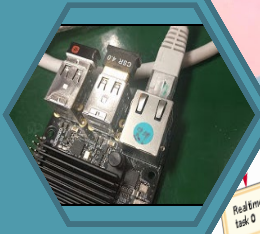
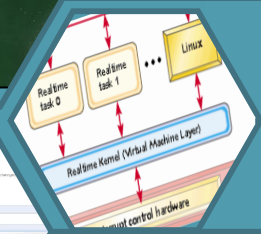
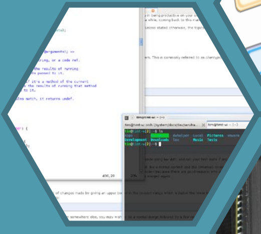
INDEX

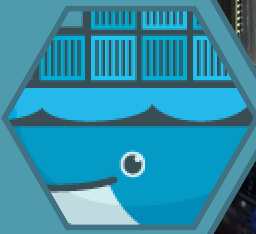
ARM
Tech

Eltechs
ExaGear
Desktop



Equalizer
Tone/Vol
Preset
Save



Realtime task 0
Realtime task 1
Linux
Realtime Kernel (Virtual Machine Layer)
Interrupt control hardware




HARDKERNEL AT ARM TECHCON 2016

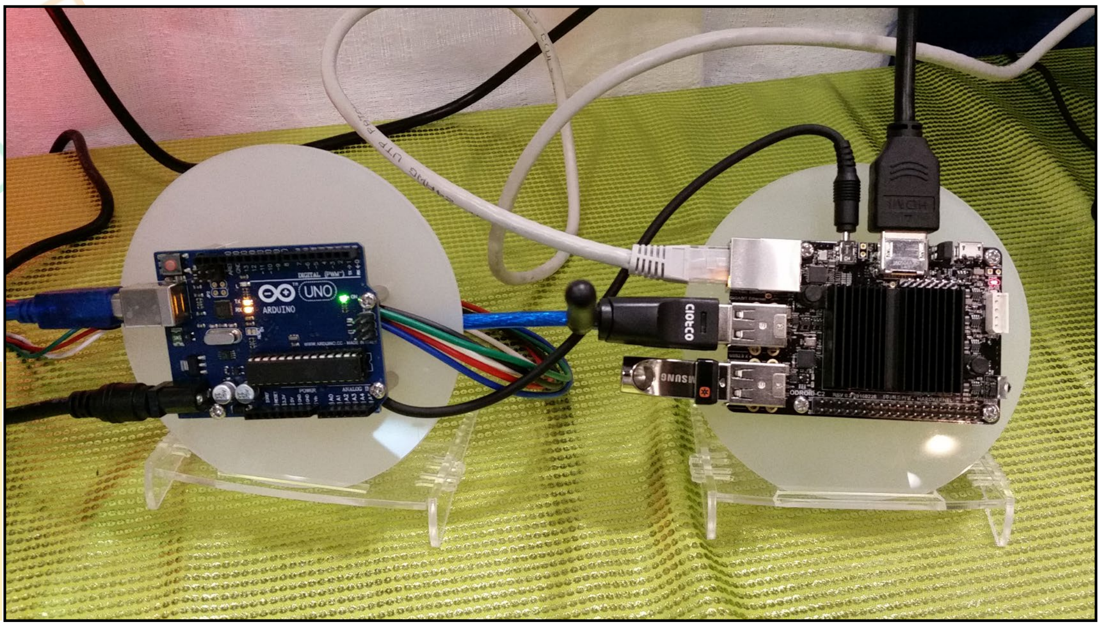
SHOWCASING THE ODROID-C2

by Rob Roy (@robroy)

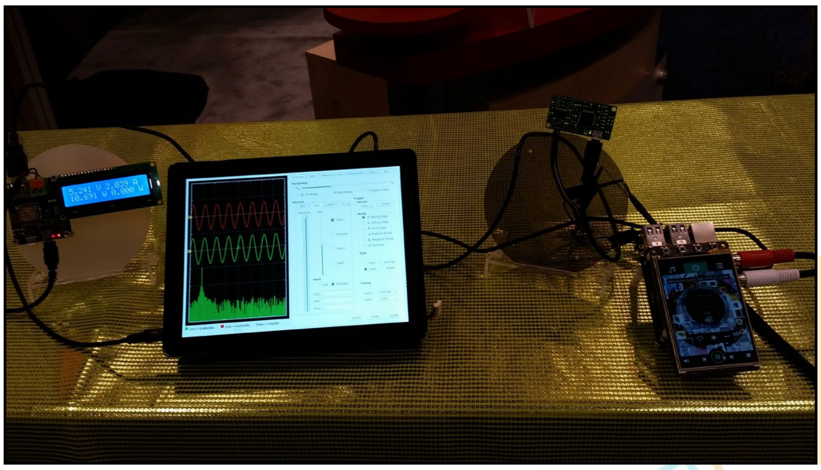
This year at ARM TechCon in Santa Clara, California, the engineers at Hardkernel displayed several demonstration projects using the ODROID-C2, including a HiFi Shield setup with a portable Volumio player, and a gorgeous 55-inch 4K Ambilight display. Many ODROIDians stopped by to admire some of the next generation products, such as the new ODROID-VU8 8-inch touchscreen with case, and the new CloudShell 2, which was set up to run a RAID array using two SATA hard drives. Check out the pictures to get a peek at the products Hardkernel is offering soon!

Hardkernel
AArch64, HDMI-2.0, 4K-HEVC, Gbit-Ethernet, 2GB DDR3 and much more on a tiny SBC

The Hardkernel booth was very eye-catching with an Ambilight display

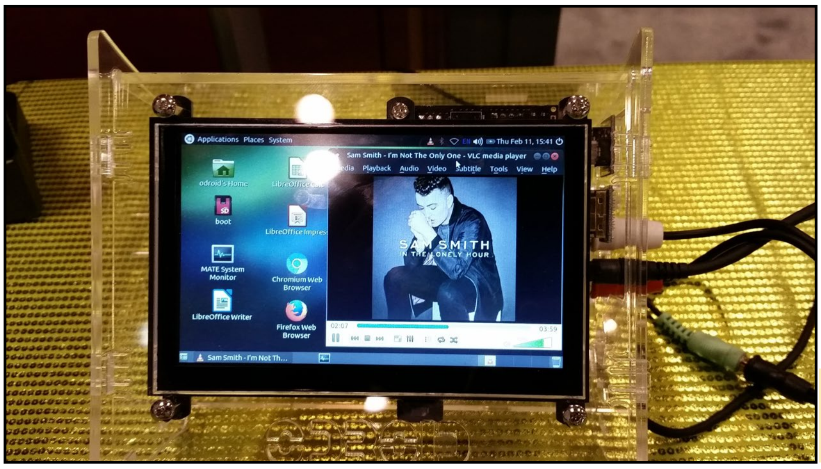


The Ambilight system ran on an Arduino which analyzed the 4K video in realtime while the ODROID-C2 used Kodi to display the video simultaneously on the 55-inch 4K monitor

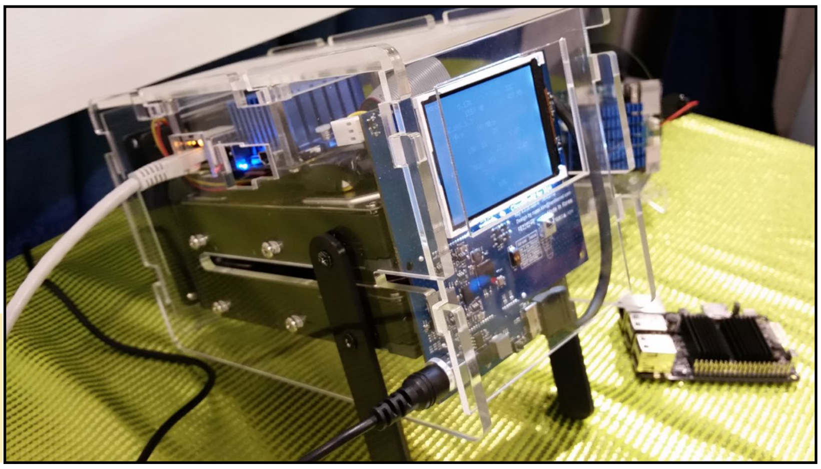


The ODROID-C2 on the right is playing music through the HiFi Shield, and the ODROID-C2 on the left is running oscilloscope software on the new ODROID-VU8 8-inch touchscreen with case

High quality music played through a HiFi Shield during the conference



The rarest of all devices: An Ubuntu 16.04 touchscreen tablet



The new CloudShell 2, housing an ODROID-XU4 running Linux kernel version 4.7.8 and a SATA RAID array

RETRO GAMING WITH EXAGEAR

by Gaukhar Kambarbaeva

It's no secret that retro gaming is now back in style. After buying a new gadget, any true gamer instantly wants to download and play their favorite titles on it. Luckily for ODROID owners, there is a way to turn their device into a retro-gaming machine with Exagear Desktop to play your favorite Linux and Windows PC games on your ODROID. In this article, I will detail how to run some iconic PC games on the ODROID platform: Arcanum, Heroes of Might and Magic 3 and Sid Meier's Alpha Centauri.

Installation

Before starting, install Exagear Desktop from http://bit.ly/2cul90r. Put the ExaGear Desktop archive, installation packages, and ExaGear Desktop license key in the same folder. Open your MATE (command line) Terminal, move to this folder, and unpack the archive:

```
$ tar -xvzpf exagear-desktop-
odrxu4.tar.gz
```

The next step is to install and activate ExaGear on our ODROID. You can do this with Exagear's installation script in the folder, which will automatically detect the packages and license key:

```
$ sudo ./install-exagear.sh
```

Installation is relatively easy and should essentially complete itself.

Getting started

ExaGear Desktop essentially runs a virtualized version of Ubuntu that uses the x86 architecture, instead of the ARM architecture that our ODROIDs utilize. Let's start the environment from our AMTE Terminal and take a look around:

```
$ exagear
```

You can confirm that you're in an x86 environment by running the "arch" command:

```
$ arch
I686
```

We recommend that you update your apt-get repositories during the first launch of the guest system:

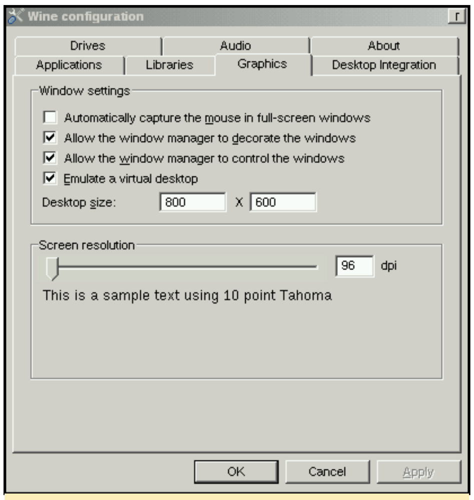
```
$ sudo apt-get update
```

Since we are going to launch Windows games in this environment, we'll also need to install Wine. Wine exists for Ubuntu natively on our ODROIDs, but no ARM Windows games exist yet. We need to install x86 Wine inside the Exagear virtual environment. This can be done easily using the apt-get command:

```
$ sudo apt-get install wine
```

After installing Wine, we recommend running winecfg and enabling the "Emulate a virtual desktop" bar. Otherwise, you can face issues when applications try to switch to full-screen mode:

```
$ winecfg
```



Window settings
- [ ] Automatically capture the mouse in full-screen windows
- [x] Allow the window manager to decorate the windows
- [x] Allow the window manager to control the windows
- [x] Emulate a virtual desktop

Desktop size: 800 X 600

Screen resolution 96 dpi

This is a sample text using 10 point Tahoma

Figure 1 - The Wine configuration screen

At this point, you're prepared and ready to try installing some PC games. It's possible to run all kinds of distributionss and installers, but the process is quite tricky and can take some time to figure out how to make each game work, especially for older games with compatibility issues. We recommend using a

DRM-free version of a game from a service like Good Old Games (GOG), which offers a consistent way to download and install games that the ExaGear Desktop environment usually supports.

Arcanum

This awesome early 2000s adventure game can run on ExaGear Desktop. First, purchase and download Arcanm (http://bit.ly/2fjmVa0). Then, run the installer with Wine in the MATE Terminal, and the graphical interface of the game installer will appear:

```
$ exagear
$ wine setup_arcanum_2.0.0.15.exe
```

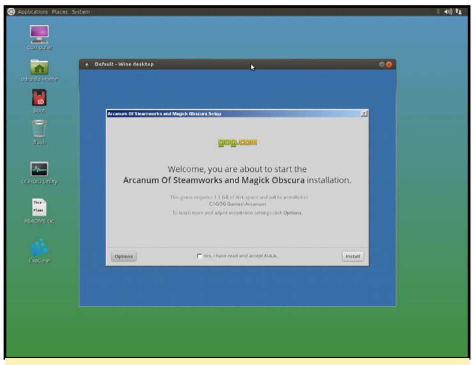


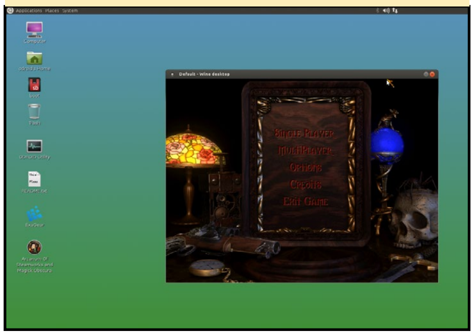
Figure 2 - Arcanum

After the installation is finished, you can run Arcanum from your desktop. If ExaGear installs any icons on your main ODROID Ubuntu desktop, they should automatically start the virtual environment and load the game. Select your favorite hero and start playing Arcanum on an ODROID!

Heroes of Might and Magic 3

Like Arcanum, you can get HoMM3

Figure 3 - The Arcanum menu screen



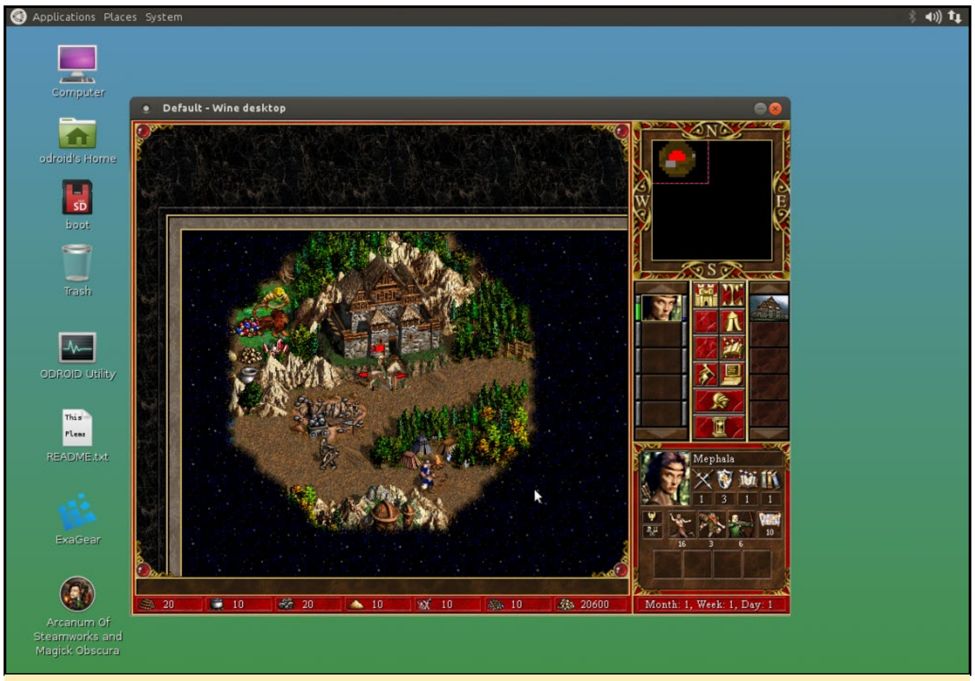


Figure 4 - Heroes of Might and Magic 3 in action

from GOG at http://bit.ly/2gftTtl. This turn-based strategy game also loads normally with the installer and Wine in a MATE Terminal:

```
$ exagear
$ wine setup_homm3_com-
plete_2.0.0.16.exe
```

After loading, the game installer will show up and you can proceed with installation. Once the installation is finished, you can run HoMM3 directly from the desktop. Wait until the game is loaded and then you will see the Start menu, from which you can select the map and play Heroes of Might and Magic 3 on your ODROID.

Sid Meier's Alpha Centauri

This awesome strategy game from the maker of Civilization is also available on GOG at http://bit.ly/2fyfVBf. Since it uses the GOG installer, you can download and install it on the guest x86 system with Wine just like the other games:

```
$ exagear
$ wine setup_sid_meiers_alpha_
centauri_2.0.2.23.exe
```

After installation, is finished you can run Alpha Centauri directly from the desktop or from the Start menu. Select a faction and start your space colonization adventure.

Figure 5 - Alpha Centauri screenshot

Exagear Desktop (http://bit.ly/2cul90r) allows you to run a lot of great 1990s and early 2000s PC games on your ODROID. You can run other x86 applications too. For comments, questions and suggestions, please visit the original post at http://bit.ly/2fH6Fwf.

LINUX GAMING

CHOOSING THE RIGHT GAME CONTROLLER FOR YOUR PLAYSTYLE

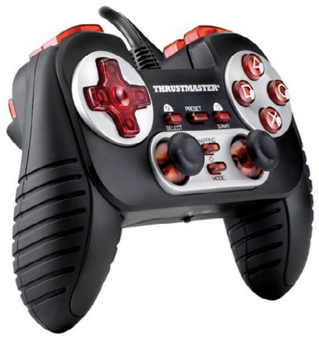
by Tobias Schaaf (@meveric)

I normally discuss a lot about different games and emulators that work on the ODROIDs, but in this article, I want to talk about one of the most important assets used by gamers: the gamepad. There are plenty of gamepads out there, and I want to talk about a few that I own myself, indicating what I use them with and which I prefer. I will try to explain how to setup your own controller based on my ODROID GameStation Turbo image, which is also applicable to my Debian Jessie images. I will also point out some special cases.

Trustmaster Dual Trigger 3 in 1 Rumble Force

I have a few different controllers that I've collected over the years, some of which I wanted / needed to create compatible images. The first one I got from a German shop, almost 15 years ago, while I still lived with my parents. Up to that point, I was only using joysticks, and this was my first gamepad.

Figure 1 - Trustmaster Dual Trigger 3 in 1 Rumble Force Gamepad, which was my first Gamepad



I got this controller (http://bit.ly/2fo53Wy) for my PC back then, knowing well that as a PC player, one rarely uses any controllers. While most games run with a mouse and keyboard, some work with joysticks, especially space simulations and fighter simulations. However, responsiveness was often inadequate. Back then, I blamed it on Windows, because every time I calibrated the joystick it was working fine for a little while, but later it was always off again. Today, I know that the left analog stick did not work correctly and always points to the upper left direction which makes it nearly impossible to use this gamepad.

Apart from that, it is actually a very nice controller. It has a nice touch to it, using some kind of a rubbery finish. It gives the controller a very good grip and prevents it from slipping through your fingers. It is slightly heavier than other controllers that I have, but not unpleasant. Besides that, it comes with 12 buttons, a D-Pad, two analog sticks and two triggers on the back. It has "rumble" support and actually has two connectors one for PS2 and one USB connector for PS3 and PC. It also has 4 shoulder buttons which is quite nice and allows for some nice mapping. The gamepad is programmable, but I never took the time to program it.

Pros

- Very good touch and feeling, rests very good in the hand
- Very good and sturdy buttons, if you shake the gamepad nothing makes a sound, the buttons feel like they are high quality
- Four shoulder buttons + 2 trigger buttons (extra buttons for special functions)
- Playstation symbols and support (good for PSP or PS1 emulation)

Cons

- Very old model, probably hard to find
- Not all emulators detect or use the rumble support
- Some issues with Retroarch
- No support for most emulators

Hama Black Force

The Hama gamepad "Black Force" is a gamepad that I got much later. It is an inexpensive model, but is still a rather good gamepad. It is a rather cheap copy of a PS3 controller, but works nonetheless. None of the buttons has a description other than numbers. There is an "analog" button where the home button would be on the PS3 controller. If you press the button, a red LED shines below

Figure 2 - Hama Black Force Gamepad, courtesy of game-debate.com

the button indicating that analog mode is activated. This is actually needed to send analog signal for the D-Pad, or else most emulator and programs won't recognize the D-Pad at all. The controller has 4 shoulder buttons, but is missing any triggers.

This gamepad is only registered as a "Generic USB Joystick", but it still works out of the box with Retroarch and other programs and emulators. The controller is not that high of a build quality, and if you shake it, you can hear the buttons rattling. Most of the noise appears to come from the shoulder buttons, but also some from the D-Pad. The analog sticks have a rough surface which gives extra grip, which is not bad, while not as comfortable as the rubbery finish of the real PS3 controller. They have made a new model having the Playstation X, O, Square, Triangle button markings as well.

Pros

- Inexpensive - this gamepad is really intended for those with a low budget, and clones of it are normally between 5-15 €uros
- It may not look or sound like a good controller, but many reviews of this controller are quite positive. It is generally "good" in all categories, but neither awesome nor great
- Works out of the box with many games, programs and emulators
- Easy to set up and configure for most emulators and games

Cons

- Not a "name-brand" that is in any way recognized by any program. Therefore other "no-name" controllers will be found under the same name, and they might have different button layouts which makes it nearly impossible to have different configuration files for these controllers
- The 1.9m cable length could be too short for some players

XBox 360 Wireless Controller

When I started my work on ODROID GameStation Turbo, I knew I needed to get my hands on one of the "main-line" controllers out there. It turned out the XBox 360 Controllers were the most widely supported, so it made sense to get one. I also wanted to have a wireless controller to get rid of the cable. Since the system that I was building was meant for kids, I did not want them to trip over any cables.

The controller is probably the heaviest of all controllers that I have, but it feels good in the hands, and so I do not think that its weight is much of an issue. It is the controller that my OGST image is primary configured for, and all games should work out of the box with this one. There are some minor issues with the drivers: your controller always blinks, or it may not be recognized again after it turned off and you turn it back on while you are within a game or a program such as Kodi. Besides that, it is the gamepad that is probably best supported under Linux. It does not mean that it gets a lot of patches and bugfixes, but it means that most games/programs are developed with XBox 360 controller support.

Figure 3 - The XBox 360 wireless controller is probably the best supported controller under Linux

Some games lack all but XBox 360 controller support. The wireless variant needs a proprietary PC remote adapter since it does not use a standard like Bluetooth, but that is actually not a bad thing, since the adapter supports up to 4 controllers simultaneously. This is quite nice, because it means that with just one USB port in use, you can use 4 separate gamepads. It also comes with either standard AA batteries, or you can buy a rechargeable battery pack and use a charging station to keep your controllers ready for play. This is quite nice, since you always have a working controller as long as you have batteries in the house, or have another charged battery pack.

What I do not like about this controller is the so called "deadzone" and the inability of the controller to go back to "0" (zero) when you release the controls. The analog sticks go from -32768 to +32768 either from left to right or top to bottom. When you release them they should return to 0. The range that it is straying away from 0 is what one refers to a "deadzone". It signifies that any value in this range should be ignored and considered 0 in order to prevent unwanted movements. On good controllers these dead zones can be very small. For example, a deadzone of say 4-5% of the maximal possible value is very common, which would be 1300-1700. Even if you go a bit further and say every value smaller than 2000 should be ignored, that should be fine for most controllers.

With the XBox 360 controllers, when I released the analog stick for it to go back to 0, I could see values as high as +/- 7500 or higher . That is a deadzone of over 20%. The two previous controllers have values between 0 and 255, so the center should be 128. They normally do not stray more than 124 or 132 which is about 3-4% and mostly they settle even closer than that. The PS3 controller has values between -127 and +127 and will always return to 0 when released. Not even once did I see it not return to 0.

Pros

- Best supported controller by games, emulators and programs
- Up to 4 controllers simultaneously over one USB port
- Easy to switch batteries / battery packs, so it is always ready to play
- Not too expensive and easy to buy, with lots of different models and designs to choose from

Cons

- Not very precise
- Rather heavy compared to other controllers

Playstation 3 DualShock 3 Controller

The Playstation 3 controller, often called the "Sixaxis", uses standard Bluetooth communications. It makes a preferred controller for mobile devices such as tablets and smartphones, although they often need to be rooted. They can be used both with a USB cable as well as over a Bluetooth adapter on the ODROID. This makes them very flexible, though you can only connect one PS3 controller per Bluetooth dongle.

Figure 4 - Playstation 3 DualShock 3 "Sixaxis" controller

Figure 5 - Leaning into the turn of your favorite racing game

This controller is no doubt one of the best controllers out there. It feels good in the hand, is very sturdy, relatively easy to use, has good supported, and is a lot of fun to play with. If you happen to connect the PS3 to a Linux or Windows PC that can read all the inputs from the gamepad, you can see how feature rich this controller is. Everything on this is actually an axis or ramp. Pressing a button is not just 0 and 1, but it actually registers how strong you press the button.

The PS3 controller has a motion sensor or gyroscope that registers how you turn the controller. When it comes to functions, the PS3 controller is probably the best I have. It is also the most expensive one I have. A new one can cost 50-90 €uros.

Pros

- Very good controls, very sturdy, good feeling in the hand
- Very precise control
- Lots of functions
- Wireless and wired options in one controller

Cons

- Very expensive
- Has to be loaded over USB (no exchangeable batteries)

Sabrent USB 2.0 Twelve Button Game Controller for PC

This is the latest gamepad that I bought. The reason for this is that I wanted a gamepad with 6 buttons on

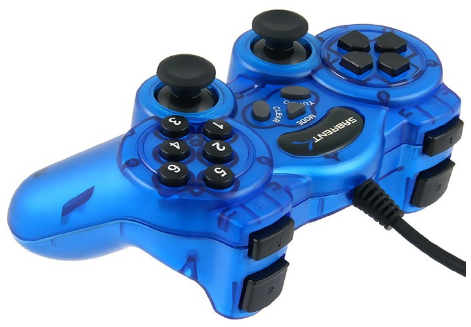


Figure 6 - Sabrent 12 Button gamepad cheap and gets the job done

the side for emulators like yabause to play Sega Saturn games. They required 6 buttons instead of 4 like most controllers have nowadays. It should also work fine with some MAME games that require 6 buttons instead of 4.

This controller registers as the same generic USB joypad as the Hama controller does, which brings me back to my point that having one configuration for different controllers will not work with these kind of controllers. It also registers 12 buttons, which is the same as the Hama gamepad, but they are just ordered differently.

In the middle there are three buttons: "Mode", which is the same as the "Analog" button on the Hama gamepad and is required to get the D-pad to work, and a "Turbo" and "Clear" button which do not seem to be recognized as extra buttons, but must have some internal meaning. This also means that it is missing two buttons that would normally be used as "Start" and "Select". These buttons appear as buttons 5 and 6 on the controller. The 4 shoulder buttons are once again just buttons and do not have triggers. I normally use L2 and R2 as substitute for "Start" and "Select". The D-pad is slightly different as well. This is actually the only controller that has completely separated buttons for the D-pad. On all other controllers, the D-pad is in some way a cross and all direction keys are somewhat connected, but not on the Sabrent.

I also tested the deadzone on this controller, and while the right analog stick is perfectly centering at 128 and stands there (values are between 0 and 255), the left analog stick does not return to the center and actually permanently jumps between 135 and 140 instead of 128, which means it's about 5-10% off. Since this controller acts similar to the Hama controller, it will also work fine in Retroarch without configuration.

Pros

- Cheap and light
- Very long cord 3m (about 9 feet)
- 6 button layout is very good for Sega Saturn and other emulators
- Works out of the box with retroarch

Cons

- Feels cheap and not very durable
- For my taste, this controller is slightly too small. It should be a little bigger, with buttons a little more separated. It also feels a little uncomfortable in the hands after a while
- Turbo and Clear buttons should rather work as Start and Select
- One analog stick does not center, but this is not the only controller that has this issue. The deadzone is not too bad

I've already talked about how these controllers are often detected automatically by Retroarch or another emulator. However, do they all work out of the box? The answer is sadly, no, but that does not mean you cannot use them.

On my ODROID GameStation Turbo image (OGST), I have a lot of different emulators and games that use controllers, so let's find out what is working. Currently, I have four major emulators that I find important to run with controllers: Retroarch – for most of the emulators, PPSSPP – Playstation Portable emulator, reicast – Sega Dreamcast emulator, and Yabause – a fairly new Sega Saturn emulator.

Retroarch is actually the best supported emulator when it comes to gamepad and joystick support. The way it is configured on my image is that it uses udev to determine the controller and has a large number of configuration files for different controllers so that they work out of the box without additional setup. About 130 different controllers are supported, so there is a very high chance that your controller is supported as well.

XBox 360 and PS3 work 100% out of the box with Retroarch. So does the Hama controller, and for most games, probably the Sabrent as well. The Trustmaster is detected correctly, but due to the broken left analog stick, it can cause some issues. Not all controllers are found correctly when it comes to rumble support. The N64 emulator and a few other cores can use rumble but, for some reason, they don't always work.

PPSSPP is actually not well supported when it comes to controllers. It officially only has a mapping for XBox 360 controllers, but that was actually broken on ODROIDs, so I created my own. I also added a controller mapping for the PS3 controller, which normally should work out of the box with the emulator, but I am not completely sure. I know that you can configure the PS3 controller manually, thanks to my changes, and map all the buttons you need correctly.

None of the other controllers are officially supported, so their use may not be guaranteed going forward. Generally, all controllers should work, since PPSSPP uses SDL2 as a backend, which should allow for some level of controller support. As long as you map all the keys you need, it should work fine for all controllers.

Reicast is a little tricky. It officially also only has support for the XBox 360 controller, but I also added configuration files for PS3. Other controllers are not officially supported, and are unlikely to work out of the box. You cannot remap the buttons within the emulator as with PPSSPP but there is a way that might work, which I will detail sometime in the future.

Since the last update, Reicast has rumble support. In my testing, all but the Trustmaster and PS3 controllers were working with Reicast and rumble support. The Trustmaster was said to not have rumble support, but the PS3 is detected as a rumble. However, it was either so weak that I could not feel it, or it is simply not working at all.

With Yabause, you have either the GTK or the Qt interface, both of which are configured separately. Buttons can be set up from within the emulator. In my experience, once the D-pad and the buttons are configured, they seem to work with every controller. Since you can reconfigure the buttons from within the emulator, this is not much of an issue.

As previously mentioned, having a configuration for different "Generic USB Joysticks" might not work, since they are not all configured in the same way. I also found that the PS3 controller on the C2 only works via wireless Bluetooth connection, and the cabled connection will not work.

Suitability

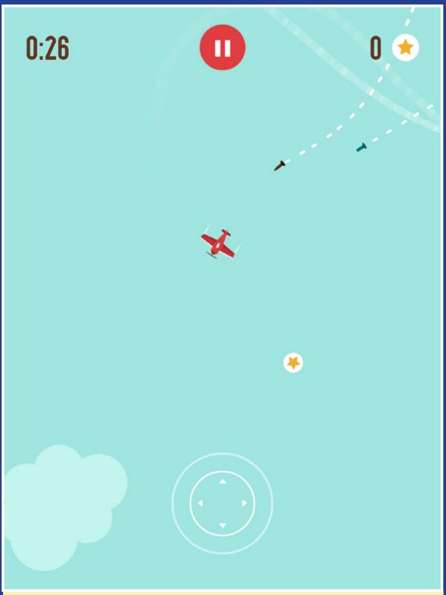
I normally have my entire image configured for the use of XBox 360 controllers. This makes it easier, since everything has the same configuration and you do not have to switch. I recently created my own unique setup so that I can switch between controllers depending on the games I play.

I still play all Libretro cores on Retroarch using an XBox 360 controller. It is often the most convenient controller, and I also can control Kodi with it just fine. I can navigate the menus without a mouse and keyboard, which is quite nice, especially since the XBox 360 controller is wireless. Since Retroarch uses automatic configuration for its controller setup, you can easily switch between controllers if you want to. I also use

MISSILES!
THERE IS NO BETTER WAY OF FLYING THAN DODGING RELENTLESS ATTACKS

by Bruno Doiche

The sky is blue and you are flying peacefully on your beloved airplane... Then suddenly you are fending for your life. This is Missiles, which is a great twist on a casual game. You are being shock-trained to try to survive for more than 10 seconds and the difficulty keeps going up. Although it is punishingly hard, this game will keep you glued to your screen!



As soon as you get aquainted with the game, go to fast mode. You will have double the fun in half the time!

https://play.google.com/store/apps/details?id=pl.macaque.Missiles

the XBox 360 controller for Dreamcast, since it works nicely with it.

For PPSSPP, I actually switched to use the PS3 controller. It is much more convenient to use a Playstation controller for a Playstation console since the buttons all match up and it is actually very responsive.

I got the Sabrent to play Sega Saturn games on the Yabause emulator, but I don't really use the Hama. I used it in the past when I didn't have my XBox 360 yet, but lately I only use it on a spare ODROID if I need to test something and do not want to switch my remote adapter for an XBox 360 controller.

I do not use the Trustmaster at all, because of its broken left analog stick, often causes issues. I actually find that kind of disappointing, because it is actually a very nice controller which feels good in the hands.

How to configure your controller for OGST

If you have a controller that is not supported by the emulators I mentioned above here is a small guide on how to setup your controller:

Retroarch: There should be a tool installed called "retroarch-joyconfig". With that tool, you can create a configuration file for your controller. It will generate text output, which you need to copy and create as a new file in /usr/share/libretro/autoconfig/udev/ using root privileges. After that, your controller should be found automatically by Retroarch.

PPSSPP: Go to the Settings -> Controls menu to setup your controller and map your buttons.

Yabause: Just like PPSSPP, you can configure the controls from within the emulator. For example, press CTRL + S in yabause-qt.

Reicast: For Reicast, you use a tool called reicast-joyconfig that is similar to retroarch-joyconfig It is written in Python and requires a python module called evdev, which you might have to install via pip. After that, it's similar to Retroarch in how you create the configuration file. This file must be copied to /usr/local/share/reicast/mappings using root privileges. Afterwards, you need to adjust the configuration file for Reicast under /home/odroid/.reicast/emu.cfg by following the guide in the forum at http://bit.ly/2ggdO9Y. You also need to create an entry for the mappings file similar to the following in order to load the new configuration for your controller:

```
evdev_mapping_1= mycontroller.cfg
```

I also have a tool installed called antimicro which allows you to create unique configurations for your controllers. You can map keyboard keys to your controller in case a game or an emulator does not want to work with your controller. You can even map your mouse movements and buttons to a controller and use the controller as a substitute.

Final thoughts

Whatever controller you like, there is probably a way to get it to work on my ODROID pre-built images. Some users have even been able to attach authentic arcade joysticks for use on full cabinets. Whatever controller you prefer, there should be a way to make your gaming experience enjoyable on ODROIDs and my OGST images.

INSTALLING A THERMAL RECEIPT PRINTER ON LINUX

CHRONICLES OF A MAD SCIENTIST

by Bo Lechnowsky (@respectech)

"One day, everything will be paperless," you mutter under your breath as you look at your workspace cluttered with a mixture of electronics, cables, and papers. In the future, you will only have to dig through just electronics and cables to find that lost eMMC module. But right now, looking for that lost eMMC or microSD card on your workspace feels like searching for a needle in a haystack, where most of the haystack is made of paper!

You notice that a lot of the papers are handy printouts that contain well needed information. "Most of these notes are pretty small, but they take up an entire sheet of paper!" You start thinking about the problem of paper size. Then you recall seeing a long-abandoned thermal printer from a previous world-domination scheme sitting in a box in the corner of your subterranean laboratory. You grab it and see it is a STAR TSP100 thermal printer with a USB interface.

"That's it!" You start planning how you can make use of a thermal printer to print out notes of any size. Many of the notes will be small enough to tape to the physical objects that you are making the notes about. "I can add a thermal printer to the console of my fleet of vehicles in case the need for printing anything may arise on the road!"

Your first attempt is to simply add it via "cups", accessible through the Linux menu for "System/Administration/Printers" or "System Tools/Printers". Assigning it to the "Generic Text Only" printer driver didn't work, and there isn't a download for Linux/ARM on the manufacturer's website.

So how can we print to a thermal printer from Linux running on ODROID? You research and print out the following steps (on a regular-sized sheet of paper for now) for later reference:

In a terminal window, enter:

```
$ sudo apt update
$ sudo apt install libcups2-dev
$ sudo apt install libcupsimage2
$ sudo apt install libcupsimage2-
dev
```

Next, download a package containing the Linux source code for the STAR TSP100 printer from http://bit.ly/2fPybtO. From the command line, navigate to the directory containing the download zip file and type:

```
$ unzip TSP100U_v5_2_0_CD.zip
```

Next, move into the source directory and untar the files:

```
$ cd TSP100_V520/Linux/CUPS
$ tar -zxvf starcupsdrv-3.3.0_
linux_20110428.tar.gz
$ cd starcupsdrv-3.3.0_linux/
SourceCode
$ tar -zxvf starcupsdrv-src-
3.3.0.tar.gz
$ cd starcupsdrv
```

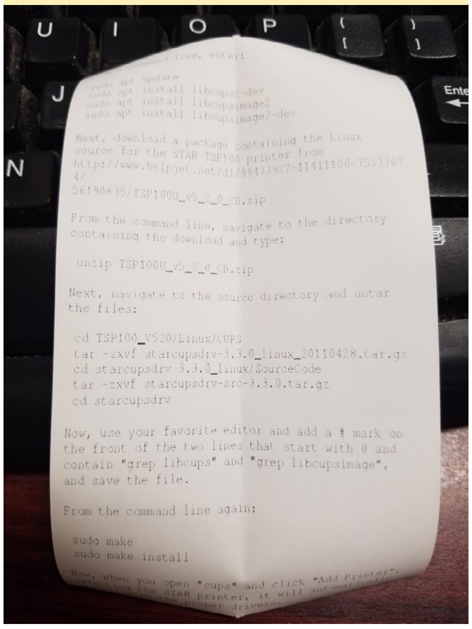
Use your favorite editor and add a # mark on the beginning of the two lines that start with @ and contain "grep libcups" and "grep libcupsimage", and save the file. Then, type the following into the command line:

```
$ sudo make
$ sudo make install
```

Now, when you open "cups" and click "Add Printer", selecting the STAR printer, it will automatically install with the proper drivers.

As you replace all the large printouts on your workspace with space-efficient thermal prints including this set of instructions, you think to yourself: "Today I conquered my workspace organization problems, tomorrow I'll conquer the world!"

Thermal printer with a receipt printed from an ODROID



```
 sudo apt update
 sudo apt install libcups2-dev
 sudo apt install libcupsimage2
 sudo apt install libcupsimage2-dev

Next, download a package containing the Linux
source for the STAR TSP100 printer from
http://www.helpjet.net/dl/99473927-11411100-7557707
4/
56190835/TSP100U_v5_2_0_CD.zip

From the command line, navigate to the directory
containing the download and type:

 unzip TSP100U_v5_2_0_CD.zip

Next, navigate to the source directory and untar
the files:

 cd TSP100_V520/Linux/CUPS
 tar -zxvf starcupsdrv-3.3.0_linux_20110428.tar.gz
 cd starcupsdrv 3.3.0_linux/SourceCode
 tar -zxvf starcupsdrv-src-3.3.0.tar.gz
 cd starcupsdrv

Now, use your favorite editor and add a # mark on
the front of the two lines that start with @ and
contain "grep libcups" and "grep libcupsimage",
and save the file.

From the command line again:

 sudo make
 sudo make install

Now, when you open "cups" and click "Add Printer",
```

BOOT.INI PERSISTENCE

PRESERVING CHANGES DURING AN UPGRADE

by Adrian Popa (@mad_ady)

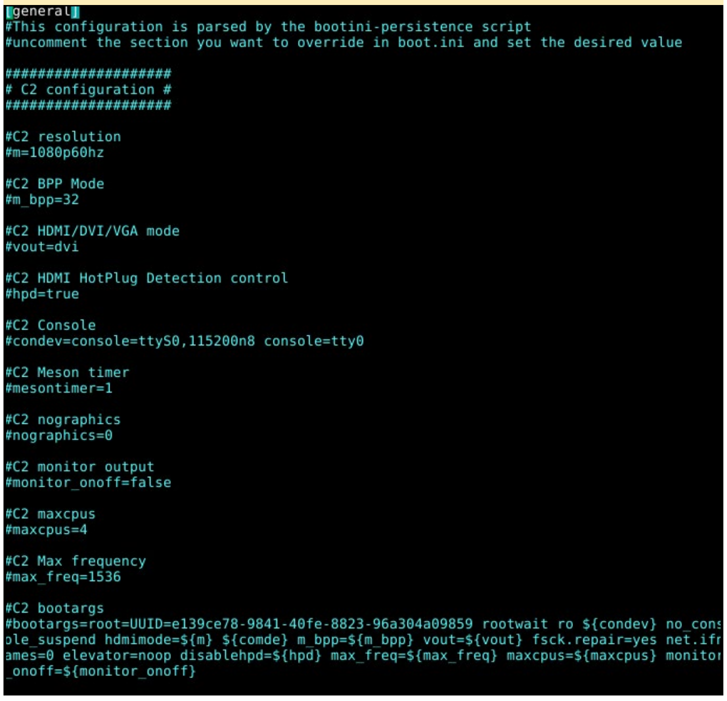
Picture this scenario: you come home, late at night, accompanied by a beautiful lady and you want to impress her by viewing a movie or listening to music on your fancy ODROID setup. You turn on the ODROID and the TV and notice that it keeps saying “No signal”. The blue led blinks as it should, but you feel a cold sweat when your date asks you what’s wrong. You politely excuse yourself and hide for a second in the bathroom, to “freshen up”. You use your phone to SSH into the ODROID to investigate the problem. You quickly realize that boot.ini had been overwritten by an update and that the resolution is not compatible with the TV. After fiddling with VI on your phone and rebooting the ODROID, you return to the living room. Now the room is filled with Kodi’s blueish glow and you are relieved. However, your date has felt something was amiss, and excuses herself for the evening. I wonder if that could have been avoided?

The problem is that when the bootini package updates, it overwrites the file /media/boot/boot.ini, and removes any customizations you had done on it, such as setting the resolution, enabling DAC support or even choosing a different root file system. Every new user stumbles over this problem and, so far, had to either give up updates or learn to live with it. Giving up on updates is a terrible choice, because you are missing out on bug fixes and new features, such as overclocking support. In order to fix this, I made some changes to the bootini package to restore user settings after a new boot.ini file is updated.

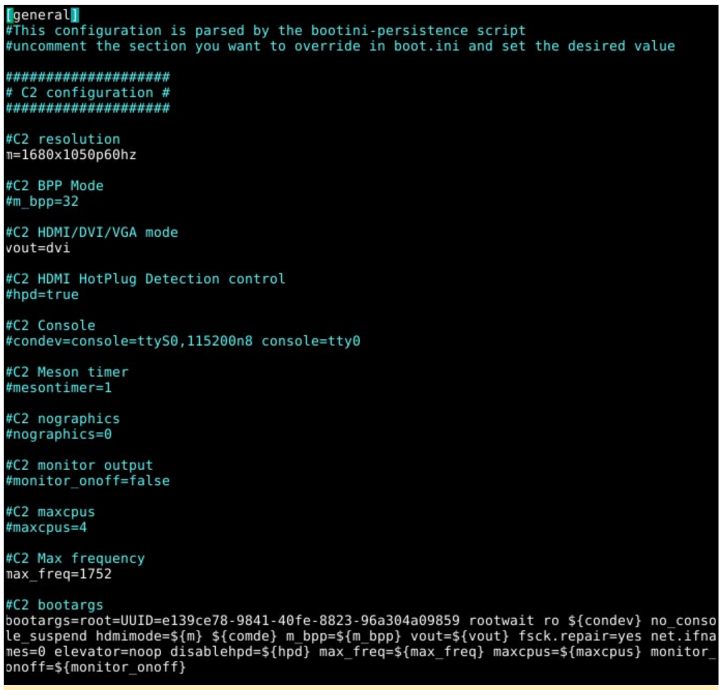
To use this mechanism, you will need to edit a file called boot.ini.default which is located in /media/boot, which is the vfat partition that is used in the boot process. This file contains commented out versions of all the default values of the boot.ini settings that you can adjust. This file will not be overwritten on future updates, but it will be recreated if you delete it.

You will need to uncomment the lines that interest you and set the values you want for those variables. For example, for the ODROID-C2 model, you can set things like resolution (“m”), video out mode (“vout”), maximum frequency (“max_freq”) as well as modify the boot arguments, like the root partition UUID. Figure 1 shows the default file, while Figure 2 shows a

Figure 1 - All settings in the boot.ini file are commented out by default



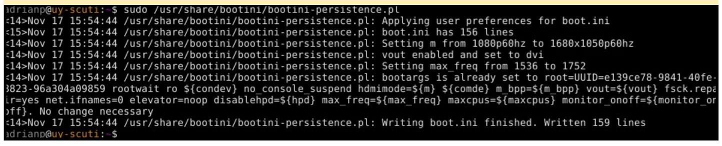
```
[general]
#This configuration is parsed by the bootini-persistence script
#uncomment the section you want to override in boot.ini and set the desired value

###################
# C2 configuration #
###################

#C2 resolution
#m=1080p60hz

#C2 BPP Mode
#m_bpp=32

#C2 HDMI/DVI/VGA mode
#vout=dvi

#C2 HDMI HotPlug Detection control
#hpd=true

#C2 Console
#condev=console=ttyS0,115200n8 console=tty0

#C2 Meson timer
#mesontimer=1

#C2 nographics
#nographics=0

#C2 monitor output
#monitor_onoff=false

#C2 maxcpus
#maxcpus=4

#C2 Max frequency
#max_freq=1536

#C2 bootargs
#bootargs=root=UUID=e139ce78-9841-40fe-8823-96a304a09859 rootwait ro ${condev} no_cons
ole_suspend hdmimode=${m} ${comde} m_bpp=${m_bpp} vout=${vout} fsck.repair=yes net.ifr
ames=0 elevator=noop disablehpd=${hpd} max_freq=${max_freq} maxcpus=${maxcpus} monitor
_onoff=${monitor_onoff}
```



```
[general]
#This configuration is parsed by the bootini-persistence script
#uncomment the section you want to override in boot.ini and set the desired value

###################
# C2 configuration #
###################

#C2 resolution
m=1680x1050p60hz

#C2 BPP Mode
#m_bpp=32

#C2 HDMI/DVI/VGA mode
vout=dvi

#C2 HDMI HotPlug Detection control
#hpd=true

#C2 Console
#condev=console=ttyS0,115200n8 console=tty0

#C2 Meson timer
#mesontimer=1

#C2 nographics
#nographics=0

#C2 monitor output
#monitor_onoff=false

#C2 maxcpus
#maxcpus=4

#C2 Max frequency
max_freq=1752

#C2 bootargs
bootargs=root=UUID=e139ce78-9841-40fe-8823-96a304a09859 rootwait ro ${condev} no_conso
le_suspend hdmimode=${m} ${comde} m_bpp=${m_bpp} vout=${vout} fsck.repair=yes net.ifna
mes=0 elevator=noop disablehpd=${hpd} max_freq=${max_freq} maxcpus=${maxcpus} monitor_
onoff=${monitor_onoff}
```

Figure 2 - The desired boot.ini options have been set

Figure 3 - Console output of the script bootini-persistence.pl



```
adrianp@uy-scuti:~$ sudo /usr/share/bootini/bootini-persistence.pl
<14>Nov 17 15:54:44 /usr/share/bootini/bootini-persistence.pl: Applying user preferences for boot.ini
<15>Nov 17 15:54:44 /usr/share/bootini/bootini-persistence.pl: boot.ini has 156 lines
<14>Nov 17 15:54:44 /usr/share/bootini/bootini-persistence.pl: Setting m from 1080p60hz to 1680x1050p60hz
<14>Nov 17 15:54:44 /usr/share/bootini/bootini-persistence.pl: vout enabled and set to dvi
<14>Nov 17 15:54:44 /usr/share/bootini/bootini-persistence.pl: Setting max_freq from 1536 to 1752
<14>Nov 17 15:54:44 /usr/share/bootini/bootini-persistence.pl: bootargs is already set to root=UUID=e139ce78-9841-40fe-
8823-96a304a09859 rootwait ro ${condev} no_console_suspend hdmimode=${m} ${comde} m_bpp=${m_bpp} vout=${vout} fsck.repa
ir=yes net.ifnames=0 elevator=noop disablehpd=${hpd} max_freq=${max_freq} maxcpus=${maxcpus} monitor_onoff=${monitor_on
off}. No change necessary
<14>Nov 17 15:54:44 /usr/share/bootini/bootini-persistence.pl: Writing boot.ini finished. Written 159 lines
adrianp@uy-scuti:~$
```

customized version. Leave commented the parameters that you don't need, and they will be ignored.

In order to apply new settings and changes to boot.ini.default, you can reconfigure the bootini package with the following command:

```
$ sudo dpkg-reconfigure bootini
```

Alternatively, you can just run the bootini-persistence.pl script:

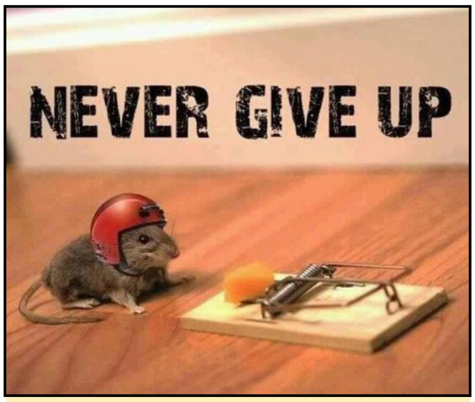
```
$ sudo /usr/share/bootini/booti-
ni-persistence.pl
```

The output shown will tell you which parameters have been changed, as shown in Figure 3.

To return to the original boot.ini file, you can recopy it from the file /usr/share/bootini/boot.ini with the following command:

```
$ sudo cp /usr/share/bootini/
boot.ini /media/boot/boot.ini
```

I have submitted a GitHub pull request to the Hardkernel engineers for these improvements to be included with their official release, which should be available soon. I hope that these changes will make your life easier on Linux on all of your present and future ODROIDs.



Persistence pays off!

SOFTWARE EQUALIZER FOR ANDROID

CHRONICLES OF A MAD SCIENTIST

by Bo Lechnowsky (@respectech)

Your mind races as you sit in your dark lab. The latest hurdle you've encountered threatens to derail your world domination plans. "What changed?" you wonder, as you evaluate each of the ideas that floods your mind. You think back on the events that led up to this unwelcome detour of your plans.

You were driving one of your vehicles to pick up some supplies for my latest invention. You started up your in-car touchscreen unit powered by the ODROID-C2 and began playing your "world domination" playlist. Then, it hit you like that time your nemesis, Dr. Usual, tested out his nausea ray gun on you. You felt a little sick. You had the equalizer settings on Android set to emphasize the heavy bass on your favorite tracks but, whenever there was a bass hit, the vocals and other high range notes would cut out, and the bass hits didn't have any punch to them.

Your incredible Software Equalizer in Android.

How can you proceed with your world domination plans without punchy bass and strong high-range notes? It's unthinkable! After spending the whole night trying to forget about it and work on other aspects of your inventions, it finally comes to you. "Eureka! I'm sure it has something to do with the master volume level in Android, and the software equalizer settings hitting the volume ceiling when trying to modify the audio stream!"

You rush to your lair's garage, start up your in-vehicle system and check the master volume settings in Android. They were set to 100%, just as you had suspected! You quickly activate the sound system and subwoofers and turn up the volume level on the sound system while reducing the master volume setting down to 50% on Android. You nervously start your playlist. It hits you like Dr. Usual's nausea ray gun, except with a lot more force. "I did it!" you yell, with your voice drowned out by the sounds of industrial space opera electronica thumping through the speakers. "This deserves a celebration," you think, as you slowly drive with pulsating speakers to the ice cream shop, until you realize it's 5:30am and they're not open yet.

USING THE HARDWARE ENCODER FUNCTIONS OF THE ODROID-XU4

by Marian Mihailescu (@memeka)

Hardware encoders use a designed algorithm to encode video and data into streamable content, and is generally the most efficient way to watch video. Hardware encoding on the ODROID-XU4 can be achieved using two options:

- a custom FFmpeg which supports the hardware encoder, or
- the GStreamer framework available for creating streaming media applications

The steps to enable hardware encoding presented below are grouped into common steps and specific steps. Please note that this tutorial is designed for medium/advanced users, and if things go wrong, you might have an unbootable system. It might be wise to make a backup before starting by following the steps listed at http://bit.ly/2gg5KGc.

Common steps

First, install a mainline kernel that supports the MFC encoder. The instructions here use @elatllat's branch (http://bit.ly/2gg82Fj), but @mdrjr is also working on a branch of his own (http://bit.ly/2g2pVVc) and so am I (http://bit.ly/2gf0dfh). You can use my kernel configuration, which tries to add all of HardKernel's modules, or you can modify my configuration to fit your needs (http://bit.ly/2gAr75I). You can alternatively use whatever default configuration comes with the kernel you choose.

The kernel does not support HMP (big.LITTLE extensions) and it treats all cores as equal. This is because HMP patches are unstable and can lock up the system. Consequently, overall system performance will be lower than when using Kernel 3.10.

I have added the kernel compilation procedure here for convenience, but you should read and discuss the official kernel compilation thread at http://bit.ly/2fo18cv, or review the guide at http://bit.ly/1NVRprY.

First, make a copy of your existing kernel, initrd, dtb and boot.ini:

```
$ cd /media/boot
$ sudo -i
# cp zImage zImage-3
# cp uInitrd uInitrd-3
# cp exynos5422-odroidxu3.dtb
exynos5422-odroidxu3-3.dtb
# cp boot.ini boot3.ini
```

Edit boot3.ini to point to the newly copied files by appending -3 to the name of the zImage, uInitrd and dtb. By replacing the stock boot.ini with this modified boot.ini you will be able to boot your old kernel in case of problems.

Next, download the new kernel:

```
$ sudo apt-get -y install bc curl
\
 gcc git libncurses5-dev lzop \
 make u-boot-tools dos2unix
$ git clone --depth 1 \
 -b odroidxu4-mihailescu2m-4.8 \
 https://github.com/Dmole/linux.
git linux
$ cd linux
$ make odroidxu4_defconfig
```

Optionally, you can get my config which supports most of the USB peripherals (TV tuners, sound cards, wifi cards) and a lot of networking modules (LXC support, VLANs, iptables):

```
$ wget http://pastebin.com/
raw/7YnakKmP -O .config
$ dos2unix .config
```

Next, compile the kernel:

```
$ make menuconfig
$ make -j 8 zImage dtbs modules
$ kver=`make kernelrelease`
$ sudo cp arch/arm/boot/zImage
arch/arm/boot/dts/exynos5422-
odroidxu[34].dtb /media/boot
$ sudo cp .config /media/boot/
config
$ sudo make INSTALL_MOD_STRIP=1
modules_install
$ sudo make firmware_install
$ sudo cp .config /boot/config-
${kver}
```

```
$ cd /boot
$ sudo update-initramfs -c -k
${kver}
$ sudo mkimage -A arm -O linux -T
ramdisk -a 0x0 -e 0x0 -n initrd.
img-${kver} -d initrd.img-${kver}
uInitrd-${kver}
$ sudo cp uInitrd-${kver} /media/
boot/uInitrd
```

Modify /media/boot/boot.ini and load exynos5422-odroidxu4.dtb instead of exynos5422-odroidxu3.dtb and save boot.ini. Shut the system down and unplug the HDMI and power cables. Without this step, when you boot with the new kernel, you will not have USB3 bus and onboard networking. This is needed only when switching between 3.x and 4.x kernels as far as I've seen. Make sure to re-attach the cables prior to power up.

Steps when using FFMPEG

Now that the kernel is ready, compile/install a custom ffmpeg which supports the hardware encoder. Compile your own ffmpeg using the commands:

```
# debian build tools
$ sudo apt-get install build-es-
sential fakeroot devscripts \
 libchromaprint-dev librubber-
band-dev libjs-bootstrap
# get the patched ffmpeg version
$ git clone -b v4l2_m2m-3.0.2
--depth=1 \
 https://github.com/mihailescu2m/
FFmpeg.git
# install ffmpeg's build depen-
dencies (~190 packages)
$ sudo apt-get build-dep ffmpeg
# build ffmpeg as deb packages
with no checks (some checks fail)
$ cd FFmpeg
$ DEB_BUILD_OPTIONS="nocheck" de-
build -b -uc -us
```

When building is done, you should have 23 deb packages one level up, in the same directory as FFmpeg. You can download these packages pre-compiled from here: http://bit.ly/2g2m2iZ. Then, install the needed packages:

```
$ cd ../
$ sudo dpkg -i *.deb
```

This should replace any system-installed ffmpeg which supports the hardware encoder. Let us now use the hardware encoder when transcoding. Depending on what you want to transform with ffmpeg, you might get better or worse framerate. For example, changing the output fps of a video cuts off about 20-30fps from encoding time. The examples below do not change the fps:

```
# encode video only, about 50fps
(max)
$ ffmpeg -i big_buck_bunny_720p_
h264.mov -acodec aac -vcodec h264
-b:v 2M -pix_fmt nv21 bbb.mp4


# encode video only, about 110fps
(max)
$ ffmpeg -i big_buck_bunny_720p_
h264.mov -codec:v copy -codec:a
none -bsf:v h264_mp4toannexb
-f rawvideo - | ffmpeg -r 24
-i - -an  -vcodec h264 -b:v 2M
-profile:v 10 -pix_fmt nv21 bbb.
mp4


#encode video and audio, about
75fps (max)
$ ffmpeg -i big_buck_bunny_720p_
h264.mov -codec:v copy -codec:a
none -bsf:v h264_mp4toannexb -f
rawvideo - | ffmpeg -r 24 -i -
-i big_buck_bunny_720p_h264.mov
-map 0:v:0 -vcodec h264 -b:v 2M
-profile:v 10 -pix_fmt nv21  -map
1:a:0 -c:a:1 aac bbb.mp4
```

The encoder automatically selects h264_v4l2m2m which does the hardware encoding:

```
Stream #0:0 -> #0:0 (h264 (na-
tive) -> h264 (h264_v4l2m2m))
Stream #0:1 -> #0:1 (ac3 (native)
-> aac (native))
```

You may see the following errors:

```
[h264_v4l2m2m @ 0xf3fa0] H264
codec detected, init annexb con-
verter
[h264_v4l2m2m @ 0xf3fa0] Device
path not set, probing /dev/video*
[h264_v4l2m2m @ 0xf3fa0] exynos-
gsc.1:m2m is not the one we want
[h264_v4l2m2m @ 0xf3fa0] exynos-
gsc.0:m2m is not the one we want
[h264_v4l2m2m @ 0xf3fa0] s5p-mfc-
dec is not the one we want
[h264_v4l2m2m @ 0xf3fa0] Could
not find a valid device
```

If so, make sure the user you are running ffmpeg with, is part of the video group:

```
odroid@odroid:~$ id
uid=1000(odroid) gid=1000(odroid)
groups=1000(odroid),4(adm),20(dia
lout),24(cdrom),27(sudo),30(dip),
44(video),46(plugdev),115(lpadmin
),116(lightdm)
```

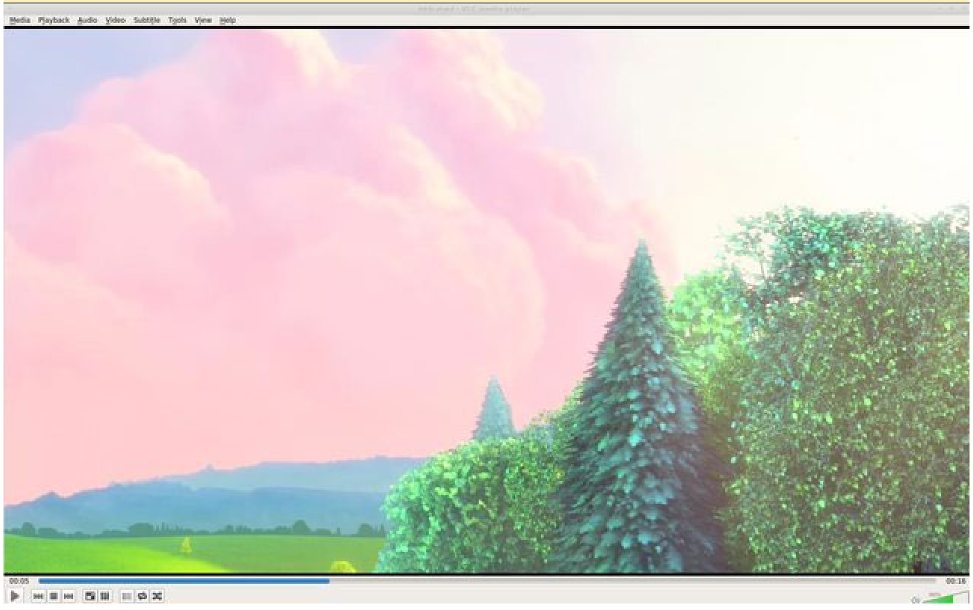
Sample output is as follows:

```
adrianp@odroid:~> ffmpeg -i
Sintel.2010.720p.mkv -acodec aac
-vcodec h264 -b:v 2M  -pix_fmt
nv21 sintel-encoded.mp4
ffmpeg version 3.0.2-1ubuntu4
Copyright (c) 2000-2016 the FFm-
peg developers
  built with gcc 5.4.0 (Ubuntu/
Linaro 5.4.0-6ubuntu1~16.04.2)
20160609
  configuration: --prefix=/
usr --extra-version=1ubuntu4
--toolchain=hardened --libdir=/
usr/lib/arm-linux-gnueabihf
--incdir=/usr/include/arm-linux-
```

```
gnueabihf --cc=cc --cxx=g++
--enable-gpl --enable-shared
--disable-stripping --disable-
decoder=libopenjpeg --disable-
decoder=libschroedinger --enable-
avresample --enable-avisynth
--enable-gnutls --enable-ladspa
--enable-libass --enable-lib-
bluray --enable-libbs2b --en-
able-libcaca --enable-libcdio
--enable-libflite --enable-lib-
fontconfig --enable-libfreetype
--enable-libfribidi --enable-
libgme --enable-libgsm --enable-
libmodplug --enable-libmp3lame
--enable-libopenjpeg --enable-
libopus --enable-libpulse --en-
able-librubberband --enable-
librtmp --enable-libschroedinger
--enable-libshine --enable-lib-
snappy --enable-libsoxr --en-
able-libspeex --enable-libssh
--enable-libtheora --enable-libt-
wolame --enable-libvorbis --en-
able-libvpx --enable-libwavpack
--enable-libwebp --enable-libx265
--enable-libxvid --enable-libz-
vbi --enable-openal --enable-
opengl --enable-x11grab --enable-
libdc1394 --enable-libiec61883
--enable-libzmq --enable-frei0r
--enable-chromaprint --enable-
libx264
  WARNING: library configuration
mismatch
  avcodec     configura-
tion: --prefix=/usr --ex-
tra-version=1ubuntu4
--toolchain=hardened --libdir=/
usr/lib/arm-linux-gnueabihf
--incdir=/usr/include/arm-linux-
gnueabihf --cc=cc --cxx=g++
--enable-gpl --enable-shared
--disable-stripping --disable-
decoder=libopenjpeg --disable-
decoder=libschroedinger --enable-
avresample --enable-avisynth
--enable-gnutls --enable-ladspa
--enable-libass --enable-lib-
bluray --enable-libbs2b --en-
able-libcaca --enable-libcdio
--enable-libflite --enable-lib-
fontconfig --enable-libfreetype
--enable-libfribidi --enable-
libgme --enable-libgsm --enable-
libmodplug --enable-libmp3lame
--enable-libopenjpeg --enable-
libopus --enable-libpulse --en-
able-librubberband --enable-
librtmp --enable-libschroedinger
--enable-libshine --enable-lib-
snappy --enable-libsoxr --en-
able-libspeex --enable-libssh
--enable-libtheora --enable-
libtwolame --enable-libvorbis
--enable-libvpx --enable-lib-
wavpack --enable-libwebp --en-
able-libx265 --enable-libxvid
--enable-libzvbi --enable-openal
--enable-opengl --enable-x11grab
--enable-libdc1394 --enable-li-
biec61883 --enable-libzmq --en-
able-frei0r --enable-chromaprint
--enable-libx264 --enable-ver-
sion3 --disable-doc --disable-
programs --disable-avdevice
--disable-avfilter --disable-
avformat --disable-avresample
--disable-postproc --disable-
swscale --enable-libopencore_am-
rnb --enable-libopencore_amrwb
--enable-libvo_amrwbenc
  libavutil      55. 17.103 / 55.
17.103
  libavcodec     57. 24.102 / 57.
24.102
  libavformat    57. 25.100 / 57.
25.100
  libavdevice    57.  0.101 / 57.
0.101
  libavfilter     6. 31.100 /  6.
31.100
  libavresample   3.  0.  0 /  3.
0.  0
  libswscale      4.  0.100 /  4.
0.100
  libswresample   2.  0.101 /  2.
0.101
  libpostproc    54.  0.100 / 54.
0.100
Input #0, matroska,webm, from
'Sintel.2010.720p.mkv':
  Metadata:
    encoder         : libebml
v1.0.0 + libmatroska v1.0.0
    creation_time   : 2011-04-24
17:20:33
  Duration: 00:14:48.03, start:
0.000000, bitrate: 6071 kb/s
    Chapter #0:0: start 0.000000,
end 103.125000
    Metadata:
      title           : Chapter
01
    Chapter #0:1: start
103.125000, end 148.667000
    Metadata:
      title           : Chapter
02
    Chapter #0:2: start
148.667000, end 349.792000
    Metadata:
      title           : Chapter
03
    Chapter #0:3: start
349.792000, end 437.208000
    Metadata:
      title           : Chapter
04
    Chapter #0:4: start
437.208000, end 472.075000
    Metadata:
      title           : Chapter
05
    Chapter #0:5: start
472.075000, end 678.833000
    Metadata:
      title           : Chapter
06
    Chapter #0:6: start
678.833000, end 744.083000
    Metadata:
      title           : Chapter
07
    Chapter #0:7: start
744.083000, end 888.032000
    Metadata:
      title           : Chapter
08
    Stream #0:0(eng): Video:
h264 (High), yuv420p(tv, bt709/
unknown/unknown), 1280x544, SAR
1:1 DAR 40:17, 24 fps, 24 tbr, 1k
```

```
tbn, 48 tbc
    Stream #0:1(eng): Audio: ac3,
48000 Hz, 5.1(side), fltp, 640
kb/s
    Metadata:
      title           : AC3 5.1 @
640 Kbps
    Stream #0:2(ger): Subtitle:
subrip
    Stream #0:3(eng): Subtitle:
subrip
    Stream #0:4(spa): Subtitle:
subrip
    Stream #0:5(fre): Subtitle:
subrip
    Stream #0:6(ita): Subtitle:
subrip
    Stream #0:7(dut): Subtitle:
subrip
    Stream #0:8(pol): Subtitle:
subrip
    Stream #0:9(por): Subtitle:
subrip
    Stream #0:10(rus): Subtitle:
subrip
    Stream #0:11(vie): Subtitle:
subrip
Codec AVOption preset (Configura-
tion preset) specified for output
file #0 (sintel-encoded.mp4) has
not been used for any stream.
The most likely reason is either
wrong type (e.g. a video option
with no video
 streams) or that it is a private
option of some encoder which was
not actually used for any stream.
File 'sintel-encoded.mp4' already
exists. Overwrite ? [y/N] y
[h264_v4l2m2m @ 0xf3fe0] H264
codec detected, init annexb con-
verter
[h264_v4l2m2m @ 0xf3fe0] Device
path not set, probing /dev/video*
[h264_v4l2m2m @ 0xf3fe0] exynos-
gsc.1:m2m is not the one we want
[h264_v4l2m2m @ 0xf3fe0] exynos-
gsc.0:m2m is not the one we want
[h264_v4l2m2m @ 0xf3fe0] Using
device /dev/video1
Output #0, mp4, to 'sintel-encod-
ed.mp4':
  Metadata:
    encoder         :
Lavf57.25.100
    Chapter #0:0: start 0.000000,
end 103.125000
    Metadata:
      title           : Chapter
01
    Chapter #0:1: start
103.125000, end 148.667000
    Metadata:
      title           : Chapter
02
    Chapter #0:2: start
148.667000, end 349.792000
    Metadata:
      title           : Chapter
03
    Chapter #0:3: start
349.792000, end 437.208000
    Metadata:
      title           : Chapter
04
    Chapter #0:4: start
437.208000, end 472.075000
    Metadata:
      title           : Chapter
05
    Chapter #0:5: start
472.075000, end 678.833000
    Metadata:
      title           : Chapter
06
    Chapter #0:6: start
678.833000, end 744.083000
    Metadata:
      title           : Chapter
07
    Chapter #0:7: start
744.083000, end 888.032000
    Metadata:
      title           : Chapter
08
    Stream #0:0(eng): Video:
h264 (h264_v4l2m2m) ([33][0][0]
[0] / 0x0021), yuv420p, 1280x544
[SAR 1:1 DAR 40:17], q=2-31, 2000
kb/s, 24 fps, 12288 tbn, 24 tbc
    Metadata:
      encoder         :
Lavc57.24.102 h264_v4l2m2m
    Stream #0:1(eng): Audio: aac
(LC) ([64][0][0][0] / 0x0040),
48000 Hz, 5.1(side), fltp, 341
kb/s
    Metadata:
      title           : AC3 5.1 @
640 Kbps
      encoder         :
Lavc57.24.102 aac
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (na-
tive) -> h264 (h264_v4l2m2m))
  Stream #0:1 -> #0:1 (ac3 (na-
tive) -> aac (native))
```

Figure 1 - Big Buck Bunny demo



```
Press [q] to stop, [?] for help
[h264_v4l2m2m @ 0xf3fe0] Perform-
ing useless memcpy() on output
pool because buffers do not match
[h264_v4l2m2m @ 0xf3fe0] This
could be avoided by using av_
v4l_buffer_pool_get_buffer*() or
av_v4l_buffer_pool_make_pipe()
[mp4 @ 0xb9c70] Timestamps are
unset in a packet for stream 0.
This is deprecated and will stop
working in the future. Fix your
code to set the timestamps prop-
erly
[mp4 @ 0xb9c70] Encoder did not
produce proper pts, making some
up.
[h264_v4l2m2m @ 0xf3fe0] No
event occurred while wait-
ing.01 bitrate=36881.3kbits/s
speed=1.84x
frame= 2027 fps= 44 q=-0.0
Lsize=  383559kB time=00:01:25.20
bitrate=36877.0kbits/s
speed=1.84x
video:379964kB audio:3551kB
subtitle:0kB other streams:0kB
global headers:0kB muxing over-
head: 0.011593%
```

Steps when using gstreamer

Compile gstreamer from memeka's branch using the commands:

```
$ apt-get install gstreamer1.0-
plugins-bad
$ git clone https://github.com/\
 mihailescu2m/gst-plugins-good
$ cd gst-plugins-good/
$ sudo apt-get install
libgstreamer1.0-dev libgudev-1.0-
dev \
 libgstreamer-plugins-base1.0-dev
dh-autoreconf automake autoconf \
 libtool autopoint cdbs gtk-doc-
tools libshout3-dev libaa1-dev \
 libflac-dev libdv4-dev libdv-dev
libgtk-3-dev libtag1-dev \
 libsoup2.4-dev gstreamer1.0-doc
gstreamer1.0-plugins-base-doc
```

```
$ dpkg-buildpackage -us -uc -b
-j4
$ cd ../
$ sudo dpkg -i gstreamer*.deb
```

Alternatively, you can get the pre-built deb packages from http://bit.ly/2gj7Iqm. To transcode something, you should first identify the encoding and the decoding interfaces, since they change on every boot:

```
$ decoder=`gst-inspect-1.0 | grep
v4l2 | grep videodec | cut -d ":"
-f 2`
$ encoder=`gst-inspect-1.0 | grep
v4l2 | grep h264enc | cut -d ":"
-f 2`
$ gst-launch-1.0 filesrc
location=big_buck_bunny_720p_
h264.mov ! qtdemux ! h264parse
! $decoder !  $encoder ex-
tra-controls="encode,h264_
level=10,h264_profile=4,frame_lev-
el_rate_control_enable=1,video_
bitrate=2097152" ! h264parse
! matroskamux ! filesink
location=bbb.mkv
```

For comments, questions and suggestions, please visit the original post at http://bit.ly/2g0vnsn.

Your dog will be pleased with your ODROID-U4's hardware encoding!

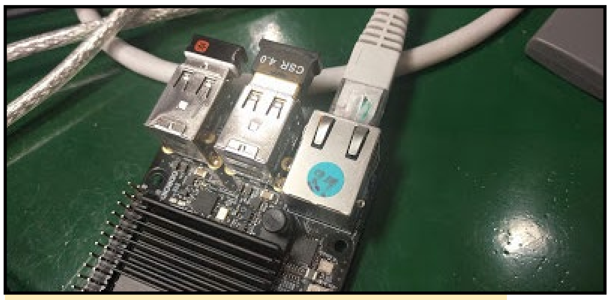
HOW TO USE AN ODROID AS A BLUETOOTH SPEAKER

MUSIC FOR THE MASSES

by @codewalker

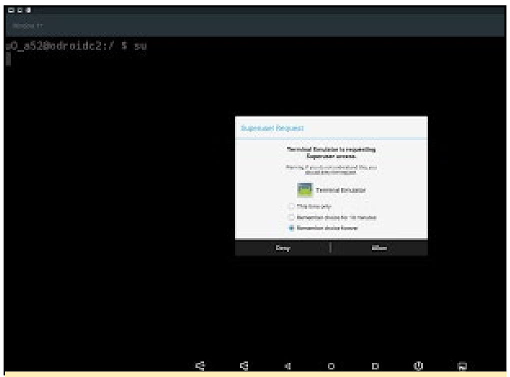
If you have an ODROID installed as a Car PC, you may want to keep your music collection on a portable device such as a phone or tablet, and have it automatically connect to the vehicle's speakers when you drive. Or, you may have a HiFi Shield attached to your ODROID-C2 along with a high-end stereo system, and want to play music from a friend's smartphone through it. By following the steps in this article, you can use your ODROID as a bluetooth speaker.

To begin, make sure that you have a bluetooth dongle attached to the ODROID, as shown below.



ODROID with Bluetooth dongle on USB port

Next, open the Terminal Emulator app on your ODROID, as shown.



```
u0_a52@odroidc2:/ $ su
```

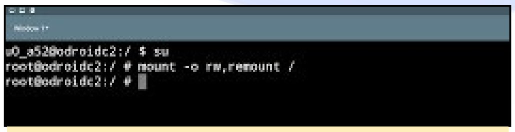
Launching the Terminal emulator

Then, switch to the root user and confirm the dialog box that appears:

```
$ su
```

Remount the root filesystem with read/write privileges so that you can make changes:

```
# mount -o rw,remount /
```

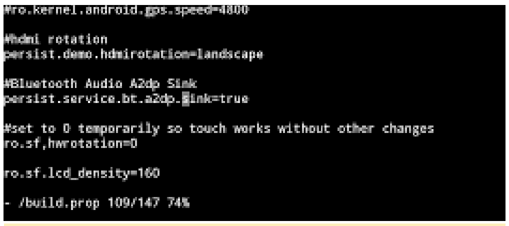


```
u0_a52@odroidc2:/ $ su
root@odroidc2:/ # mount -o rw,remount /
root@odroidc2:/ #
```

Running the mount command

Use a text editor such as vi to edit the file called "build.prop":

```
# vi /build.prop
```



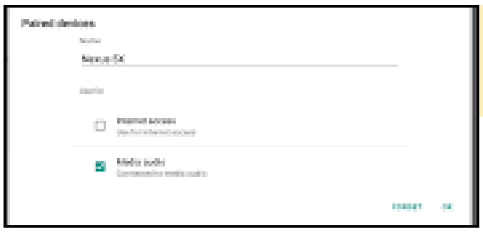
```
#ro.kernel.android.gps.speed=4800

#hdmi rotation
persist.demo.hdmirotation=landscape

#Bluetooth Audio A2dp Sink
persist.service.bt.a2dp.sink=true

#set to 0 temporarily so touch works without other changes
ro.sf.hwrotation=0

ro.sf.lcd_density=160

- /build.prop 109/147 74%
```

Editing the build.prop file using vi

After the file has loaded, add the following two lines to the end, save the file, and reboot the system:

```
#Bluetooth Audio A2dp Sink
persist.service.bt.a2dp.sink=true
```

After the ODROID has rebooted, enable the "Media Audio" option dialog box, as below:



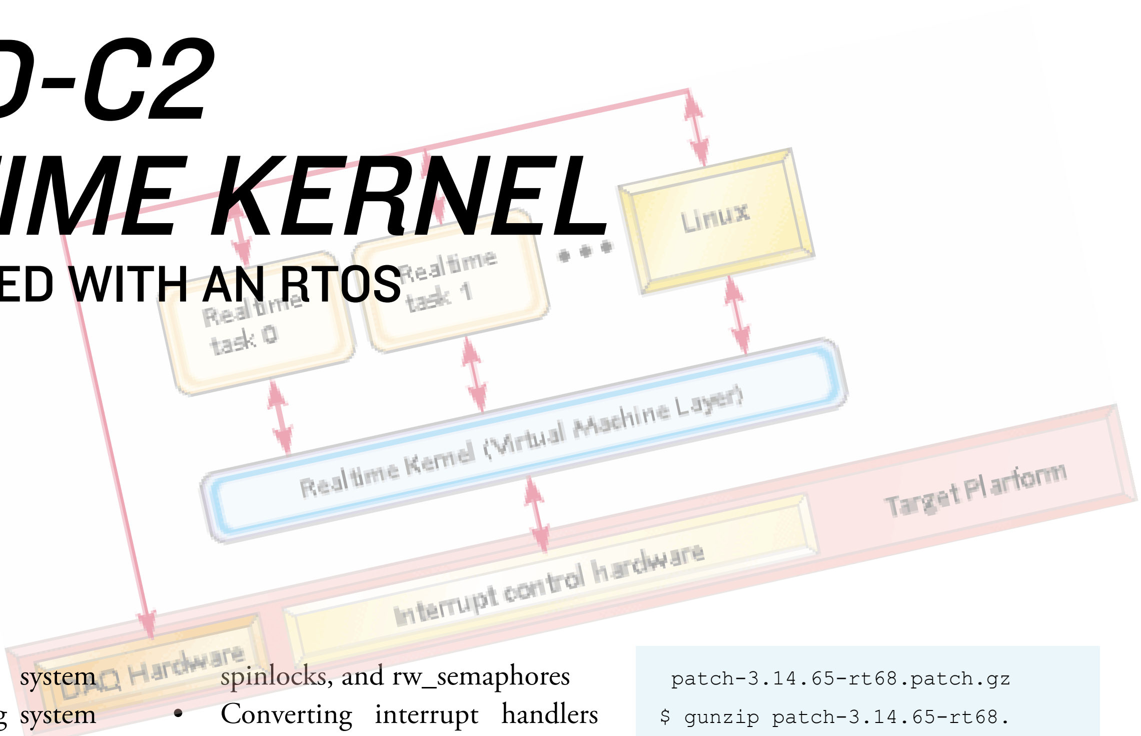
Enabling the Media Audio option

Finally, connect the smartphone or tablet to the ODROID by pairing it via bluetooth, and play your music. For comments, questions and suggestions, please visit the original post at `http://bit.ly/2f2dO8H`.

ODROID-C2 REAL-TIME KERNEL

GETTING STARTED WITH AN RTOS

by Anand Moon (@moon.linux)



Linux
Real time task 0
Real time task 1
Real time Kernel (Virtual Machine Layer)
Target Platform
Interrupt control hardware
DAQ Hardware

A real-time operating system (RTOS) is an operating system intended to serve real-time application process data as it comes in, typically without buffering delays. The standard Linux kernel only meets some real-time requirements by providing basic POSIX operations for userspace time handling, but it does not guarantee hard timing deadlines. If we apply Ingo Molnar's real-time Preemption patch (RT-Preempt), and Thomas Gleixner's generic clock event layer with high resolution support, the kernel gains full real-time capabilities.

The RT-Preempt patch has raised a lot of interest throughout the industry. Its clean design and consequent aim towards mainline integration makes it an interesting option for hard and firm real-time applications. It's no surprise to see applications ranging from professional audio to industrial control using RT Linux.

Use cases

- Making in-kernel locking-primitives (using spinlocks) preemptible though reimplementation with rtmutexes.
- Critical sections protected by spinlock_t and rwlock_t are now preemptible
- Implementing priority inheritance for in-kernel mutexes, spinlocks, and rw_semaphores
- Converting interrupt handlers into preemptible kernel threads
- Converting the old Linux timer API into separate infrastructures for high resolution kernel timers plus one for timeouts
- Timer improvements leading to userspace POSIX timers with high resolution.

Building the kernel

To build a real-time (RT) Linux kernel, we need to update the kernel with Real Time kernel patches. The RT patches file can be found at http://bit.ly/2g3MiJ2, and the latest patches for the 3.14.x kernel can be found at http://bit.ly/2goVUSQ. You can checkout the Linux kernel source tree for the ODROID-C2 from http://bit.ly/2fNFOi4. RT patches need to match the Linux kernel version, so you must choose the relevant patch series.

```
$ git clone --depth 1 \
 -b odroidc2-3.14.y \
 https://github.com/hardkernel/
linux.git\
 odroidc2-3.14.y-rt
$ cd odroidc2-3.14.y-rt
$ wget \
 https://www.kernel.org/pub/
linux/kernel/\
 projects/rt/3.14/older/\
 patch-3.14.65-rt68.patch.gz
$ gunzip patch-3.14.65-rt68.
patch.gz
$ patch -p1 < \
 patch-3.14.65-rt68.patch
```

The ARM64 RT kernel has some missing patches, so we have to look at the 3.18.y kernel and apply those patches as well. Once the additional patches are applied, we can build the kernel. The necessary patches can be found in 3.18.y-rt patch series:

arm64: Mark PMU interrupt IRQF_NO_THREAD.patch
arm64: Allow forced irq threading.patch
arch/arm64: Add lazy preempt support.patch
arm64: replace read_lock to rcu lock in call_step_hook

A few files have conflicts, so you need to update them manually before building the kernel. You can find the links for all the updated patches in the following repository at http://bit.ly/2g6R3Di.

Building the kernel

Compilation is done with make. Adding -j to the make command will speed up compilation:

```
$ make -j4 Image dtbs modules
```

After this has completed, you have will have a compiled Linux kernel (image), the device tree file (.dtb) and kernel modules (.ko). The following steps assume that your USB memory card reader is assigned to /dev/sdc. Be careful and double check how your card is assigned!

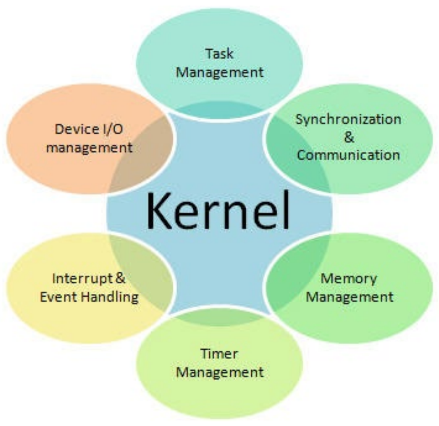
First, insert the boot-medium, either the eMMC module or SD card, into the USB memory card reader and connect the USB memory card reader to your Linux host PC. Then, copy the image and DT (meson64_odroidc2.dtb) to the 1st partition (FAT) of the boot-medium:

```
$ mkdir -p mount
$ sudo mount /dev/sdc1 ./mount
$ sudo cp arch/arm64/boot/Im-
age\ arch/arm64/boot/dts/meson64_
odroidc2.dtb\
 ./mount && sync && sudo umount
./mount
```

Finally, copy the driver modules to the 2nd partition(EXT4) of the boot-medium:

```
$ sudo mount /dev/sdc2 ./mount
$ sudo make modules_install \
ARCH=arm64 \
INSTALL_MOD_PATH=./mount && sync
\
&& sudo umount ./mount
$ rm -rf mount
```

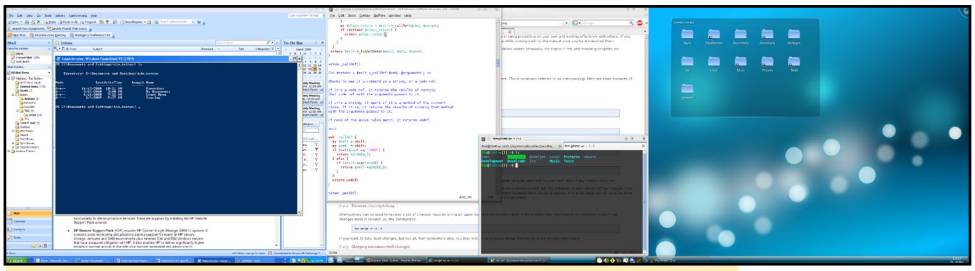
For comments, questions or suggestions, please visit the original post at http://bit.ly/2f2b0s8.



RULING THE WORLD WITH SYNERGY

CHRONICLES OF A MAD SCIENTIST

by Bo Lechnowsky (@respectech)



Run even heterogeneous environment X86 Windows x ARM Linux

If you had a minion, not the animated kind but the "Pull the Switch" kind, he would watch in amazement as you effortlessly launched windows on multiple monitors run by multiple computers, copying and pasting madly between them. You know that as a scientist with world-domination plans, you cannot be slowed down by latency when launching applications and sharing data between them. That is why you installed Synergy in the first place, as outlined in the June 2016 issue of ODROID Magazine (`http://bit.ly/1XxSbRw`).

Recently, you decided to dust off a venerable ODROID-U2 in order to add to Synergy. However, you did not want to recompile Synergy from scratch. You updated the system with the following commands:

```
$ sudo apt update
$ sudo apt dist-upgrade
```

Then, you downloaded the version compiled for the ODROID-XU4 from `http://bit.ly/221NiL1`, but after decompressing and moving the files to the /usr/bin folder, you were greeted with:

```
synergyc: /usr/lib/arm-linux-gnueabihf/libstdc++.so.6: version `GLIB-
CXX_3.4.20' not found (required by synergyc)
```

After some searching around on the internet, you found an easy fix for this problem:

```
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$ sudo apt update
$ sudo apt install g++-4.9
```

After setting up the U2 using the "Configure Server..." button on Synergy Server running on your main machine and running the synergyc command on the U2, all that was needed was to "Stop" and "Start" the Synergy Server and your U2 is up and running. Life is good, and world domination is one step closer!

WHY ADD LIQUID COOLING TO AN ODROID-XU3 OR XU4?

by Michael Lee Wood (@ mlwood37)

It's well known throughout the ODROID community that the temperature of the XU4 and XU3 starts to rise sharply under heavy loads. This can quickly lead to something called thermal throttling, where the board automatically slows down after reaching a certain temperature to prevent damage to the board. The XU3 and XU4 tend to underclock from 2 GHz to 900 MHz in order to cool off for a little while. However, if we improve the ability for the ODROID to cool itself, then it can maintain its peak performance for a greater amount of time. This is where liquid cooling comes in. There are many ways to improve cooling, including a larger heatsink with a standard 40mm fan, but liquid cooling has two key advantages: it has a far greater thermal capacity, and is quieter, since most liquid cooling kits on the market use larger fans that are less noisy than the stock fan on an XU3 or XU4.

When considering how to liquid cool the ODROID, you need to consider how you are going to cool it and what hardware you are going to use. There are many different types of custom cooling components on the market, and if you're not careful, your cooling setup can cost a lot more than the board itself. Every water cooler will need three main components: a water block that fits your ODROID's chip in order to transfer heat away, a radiator and fan to cool the water, and a pump that lets water flow through the system. On my particular build, I opted for a Corsair Hydro Series H45 for many reasons, one being that the pump is built into the radiator itself. Many AIO (All in One) liquid cooling system have the pump on top of a CPU cooler, and this is useless to us, since we're not installing this on a standard PC. Having the pump on the radiator also allows us to keep the system as compact as possible, while making it easier to cut the CPU block off and connect our own liquid cooling block shaped to fit our ODROID.

No matter what, most CPU water cooling systems probably won't have a water block that supports your ODROID given their size, so you'll need to find one that will fit on a North Bridge chipset, which is similar in size to our ODROID SoC. For this build, I didn't want to spend any money, since I had an older EKWB Chipset water-cooling block lying around doing nothing. This wasn't the perfect solution, as it needed to be trimmed to be able to fit on the XU4 chipset. I chamfered the copper so that, when it was fitted, it didn't touch the HDMI or Sound Ports. The last thing we want is our XU4 shorting out! I also had to cut out a section of the block to get around the sound output port. There are some blocks on the market that will fit perfectly, however, they can cost in the region of £35.00 (USD$43) plus post and packing, and that probably won't include the rest of the cooling system.

To help keep things compact, I also mounted my ODROID to the radiator using some screws and custom laser cut plastic parts. You can get these ma-

Figure 1 - The final result of the liquid cooling project

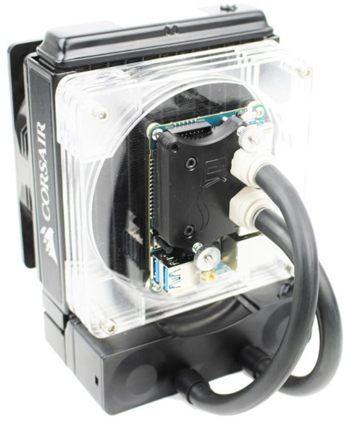
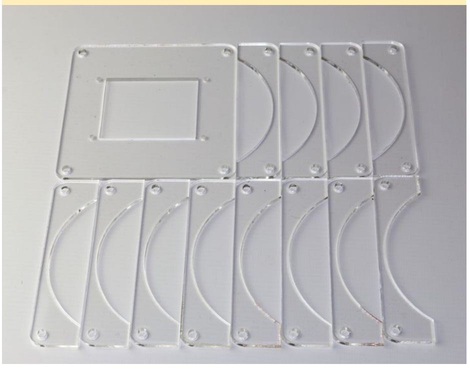


Figure 2 - The acrylic parts needed for mounting your ODROID-XU4



chined, as shown in Figure 2, by anyone with access to a decent laser cutter using plans I've made available at http://bit.ly/2fy3llS. My build also requires a customization with a drill and self-tapping screw to access the coolant and reduce the tubing size to fit my more compact size.

Tools and parts

Now, let me go into the nitty gritty of achieving this water cooled system, including the steps I took to cut down the water block to size. First, here's a list of the tools you'll need:

- Screw drivers.
- Dremel rotary tool with disk cutting tools.
- Access to a laser cutter or 3D printer
- Hack Saw
- Sand paper, a grinder, or anything you can use to grind down the copper block
- Drill with a bit that can cut and is the same size as your self-tapping screw.
- Stanley knife or Tube cutters
- Syringe for filling the AIO with fresh coolant

And here are the parts we'll be using:

- Hardkernel ODROID XU4 or XU3
- Corsair Hydro H45 (part Number CW-9060028-WW)
- Chipset water cooling block
- John Guest Straight Adaptor 3/8" NPTF Thread 1/4" Tube Connection
- 0.25 meters of 10mm Hard tube Linear Low Density Polyethylene
- 3mm acrylic clear or coloured (for machining your parts)
- 12v Power supply specifically for the pump and 120mm Fan.
- Screws / bolts to fit everything.
- Thermal paste (MX4 is recommended)
- Mayhems XT1 Coolant + DI water
- Self-tapping screw and rubber washer

Build process

First, you'll need to check the fit of the water block that you have selected for your ODROID. When test fitting, check to make sure you are not anywhere near the HDMI port or the sound ports. You may have to remove the top and cut and chamfer the copper block so it fits perfectly. Once this is done, put the water block and your ODROID to the side.

Next, take the John Guest Straight Adaptor 3/8" NPTF Thread 1/4" Tube Connection and cut down the threaded ends with your hacksaw so they screw flushly into your water block. If needed, use some PFTE tape around the fittings so they create a good seal. You want to make absolutely sure that your fittings are tight and have a strong seal to prevent leakage.

Next, take your 10mm tube Linear Low Density Polyethylene and cut it down to size, pushing it into the end of your John Guest fittings. Make sure they are tight and flush just like the fittings. This is where you will be pushing your AIO into the block.

After this has been completed, we're ready to attach our Corsair Hydro H45 to the water block. Cut off the tubes nearest to the water cooling block that comes attached to your AIO system. You should leave some extra tubing in case you want to use it again in the future. Take a container and collect or dump the coolant inside your Corsair cooler. It's up to you if you want to reuse this coolant, but it isn't a very high quality coolant and will likely be contaminated with flecks of aluminum too.

Next, we'll prepare our radiator to receive coolant, since it was designed as a closed loop system. Take your drill and slowly drill in the top (the top right hand side is the best place for this) of

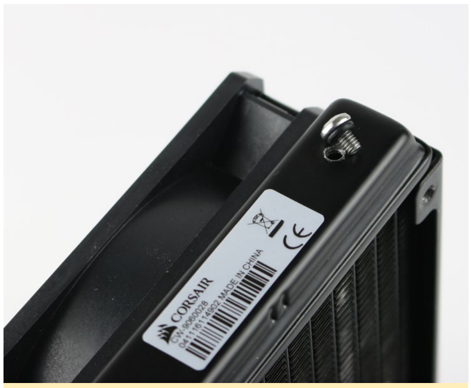


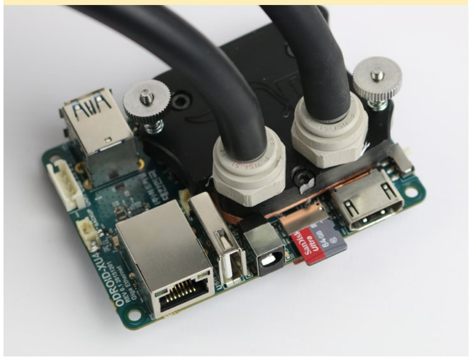
Figure 3 - The self-tapping screw and hole for filling your cooling block with coolant

the Corsair Hydro H45 radiator and, if possible, do this upside down so no metal flakes go into the radiator. Once you have done this, take a small self-tapping screw and add a rubber washer to it. Now screw your self-tapping screw in carefully, making sure that you don't overtighten it.

Using the plans that I mentioned above, laser cut the acrylic 3mm fins and mounting plate for the XU4 / XU3 for the mod. You can do them in any color you wish. Some of you may wish to do a 3D print or make them out of wood. Use this article as inspiration to mod it in whatever way you'd like.

Next, we'll fit the water block to the AIO. Look at how it will all fit and cut the AIO tubes down as much as possible. Once you have done this, remove the self-tapping screw and use a syringe to fill the radiator up with your Mayhems XT1 (a non-toxic coolant that won't corrode copper or aluminum). Take your time filling this, as you need to get as much air out of the system possible. Tilt

Figure 4 - A closer look at the water block assembly



MINI METRO
A PERFECT GAME TO WONDER ABOUT WAITING FOR YOUR METRO.

by Bruno Doiche

Taking the metro to go around to places isn't the thing that would keep your mind occupied, but there is a certain sense of wonder when you play this game while you wait for your metro ride to come and take you home at the end of a working day. If you have fond memories of playing Sim-City, this game will be your next best time-waster in the greatest style possible!

San Francisco
Brazil
São Paulo
Cairo
Auck
Manage the busy São Paulo network of a small number of teeming stations.
Best 82
Played 1 min
Play
Mode

For a game that had every reason to be absolutely banal, you will be hooked.

https://play.google.com/store/apps/details?id=nz.co.codepoint.minimetro

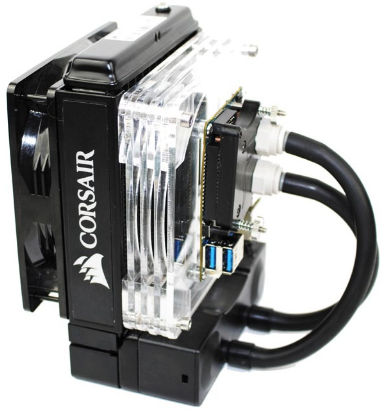
and move the AIO around while filling to aid in air removal. You can power up the pump after each fill to get trapped air to move to your fill port. However, do not turn on the pump if there is no coolant in it, as this will damage the pump.

Once the coolant is filled, screw back in the self-tapping screw with the rubber washer. Test the pump without fitting anything to it in case of any leaks. Do this for at least 1 to 2 hours before proceeding any further to ensure that everything is properly fitted.

You are now ready to fit the plastic fins and water block to the front of the AIO. Add thermal Paste onto your SoC. A pea-sized ball is more than enough. Take your time and remember when fitting the water block to not over tighten the bracing screws to the XU4 / XU3 as you may start to bend the PCB, which can ruin the ODROID. Then, simply layer the fins on the Corsair AIO and screw it all down. Don't forget to mount the XU4 on the larger base plate and you may wish to use the raiser washers that I have added into the plans (DXF format).

The ODROID doesn't have enough power for the water cooler, so I used a power brick that is rated at 12 volts and 5 amps, which is connected to a constant current/voltage 5-30v 5A buck regulator with 2 displays (http://ebay.eu/2gfpnuZ). We connected the pump and fan directly to the 12V power supply, which draws 12 volts and 0.45 amps, and then added the buck regulator into the circuit to drop the voltage to 5 volts and 4 amps, which we can use to power the ODROID-XU4. This gives us a way to power our system from a single outlet.

Last, but not least: Test it out! Use a CPU stress testing tool to see how things go. You can slowly adjust the screw heads on the water block to get optimum cooling. Again, do not overtighten them and potentially bend the PCB. If you wish, you can also add a brace to the back of the board to help stop any PCB bending issues. Good luck and enjoy your ODROID at maximum performance!



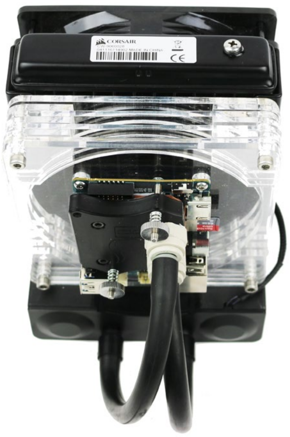
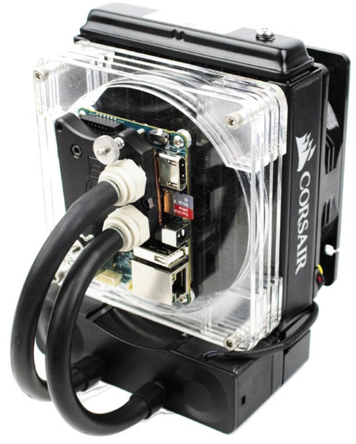


Figure 5 - Another angle of the final design



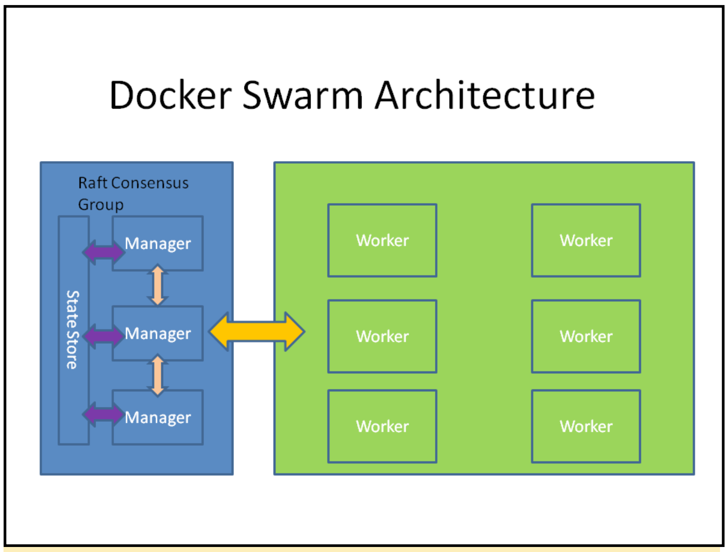
DOCKER 101

PART 2 - SWARM MODE

by Andy Yuen (@ MrDreamBot)

In Part 1 of this tutorial, we discussed the classic "docker run" and other commands. In Part 2, we are going to learn the swarm mode commands which are new to Docker version 1.12.X. So, what is swarm mode, and why do we need it?

All of the Docker commands that we discussed in Part 1 run Docker containers on the local machine. There is a limit to the number of containers that can be run on a single machine due to CPU and memory limitations. And when that machine fails, all your applications running on the machine will be unavailable. To provide high availability, scalability, orchestration and manageability, a cluster environment is needed. Swarm mode is the built-in cluster environment (the swarm) for Docker engines, although there are other orchestration engines for Docker (For instance, Kubernetes).



The Swarm Architecture

A swarm consists of multiple instances of Docker engines, called nodes. There are two types of nodes:

Manager Node, which dispatches units of work called tasks to worker nodes and performs orchestration, management functions, and maintenance of desired states of a swarm. There can be multiple manager nodes, but it is always an odd number of manager nodes due to the use of the Raft Consensus protocol (`https://raft.github.io/`). A Manager node can also be a worker nodes at the same time. The state store stores information such as state of cluster and the user defined configuration. Information is organized into objects such as clusters, nodes, services, tasks, and networks.

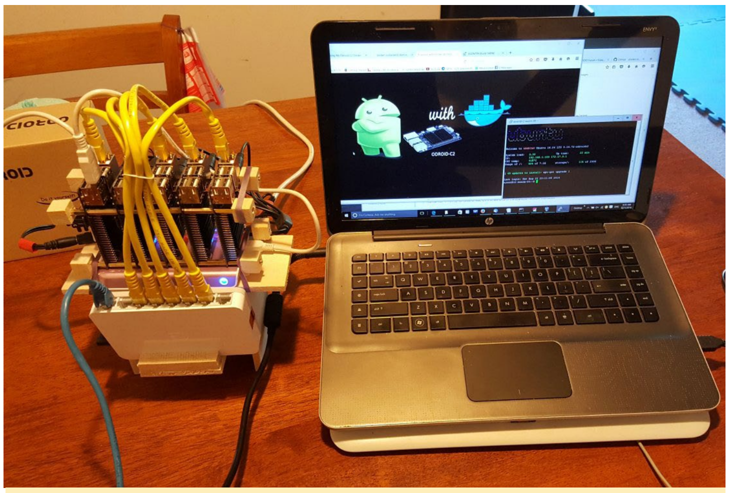
Worker Node, which executes tasks received from a manager node

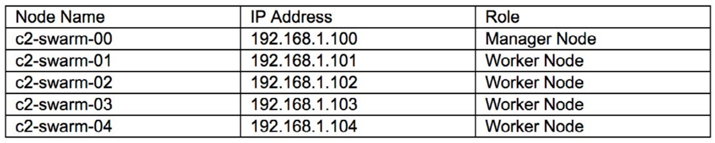
User interaction with the swarm is through services. A service is the definition of tasks to be run on worker nodes. Scaling is achieved using a replica services model in which a specific number of tasks is run on worker nodes to satisfy the desired state of the target number of replicas to be run. The swarm is administered using a command line interface via the manager node. In subsequent sections, we shall discuss how to create a swarm, services, and set the desired state. If you want to carry out the exercises in this tutorial, you will have to fulfill the prerequisites below.

Prerequisites

In order to follow this tutorial, you must have the following:

Two or more ODROID-C2 devices connected to the same Ethernet switch that your PC is connected to. Figure 2a shows my setup, which consists of 5 ODROID-C2s and my notebook all connected into a single network environment. My ODROID-C2 nodes and their roles are summarized in Figure 2b.


My setup of five ODROID-C2 devices



Node Name	IP Address	Role
c2-swarm-00	192.168.1.100	Manager Node
c2-swarm-01	192.168.1.101	Worker Node
c2-swarm-02	192.168.1.102	Worker Node
c2-swarm-03	192.168.1.103	Worker Node
c2-swarm-04	192.168.1.104	Worker Node

Chart of nodes and roles in the example swarm cluster

Your PC must be able to SSH into the c2-swarm-00 manager node where the swarm is administered.
Your nodes must be running Docker version 1.12.X. For those of you using an OS whose software repository does not provide the docker.io 1.12.x package or equivalent, you may install it following the instructions and binaries I put on Github at `http://bit.ly/2ejY1FE`**.**

Creating and Managing a Swarm

To create a swarm, issue the following command from the manager node:

```
$ docker swarm init \
 --advertise-addr 192.168.1.100
```

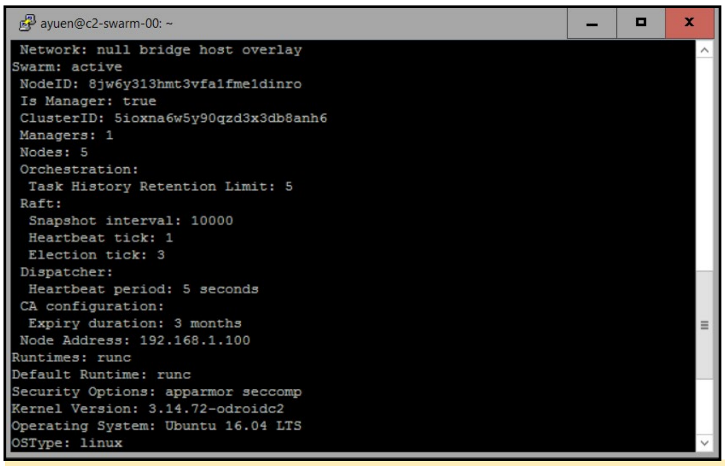
Which should return:

```
$ Swam initialized: current node (8jw6y313hmt3vfa1f-
me1dinro) is now a manager.
```

This should follow with a string including a token and open port on your manager node that can be used to invite other docker instances to the swarm. There are a number of options that you can use for your swarm. For help, you can type the following command:

```
$ docker swarm --help
```

This should show a host of information about your swarm, as shown next.



```
 Network: null bridge host overlay
Swarm: active
 NodeID: 8jw6y313hmt3vfa1fme1dinro
 Is Manager: true
 ClusterID: 5ioxna6w5y90qzd3x3db8anh6
 Managers: 1
 Nodes: 5
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot interval: 10000
  Heartbeat tick: 1
  Election tick: 3
 Dispatcher:
  Heartbeat period: 5 seconds
 CA configuration:
  Expiry duration: 3 months
 Node Address: 192.168.1.100
Runtimes: runc
Default Runtime: runc
Security Options: apparmor seccomp
Kernel Version: 3.14.72-odroidc2
Operating System: Ubuntu 16.04 LTS
OSType: linux
```

Swarm mode at a glance

To add workers to this swarm, run the following command on each of the remaining nodes:

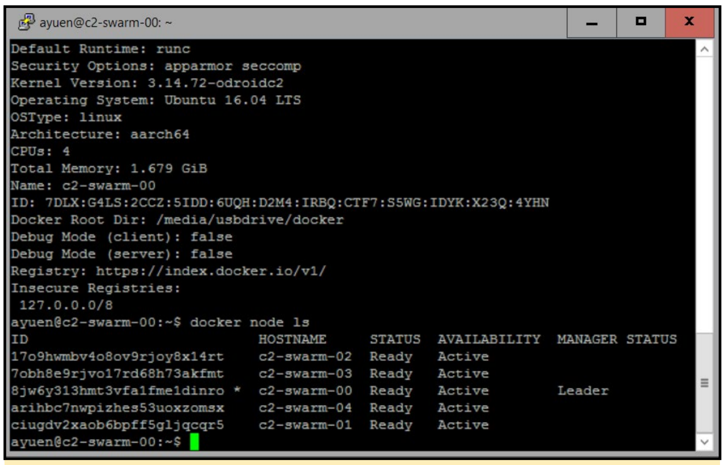
```
$ docker swarm join \
--token tokengoeshere \
192.168.1.100:2377
```

Your unique will be returned from the docker swarm init command we ran earlier. This is a security mechanism ensuring that only legitimate nodes can join the swarm. Note that you only have to do this once since the configuration is saved in the state store, which survives reboots.

We only need 1 manager node to make the swarm work. For this tutorial, we are not going to add additional manager nodes. To see the result, issue the following command from the manager:

```
$ docker node ls
```

Then we take a look at what that command will show in a typical list.



```
Default Runtime: runc
Security Options: apparmor seccomp
Kernel Version: 3.14.72-odroidc2
Operating System: Ubuntu 16.04 LTS
OSType: linux
Architecture: aarch64
CPUs: 4
Total Memory: 1.679 GiB
Name: c2-swarm-00
ID: 7DLX:G4LS:2CCZ:5IDD:6UQH:D2M4:IRBQ:CTF7:S5WG:IDYK:X23Q:4YHN
Docker Root Dir: /media/usbdrive/docker
Debug Mode (client): false
Debug Mode (server): false
Registry: https://index.docker.io/v1/
Insecure Registries:
 127.0.0.0/8
ayuen@c2-swarm-00:~$ docker node ls
ID                           HOSTNAME     STATUS  AVAILABILITY  MANAGER STATUS
17o9hwmbv4o8ov9rjoy8x14rt    c2-swarm-02  Ready   Active
7obh8e9rjvo17rd68h73akfmt    c2-swarm-03  Ready   Active
8jw6y313hmt3vfa1fme1dinro *  c2-swarm-00  Ready   Active        Leader
arihbc7nwpizhes53uoxzomsx    c2-swarm-04  Ready   Active
ciugdv2xaob6bpff5gljqcqr5    c2-swarm-01  Ready   Active
ayuen@c2-swarm-00:~$
```

A list of available docker nodes

If you lost the token earlier on from the init process, don't worry, you can retrieve it by issuing the following command from the Manager Node. To get the token for a manager, replace "worker" in the command by "manager":

```
$ docker swarm join-token worker
```

When you want a node to unjoin the swarm, issue the following command from that node:

```
$ docker swarm leave
```

By default, the manager node is also a worker node, if you do not want the manager to run any service tasks, you can "drain" the manager as follows:

```
$ docker node update --availability drain c2-swarm-00
```

In fact, you can use the drain command on any worker node. If the worker node is running service tasks when you issue the drain command, it will shut down those tasks and let the swarm start them on the other worker nodes to satisfy the desired state. When you want the node to run service tasks again, just issue the command below on the Manager Node:

```
$ docker node update --availability active c2-
swarm-00
```

Creating a service

Now that our swarm is up and running, we're going to create our first service to ping the Manager Node. We can see what this looks like in the next image. Type the following commands from the manager node:

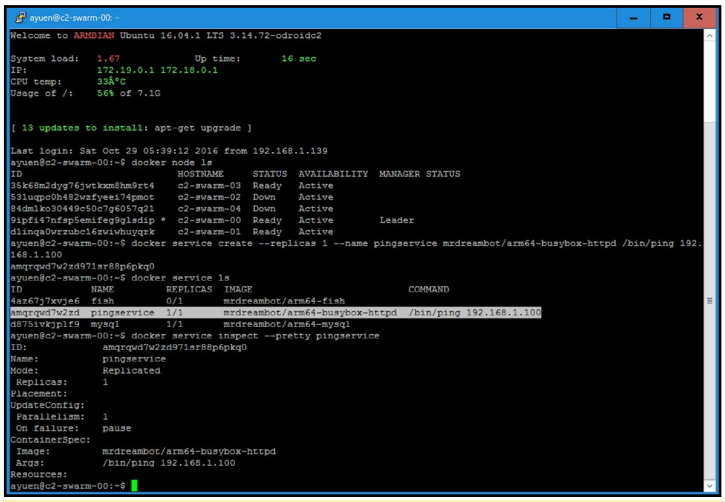
```
$ docker service create --replicas 1 --name ping-
service mrdreambot/arm64-busybox-httpd /bin /ping
192.168.1.100
$ docker service ls
$ docker service inspect --pretty pingservice
```

We can see that there is only 1 instance of the service running. Type the following command:

```
$ docker service ps pingservice
```

It tells us the service is running on c2-swarm-02. Start a terminal to c2-swarm-02 and type these commands:

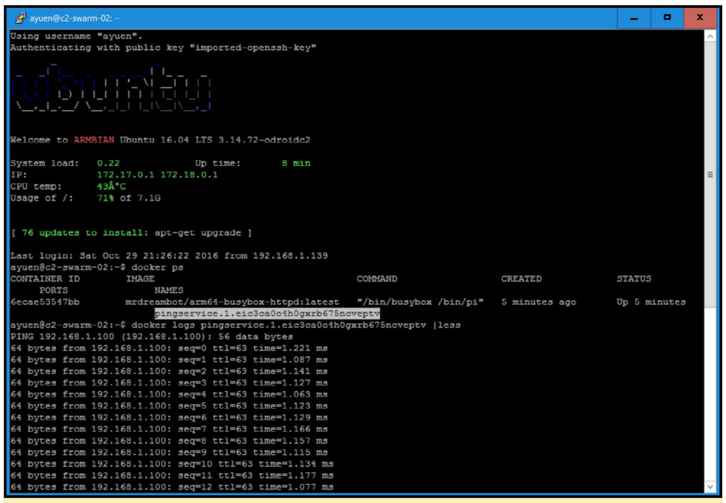
```
$ docker ps
$ docker logs pingservice.1.eic3ca0o4h0gxrb675ncveptv
```



```
Welcome to ARMBIAN Ubuntu 16.04.1 LTS 3.14.72-odroidc2

System load:   1.67              Up time:       16 sec
IP:            172.19.0.1 172.18.0.1
CPU temp:      33Â°C
Usage of /:    56% of 7.1G


[ 13 updates to install: apt-get upgrade ]

Last login: Sat Oct 29 05:39:12 2016 from 192.168.1.139
ayuen@c2-swarm-00:~$ docker node ls
ID                           HOSTNAME     STATUS  AVAILABILITY  MANAGER STATUS
35k68m2dyg76jwtkxm8hm9rt4    c2-swarm-03  Ready   Active
531uqpc0h482wzfyeei74pmot    c2-swarm-02  Down    Active
84dm1ko30449c50c7g6057q21    c2-swarm-04  Down    Active
9ipfi47nfsp5emifeg9glsdip *  c2-swarm-00  Ready   Active        Leader
d1inqa0wrzubc16zwiwhuyqrk    c2-swarm-01  Ready   Active
ayuen@c2-swarm-00:~$ docker service create --replicas 1 --name pingservice mrdreambot/arm64-busybox-httpd /bin/ping 192.
168.1.100
amqrqwd7w2zd971sr88p6pkq0
ayuen@c2-swarm-00:~$ docker service ls
ID            NAME         REPLICAS  IMAGE                           COMMAND
4az67j7xvje6  fish         0/1       mrdreambot/arm64-fish
amqrqwd7w2zd  pingservice  1/1       mrdreambot/arm64-busybox-httpd  /bin/ping 192.168.1.100
d875ivkjplf9  mysql        1/1       mrdreambot/arm64-mysql
ayuen@c2-swarm-00:~$ docker service inspect --pretty pingservice
ID:             amqrqwd7w2zd971sr88p6pkq0
Name:           pingservice
Mode:           Replicated
 Replicas:      1
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
ContainerSpec:
 Image:         mrdreambot/arm64-busybox-httpd
 Args:          /bin/ping 192.168.1.100
Resources:
ayuen@c2-swarm-00:~$
```

The service creation process

For reference, pingservice.1.eic3ca0o4h0gxrb675ncveptv is the docker container ID identified in the output of the docker ps command. The logs, as shown in below, show that pingservice, indeed, is pinging my swarm manager 192.168.1.100.



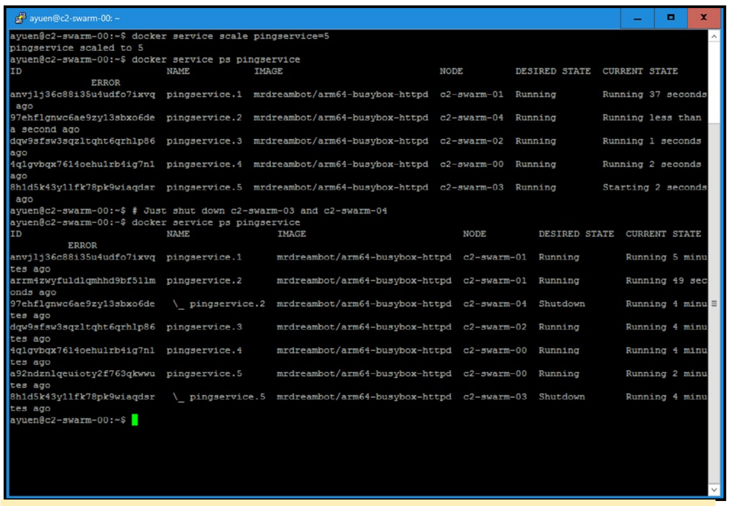
```
Using username "ayuen".
Authenticating with public key "imported-openssh-key"


Welcome to ARMBIAN Ubuntu 16.04 LTS 3.14.72-odroidc2

System load:   0.22              Up time:       8 min
IP:            172.17.0.1 172.18.0.1
CPU temp:      43Â°C
Usage of /:    71% of 7.1G


[ 76 updates to install: apt-get upgrade ]

Last login: Sat Oct 29 21:26:22 2016 from 192.168.1.139
ayuen@c2-swarm-02:~$ docker ps
CONTAINER ID        IMAGE                                  COMMAND                  CREATED             STATUS
     PORTS               NAMES
6ecae53547bb        mrdreambot/arm64-busybox-httpd:latest   "/bin/busybox /bin/pi"   5 minutes ago       Up 5 minutes
                         pingservice.1.eic3ca0o4h0gxrb675ncveptv
ayuen@c2-swarm-02:~$ docker logs pingservice.1.eic3ca0o4h0gxrb675ncveptv |less
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: seq=0 ttl=63 time=1.221 ms
64 bytes from 192.168.1.100: seq=1 ttl=63 time=1.087 ms
64 bytes from 192.168.1.100: seq=2 ttl=63 time=1.141 ms
64 bytes from 192.168.1.100: seq=3 ttl=63 time=1.127 ms
64 bytes from 192.168.1.100: seq=4 ttl=63 time=1.063 ms
64 bytes from 192.168.1.100: seq=5 ttl=63 time=1.123 ms
64 bytes from 192.168.1.100: seq=6 ttl=63 time=1.129 ms
64 bytes from 192.168.1.100: seq=7 ttl=63 time=1.166 ms
64 bytes from 192.168.1.100: seq=8 ttl=63 time=1.157 ms
64 bytes from 192.168.1.100: seq=9 ttl=63 time=1.115 ms
64 bytes from 192.168.1.100: seq=10 ttl=63 time=1.134 ms
64 bytes from 192.168.1.100: seq=11 ttl=63 time=1.177 ms
64 bytes from 192.168.1.100: seq=12 ttl=63 time=1.077 ms
```

The ping log at a glance

To scale it to run 5 instances, issue the command from the swarm manager:

```
$ docker service scale pingservice=5
```

Since I used the service scale command to set the desired state to run 5 instances of the service pingservice, it spins up new containers to make up the 5 service instances. Note that the swarm performs load balancing by spreading the load across the nodes in the swarm. When I shut down nodes c2-swarm-03 and c2-swarm-04, I expect the swarm to spin up services on the remaining nodes to maintain the 5 replica count. In fact that is the case as can be seen in Figure 7. The swarm is running 2 containers on c2-swarm-00 and c2-swarm-01, and 1 on c2-swarm-02.



```
ayuen@c2-swarm-00:~$ docker service scale pingservice=5
pingservice scaled to 5
ayuen@c2-swarm-00:~$ docker service ps pingservice
ID                         NAME           IMAGE                              NODE         DESIRED STATE  CURRENT STATE
              ERROR
anvjlj36c88i35u4udfo7ixvq  pingservice.1  mrdreambot/arm64-busybox-httpd  c2-swarm-01  Running        Running 37 seconds
 ago
97ehflgnwc6ae9zy13sbxo6de  pingservice.2  mrdreambot/arm64-busybox-httpd  c2-swarm-04  Running        Running less than
a second ago
dqw9sfsw3sqzltqht6qrhlp86  pingservice.3  mrdreambot/arm64-busybox-httpd  c2-swarm-02  Running        Running 1 seconds
ago
4q1gvbqx7614oehu1rb4ig7n1  pingservice.4  mrdreambot/arm64-busybox-httpd  c2-swarm-00  Running        Running 2 seconds
ago
8h1d5k43y1lfk78pk9wiaqdsr  pingservice.5  mrdreambot/arm64-busybox-httpd  c2-swarm-03  Running        Starting 2 seconds
 ago
ayuen@c2-swarm-00:~$ # Just shut down c2-swarm-03 and c2-swarm-04
ayuen@c2-swarm-00:~$ docker service ps pingservice
ID                         NAME              IMAGE                              NODE         DESIRED STATE  CURRENT STATE
          ERROR
anvjlj36c88i35u4udfo7ixvq  pingservice.1     mrdreambot/arm64-busybox-httpd  c2-swarm-01  Running        Running 5 minu
tes ago
arrm4zwyfuldlqmhhd9bf5llm  pingservice.2     mrdreambot/arm64-busybox-httpd  c2-swarm-01  Running        Running 49 sec
onds ago
97ehflgnwc6ae9zy13sbxo6de   \_ pingservice.2  mrdreambot/arm64-busybox-httpd  c2-swarm-04  Shutdown       Running 4 minu
tes ago
dqw9sfsw3sqzltqht6qrhlp86  pingservice.3     mrdreambot/arm64-busybox-httpd  c2-swarm-02  Running        Running 4 minu
tes ago
4q1gvbqx7614oehu1rb4ig7n1  pingservice.4     mrdreambot/arm64-busybox-httpd  c2-swarm-00  Running        Running 4 minu
tes ago
a92ndznlqeuioty2f763qkwwu  pingservice.5     mrdreambot/arm64-busybox-httpd  c2-swarm-00  Running        Running 2 minu
tes ago
8h1d5k43y1lfk78pk9wiaqdsr   \_ pingservice.5  mrdreambot/arm64-busybox-httpd  c2-swarm-03  Shutdown       Running 4 minu
tes ago
ayuen@c2-swarm-00:~$
```

The desired states performing load balancing

When you are done with the service, remove it by issuing the following command:

```
$ docker service rm pingservice
```

Testing the Routing Mesh

In Part 1, I deployed both MySQL and Fish on C2-swarm-00 using Docker run, which was in lieu of Swarm Mode. In this section, I am going to deploy them as services in Swarm Mode instead

Please note that only 1 instance of MySQL mapped to the database directory on disk can be running at a time. Did you remember from part 1 that only c2-swarm-00 has a hard disk? I am going to run MySQL as a service with only 1 instance and set up a constraint that the service can only be run on the host c2-swarm-00 using the following command:

```
$ docker service create  \
--name mysql \
-p 3306:3306 \
-e MYSQL_USER=fishuser \
-e MYSQL_PASSWORD=fish456 \
-e MYSQL_DATABASE=fish \
--constraint 'node.hostname == c2-swarm-00' \
--mount type=bind,src=/media/sata/fish-mysql,dst=/u01/
my3306/data \
mrdreambot/arm64-mysql
```

This is very similar to running MySQL using the classic docker run command. The main difference is that if the container running the MySQL service should fall over, the swarm will automatically start another instance to replace it.

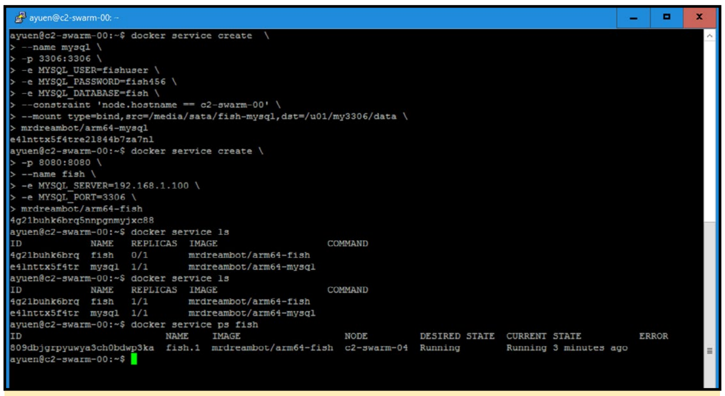
Also note the use of "--constraint 'node.hostname == c2-swarm-00'", which constrains MySQL to run on the node named c2-swarm-00 only. There are other pre-defined constraints as can be found in the docker documentation. To run fish as a service, issue the command:

```
$ docker service create \
-p 8080:8080 \
--name fish \
-e MYSQL_SERVER=192.168.1.100 \
-e MYSQL_PORT=3306 \
mrdreambot/arm64-fish
```

You can see that the services are running using the following command:

```
$ docker service ls
```

Above there is a look at how this appears when you're successful. Note that the first time I issue the "docker service ls" command, MySQL was still starting. It was started when I issued the second 'docker service ls' command. Fish is running on c2-swarm-04.



```
ayuen@c2-swarm-00:~$ docker service create  \
> --name mysql \
> -p 3306:3306 \
> -e MYSQL_USER=fishuser \
> -e MYSQL_PASSWORD=fish456 \
> -e MYSQL_DATABASE=fish \
> --constraint 'node.hostname == c2-swarm-00' \
> --mount type=bind,src=/media/sata/fish-mysql,dst=/u01/my3306/data \
> mrdreambot/arm64-mysql
e4lnttx5f4tre21844b7za7nl
ayuen@c2-swarm-00:~$ docker service create \
> -p 8080:8080 \
> --name fish \
> -e MYSQL_SERVER=192.168.1.100 \
> -e MYSQL_PORT=3306 \
> mrdreambot/arm64-fish
4g21buhk6brq5nnpgnmyjxc88
ayuen@c2-swarm-00:~$ docker service ls
ID             NAME   REPLICAS  IMAGE                       COMMAND
4g21buhk6brq  fish   0/1       mrdreambot/arm64-fish
e4lnttx5f4tr  mysql  1/1       mrdreambot/arm64-mysql
ayuen@c2-swarm-00:~$ docker service ls
ID             NAME   REPLICAS  IMAGE                       COMMAND
4g21buhk6brq  fish   1/1       mrdreambot/arm64-fish
e4lnttx5f4tr  mysql  1/1       mrdreambot/arm64-mysql
ayuen@c2-swarm-00:~$ docker service ps fish
ID                          NAME    IMAGE                  NODE         DESIRED STATE  CURRENT STATE          ERROR
809dbjgrpyuwya3ch0bdwp3ka  fish.1  mrdreambot/arm64-fish  c2-swarm-04  Running        Running 3 minutes ago
ayuen@c2-swarm-00:~$
```

Load balancing in effect with a Docker Swarm

All nodes in a swarm are on an ingress routing mesh, meaning that all nodes in the swarm can accept connections on published ports for any service (in our case, port 8080 for Fish and port 3306 for MySQL) even if the task is not running on that node. This implies that we can point our browser to any of the nodes in the swarm to access the fish application. However, this was not the behavior that I witnessed on my swarm cluster. I could only access the service on the nodes running it (c2-swarm-04) and not on any other nodes. After I scaled up and down the number of replicas of the service several times, occasionally I could not even invoke the service on the node running the service! I searched the Internet and many people reported the same issue. The problem was thought to be swarm mode not updating the IPVS (IP Virtual Server) tables correctly. IPVS is the kernel module responsible for load balancing. Automatic load balancing is a great feature once it is working.

User-defined networks

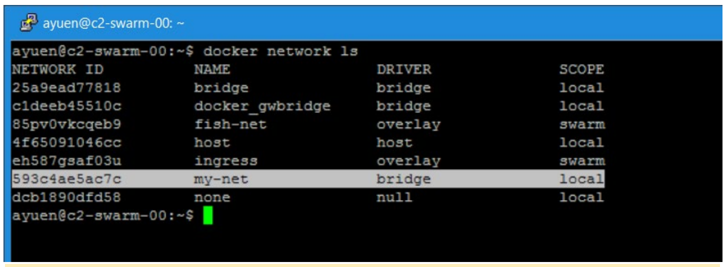
User-defined networks can be used to isolate containers. For example, if you created a user-defined network and assign containers or services to it, other containers or services not in that network will not be able to access them, and vice-versa. In addition, the containers or services in a user defined network can reference each other by name, such as the name assigned to a container upon creation. There are two types of user-defined networks: bridge networks and overlay networks.

Bridge Networks

This is mainly used by the classic docker run commands. A bridge network can only function on the local machine. For example, we can create a bridge network and add the MySQL and Fish containers to it:

```
$ docker network create --driver bridge my-net
$ docker run -d --network my-net \
--name mysql \
-e MYSQL_USER=fishuser \
-e MYSQL_PASSWORD=fish456 \
-e MYSQL_DATABASE=fish \
-v /media/sata/fish-mysql:/u01/my3306/data \
mrdreambot/arm64-mysql
$ docker run -d --network my-net \
-p 8080:8080 \
--name fish \
-e MYSQL_SERVER=mysql \
-e MYSQL_PORT=3306 \
mrdreambot/arm64-fish
```

After creating the network, one can see a bridge network (my-net) has been set up, as shown in Figure 9.



```
ayuen@c2-swarm-00:~$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
25a9ead77818        bridge              bridge              local
c1deeb45510c        docker_gwbridge     bridge              local
85pv0vkcqeb9        fish-net            overlay             swarm
4f65091046cc        host                host                local
eh587gsaf03u        ingress             overlay             swarm
593c4ae5ac7c        my-net              bridge              local
dcb1890dfd58        none                null                local
ayuen@c2-swarm-00:~$
```

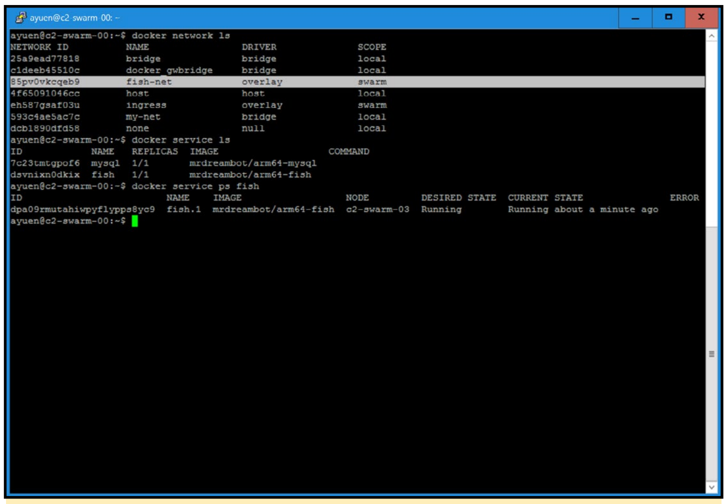
A look at the bridged network

Note that the docker run command for Fish references the MYSQL_SERVER by its name "mysql" instead of using localhost or an IP address, and we don't even need to publish the MySQL port (i.e., there is no "-p 3306:3306" in the MySQL docker run command). Although we cannot access MySQL from the docker host without publishing port 3306, Fish can access MySQL port 3306 because it is on the same user defined network called my-net. The bridge network is working as intended.

Overlay Network

To create a user-defined network spanning multiple nodes in swarm mode, we have to create an overlay network. The swarm mode equivalent of deploying the MySQL and Fish above in an overlay network is:

```
$ docker network create --driver overlay --sub-
net=172.20.0.0/16 fish-net
$ docker service create --network fish-net \
--name mysql \
-e MYSQL_USER=fishuser \
-e MYSQL_PASSWORD=fish456 \
-e MYSQL_DATABASE=fish \
--constraint 'node.hostname == c2-swarm-00' \
--mount type=bind,src=/media/sata/fish-mysql,\
dst=/u01/my3306/data \
mrdreambot/arm64-mysql
$ docker service create --network fish-net \
-p 8080:8080 \
--name fish \
-e MYSQL_SERVER=mysql \
-e MYSQL_PORT=3306 \
mrdreambot/arm64-fish
```



```
ayuen@c2-swarm-00:~$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
25a9ead77818        bridge              bridge              local
c1deeb45510c        docker_gwbridge     bridge              local
85pv0vkcqeb9        fish-net            overlay             swarm
4f65091046cc        host                host                local
eh587gsaf03u        ingress             overlay             swarm
593c4ae5ac7c        my-net              bridge              local
dcb1890dfd58        none                null                local
ayuen@c2-swarm-00:~$ docker service ls
ID            NAME   REPLICAS  IMAGE                   COMMAND
7c23tmtgpof6  mysql  1/1       mrdreambot/arm64-mysql
dsvnixn0dkix  fish   1/1       mrdreambot/arm64-fish
ayuen@c2-swarm-00:~$ docker service ps fish
ID                          NAME    IMAGE                  NODE         DESIRED STATE  CURRENT STATE                ERROR
dpa09rmutahiwpyflypps8yc9  fish.1  mrdreambot/arm64-fish  c2-swarm-03  Running        Running about a minute ago
ayuen@c2-swarm-00:~$
```

The overlay network configuration

After creating the network, one can see an overlay network (fish-net) has been set up for the swarm, as shown in Figure 10.

Unfortunately, Fish did not work. It showed the login screen, but was not able to authenticate after I entered the username and password repeatedly, and was unable to communicate to MySQL via its name. There is an overlay networking problem in the version of Docker that I use, where the implementation is not able to resolve the container or service by name.

Conclusion

Although there are some issues with the swarm mode in the version of docker engine I used in this tutorial, it should still give you some insight on how it might have worked and how awesome these features will be when working. Swarm mode is the native cluster management and service orchestration features embedded in the Docker engine since version 1.12.0. Before swarm mode comes along, creating a swarm involved using third-party tools such as consul or etcd to provide a distributed state store for service discovery. With swarm mode, everything is built-in and works out-of-the-box, except for the issues mentioned above: routing mesh network and user defined overlay network. Although this tutorial is designed to run on the ODROID-C2 swarm, all the commands that you learned are exactly the same on INTEL-based machines running the Docker engine. You can easily apply your Docker command line knowledge to different environments including Linux, MacOS, Windows and on a cloud host.

DESIGNING YOUR OWN ODROID SEEDBOX

HARNESS THE POWER OF THE CLOUDSHELL

by Joshua Sherman

Seedboxes are far from new, but I've yet to find one that will do everything that I want with my ODROID-XU4, especially when integrated with the CloudShell case and LCD interface. This guide will show you how to build a seedbox that is:

Powered by the ODROID-XU4
Uses an SSD with SATA connectivity for high speed transfers and read/write speeds
Runs the operating system and cache on the hard drive to reduce fatigue on the SD card
Supports web-access to Transmission for remote downloading
Automatically connects to a VPN upon boot for maximum security

So while nothing here is original, here's a sure-fire way to begin with a ODROID-XU4 and a CloudShell kit, and end up with a working seedbox with all of these great features.

Background

To make things easier for my dad, I built him a seedbox network attached storage (NAS) device last year for all of his downloading and storage needs. It used an old Raspberry Pi 2 I had lying around before I found out about the far more powerful (and USB 3.0 capable) ODROID-XU4. I used that RPi 2 to build him a seedbox that was a good start, but it used an attached USB hard drive that just wasn't very fast.

I also realized how convenient and well packaged the ODROID CloudShell is, so I decided to put two great things together and write a guide around turning a Cloudshell into a seedbox with VPN capabilities. There are other guides out there, often geared toward the RPi or Virtual Private Server (VPS), but I really prefer turn-key guides that don't require sifting through three different sets of instructions that don't address the nuances of the ODROID, ARM architecture, or the specific tools I want to use to achieve my ideal build. I intended it to be a newer, faster, and sleeker upgrade to his Pi2.

Some credit is due to PiMylife and MakeUseOf, who have some very useful Raspberry Pi guides that I've adapted for the XU4 and CloudShell. I also took some snippets and instructions from several ODROID community members on how to properly configure this guide for the XU4 and CloudShell. All in all, this should run for less than $200, and offer anything you'd want from a home seedbox or single drive NAS.

Parts list

ODROID XU4
CloudShell XU4 kit (featuring a USB 3.0 to SATA adapter and 2.2-inch TFT LCD Display)
RTC Battery
256GB SSD (or other 2.5-inch storage device of your choice)
32GB micro-SD card (Or other OS drive of your choice)
VPN subscription with OpenVPN support

The parts and tools we'll need to get the job done

VPN

This guide does include a step for setting up traffic encryption through a Virtual Private Network (VPN). I highly recommend you use a VPN with a seedbox or NAS you're using for downloading content for security and peace-of-mind. You sacrifice your peak speed capabilities if you have a fiber optic or other high speed Internet connection, but many VPN providers can offer fairly good speeds despite routing all traffic through a third party server. Plus, you can still locally access those files, maximizing connectivity and the value of the USB 3.0 to SATA connectivity.

Local or online?

This guide is for a local seedbox setup. Can you do it online? Sure, but you open yourself up to a number of security risks. This guide involves plain text passwords (for the sake of simplicity and a local-only implementation) to make it easier to configure the VPN connection on each launch. You may not need this depending on if you want a VPN, if your VPN service provider doesn't need a username and password. There also may be a more secure way to implement this, so feel free to make suggestions in the thread for anyone else considering such a build. I chose this method, since it's based on a previous method that I knew worked, and meets my personal needs in having a local NAS with 100 percent encrypted traffic.

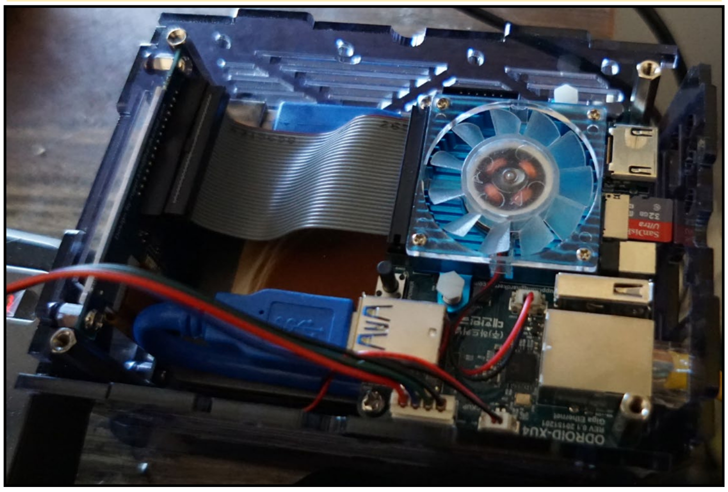
Getting Started

First, we will update our ODROID, since this guide starts with running an out-of-the-box 16.04 Ubuntu MATE image:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
```

The update will take a while to complete. We could use a more minimal image, but this guide is geared toward simplicity. Go get some coffee and come back when it's done! Figure 2 has a closer look at the hardware to look at in the meantime.

Closeup of seedbox hardware



Next, we want to configure our seedbox's CloudShell display to make sure it works. We start with smartmontools:

```
$ sudo apt install smartmontools
```

Next, we need to follow ODROID's instructions to configure the frame buffer for our TFT display in the CloudShell:

```
$ sudo -s
$ echo "options fbtft_device name=hktft9340 busnum=1
rotate=270" > /etc/modprobe.d/odroid-cloudshell.conf
$ echo "spi_s3c64xx" >> /etc/modules
$ echo "fbtft_device" >> /etc/modules
```

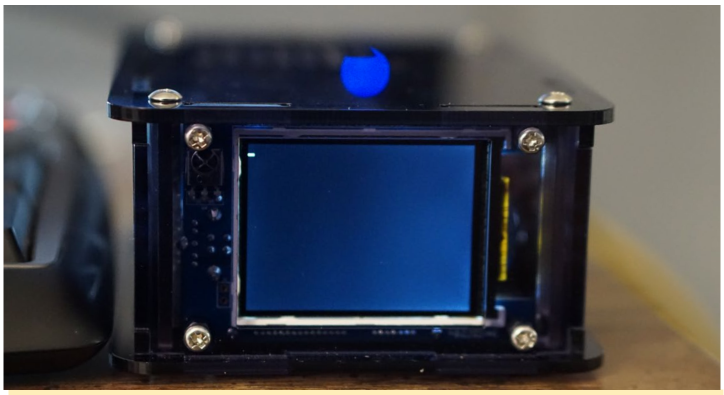
Then, we'll remove the blacklist on the Serial Peripheral Interface (SPI) through which our ODROID-XU4 and CloudShell are connected. We do this by opening blacklist-odroid.conf:

```
$ sudo nano /etc/modprobe.d/blacklist-odroid.conf
```

When you're finished, it should look like this. Note the two SPI rows and LCD row are now commented:

```
# Comment the required lines

# IO Board
blacklist ioboard_bh1780
blacklist ioboard_bmp180
blacklist ioboard_keyled

# SPI
# blacklist spidev
# blacklist spi_s3c64xx

# 3.2" LCD Touchscreen driver
blacklist ads7846
```

Reboot the XU4, but make sure you don't have any HDMI cables connected to it to ensure it connects to the LCD display.



We have power! But the display still isn't working yet

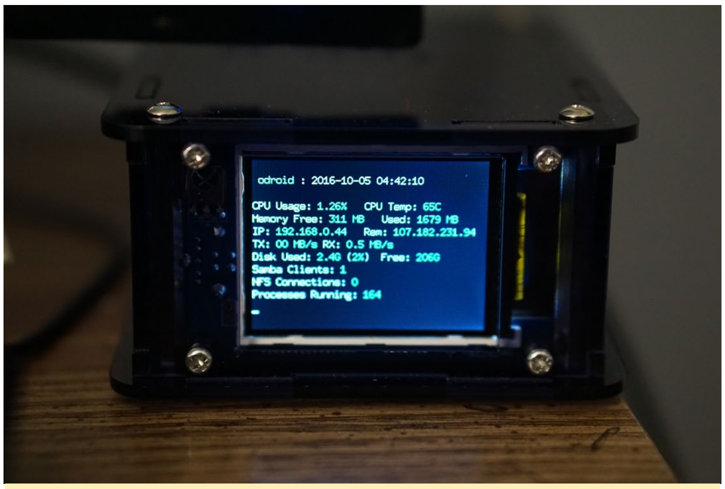
At this point, we have a working device, but the display is blank. We want it to show all sorts of useful info about our

CloudShell, so let's use a useful script to do this for us. Eventually we'll be using /dev/sda2 as our storage drive, so let's modify @mdrjr's application and install it:

```
$ sudo apt-get install curl sysstat
$ wget https://github.com/jsherm101/cloudshell_lcd/\
raw/master/cloudshell-lcd_20160913-3-fixed.deb
$ sudo dpkg -i cloudshell-lcd_20160913-3-fixed.deb
```

If we reboot again, it should now show some useful information about the device:

CPU usage and temperature
RAM availability
The local IP address of the device
Transfer rates
Disk usage, which is empty right now because we haven't configured our /dev/sda2 drive



```
odroid : 2016-10-05 04:42:10

CPU Usage: 1.26%   CPU Temp: 65C
Memory Free: 311 MB   Used: 1679 MB
IP: 192.168.0.44   Rem: 107.182.231.94
TX: 00 MB/s RX: 0.5 MB/s
Disk Used: 2.4G (2%)  Free: 206G
Samba Clients: 1
NFS Connections: 0
Processes Running: 164
_
```

Things are looking much better

Operating system configuration

Next comes the main task of this project, which is to set up a BitTorrent protocol application to work well with the SSD. This is nothing more than an apt-get command if we wanted to use our SD card where the OS is installed for storage, but this poses two issues:

There are large SD cards (>256GB), but they're more expensive or slower than their SSD and HDD counterparts
An SD card has a much shorter read/write lifecycle than an SSD or HDD, and will fail sooner

To solve this, we're going to actually move our ODROID installation to the SSD and boot from the SSD for both our operating system and our additional NAS storage folder. All credit goes to James @ MakeUseOf for this great idea. This might not be a concern for most practical uses, but is done as a precaution in this build given the high number of reads and writes anticipated in regular use.

Once we complete this, we'll be installing Transmission as our BitTorrent software of choice. It's just as popular as Deluge and other clients, and comes down to a matter of preference in this case. Both Transmission and Deluge support web-based GUIs and server-client configurations for remote access.

Next, let's configure our storage drive. We've already plugged it into our assembled CloudShell and the necessary SATA-to-USB drivers are configured with smartmontools, so it should appear in fdisk as the first drive:

```
$ sudo fdisk /dev/sda1
```

My drive is currently an NTFS drive from an old Windows installation. We'll want to press "d" in fdisk to delete the partition, then "n" to create a new one, then "p" for primary partition. We're ultimately creating two partitions: One for the OS we're moving to the SSD, and one for the storage area we'll share over the local network.

Create the first partition at the first sector available (probably 2048) and then type "+16G" to create a 16GB partition in size. Then start again typing "n" and "p" to create a second partition with the remaining storage on the drive. My 256GB SSD left about 208 GB after accounting for capacity formatting and the installation partition. Once we're finished, type "w" to write the new partitions to the SSD.

Next, convert both partitions into the ext4 format and mount them to the two folders we've set aside for this project:

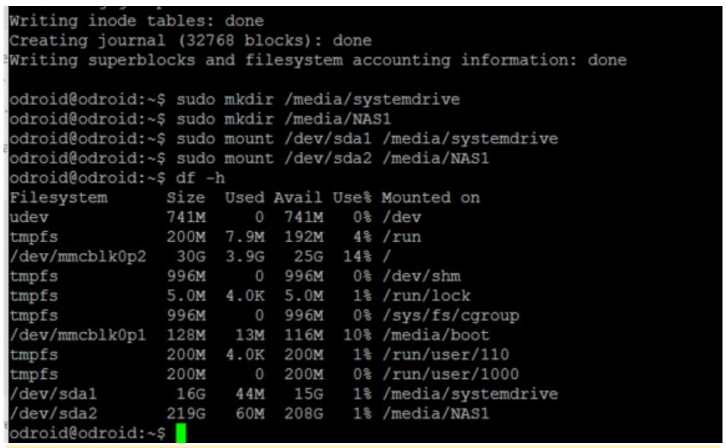
```
$ sudo mkfs.ext /dev/sda1
$ sudo mkfs.ext /dev/sda2
$ sudo mkdir /media/systemdrive
$ sudo mkdir /media/NAS1
$ sudo mount /dev/sda1 /media/systemdrive
$ sudo mount /dev/sda2 /media/NAS1
$df -h (to see a list of drives and confirm everything
is in order)
```

Once you've finished this, your drive configuration should look similar the picture next.

Now we can use rsync to move over our data on the SD card to the SSD. Before we do this, we should make a backup of the boot.ini file we'll end up editing, which is in the FAT boot partition of the SD card:

```
$ sudo cp /media/boot/boot.ini /media/boot/boot.ini.bak
```

Now we need to find the UUID of the unique partition we



```
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

odroid@odroid:~$ sudo mkdir /media/systemdrive
odroid@odroid:~$ sudo mkdir /media/NAS1
odroid@odroid:~$ sudo mount /dev/sda1 /media/systemdrive
odroid@odroid:~$ sudo mount /dev/sda2 /media/NAS1
odroid@odroid:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            741M     0  741M   0% /dev
tmpfs           200M  7.9M  192M   4% /run
/dev/mmcblk0p2   30G  3.9G   25G  14% /
tmpfs           996M     0  996M   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           996M     0  996M   0% /sys/fs/cgroup
/dev/mmcblk0p1  128M   13M  116M  10% /media/boot
tmpfs           200M  4.0K  200M   1% /run/user/110
tmpfs           200M     0  200M   0% /run/user/1000
/dev/sda1        16G   44M   15G   1% /media/systemdrive
/dev/sda2       219G   60M  208G   1% /media/NAS1
odroid@odroid:~$
```

Our partitions properly configured and mounted

made earlier.

```
$ lsblk -f
```

My disk UUID is 7d62ae29-a3cf-41d0-9127-065cf-c08fbe6, which is used as an example. Next, open boot.ini and search for "Basic Ubuntu Setup". You can search in Nano with CTRL+W:

```
$ sudo nano /media/boot/boot.ini
```

Comment out the line that's currently below that line. This is our original configuration for booting to the SD card. We can always revert it if anything goes wrong by plugging the SD card into any device (PC or *nix) and re-editing the boot.ini file. Next, we'll add our own new instructions:

```
## Boot from USB device
setenv bootrootfs "console=tty1 consoleblank=0
root=UUID=7d62ae29-a3cf-41d0-9127-065cfc08fbe6 root-
wait rootdelay=10 ro fsck.repair=yes"
```

Just make sure the UUID is whatever UUID comes up for your 16GB system partition under /dev/sda1. Before we use rsync to transfer over to the new partition, we need to edit fstab to mount our new drives on startup:

```
$ sudo nano /etc/fstab
```

Comment out the first line and now add these two lines below, assuming you've used the same partition names and folders as me:

```
$ /dev/sda1 / ext4 defaults,noatime 0 1
$ /dev/sda2 /media/NAS1 ext4 defaults 0 2
```

Finally, we can move over our OS from the SD card to the new partition we've set aside for it. Keep in mind that once you've done this, any changes to the OS won't appear after you restart the computer, as you'll be working from the new partition. It's best to reboot once you finish the following rsync command:

```
$ sudo apt-get install rsync
$ sudo rsync -axv / /media/systemdrive
```

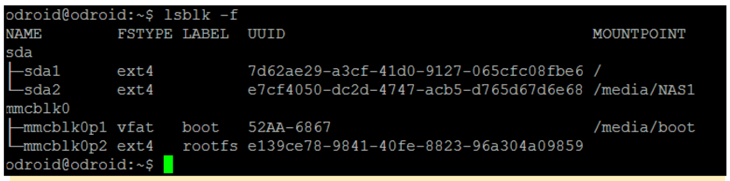
You'll probably have some time to grab some coffee while this transfers. To elaborate on what's happening, we're copying over our entire OS to /media/systemdrive, where we've mounted our /dev/sda1 partition. Keep in mind that after we restart, our fstab + boot.ini will remount and redirect the /dev/sda1 partition to "/" and serve as our operating system. We'll still be using the boot partition of your SD card, so don't remove it. Once it finishes, it's time to reboot and hold our breath:

```
$ sudo reboot
```

If we successfully reboot, then we're definitely working from one of our two partitions, either on the SD card or SSD. You can confirm we're using the SSD by checking our partitions again:

```
$ lsblk -f
```

You should see /sda1 mounted at "/" to confirm our success. You can see this yourself in Figure 6.



```
odroid@odroid:~$ lsblk -f
NAME         FSTYPE LABEL  UUID                                 MOUNTPOINT
sda
├─sda1       ext4          7d62ae29-a3cf-41d0-9127-065cfc08fbe6 /
└─sda2       ext4          e7cf4050-dc2d-4747-acb5-d765d67d6e68 /media/NAS1
mmcblk0
├─mmcblk0p1 vfat   boot    52AA-6867                            /media/boot
└─mmcblk0p2 ext4   rootfs  e139ce78-9841-40fe-8823-96a304a09859
odroid@odroid:~$
```

The SSD is mounted as the root file system

Transmission configuration

Okay, so we have our operating system transferred, and our display is showing our fancy 256GB SSD in working condition.

Now we need to install Transmission and setup our transmission-daemon. We'll be using our "odroid" default user, as well as be configuring our drive to have an incomplete and complete folder as well. Let's start by installing Transmission and adding our new incomplete and complete folders:

```
$ sudo apt-get install transmission-daemon
$ sudo mkdir -p /media/NAS1/incomplete
$ sudo mkdir -p /media/NAS1/complete
```

Our Seedbox is starting to look pretty sharp!

Now we need to configure Transmission by turning off its services and opening the settings file:

```
$ sudo service transmission-daemon stop
$ sudo nano /etc/transmission-daemon/settings.json
```

You'll want to configure several settings:

Set your incomplete to "true" and "media/NAS1/incomplete"
Set your complete to "media/NAS1/complete"
Set rpc-authentication-required to "false"
Set the whitelist to 192.168.*.* to ensure you can access it remotely from another device on your network

Each of the rows we edited should look like this, scattered around the settings file:

```
"download-dir": "/media/NAS1/complete",
"incomplete-dir": "/media/NAS1/incomplete",
"incomplete-dir-enabled": "true",
"rpc-authentication-required": "false",
"rpc-whitelist": "127.0.0.1,192.168.*.*,10.0.*.*",
```

There are also some other settings you can adjust, and I recommend checking out the Transmission website for instructions on how to configure your tool.

Before we start our transmission-daemon, we need to change ownership to our user "odroid" in order to make everything work properly. There are more ideal ways of doing this, but I prefer to stick to a single user since this server will have no other purpose other than as a seedbox and I don't see a need in this situation to stick to the transmission-daemon user that Transmission normally intends to use.

```
$ sudo chown -R odroid:odroid /etc/transmission-dae-
mon
$ sudo chown -R odroid:odroid /etc/init.d/transmis-
sion-daemon
$ sudo chown -R odroid:odroid /var/lib/transmission-
daemon
$ sudo chown -R odroid:odroid /media/NAS1/
```

We also need to open the Transmission daemon service and set user="odroid" from "transmission-daemon":

```
$ sudo nano /etc/systemd/system/multi-user.target.
wants/transmission-daemon.service
```

We also want to do this in the init.d file, switching in "odroid" for "USER":

```
$ sudo nano /etc/init.d/transmission-daemon
```

Finally, reset that related Transmission daemon and turn back on Transmission:

```
$ sudo systemctl daemon-reload
$ sudo service transmission-daemon start
```

Samba configuration

Now we have everything we need to download a file, but we need a way to access the files that we download over our local network. If you're using a Windows device, then Samba is the way to go. Let's get started by installing it and configuring our network share. We will need to configure it in a way that allows us to login with our odroid user and easily download files, as well as remove files we no longer want on our seedbox:

```
$ sudo apt-get install samba samba-common-bin
$ sudo nano /etc/samba/smb.conf
```

Let's add the following as a new config for sharing our relevant folders:

```
security = user

[odroid]
comment = odroid
path = /media/NAS1
valid users = @odroid
force group = odroid
create mask = 0775
force create mode = 0775
security mask = 0775
force security mode = 0775
```

```
directory mask = 2775
force directory mode = 2775
directory security mask = 2775
force directory security mode = 2775
browseable = yes
writeable = yes
guest ok = no
read only = no
```

Before we move on, we'll need to set a Samba password for our "odroid" user. You can also create separate accounts if you so choose, as long as those users are in the "odroid" group:

```
$ sudo smbpasswd -a odroid
```

Now we can restart, and we should be able to remotely access our seedbox folders.

```
$ sudo service smbd restart
```

Keep in mind that you can always add new compatible users, as long as you set a proper Samba password and add those users to the "odroid" group or whichever group you assign access to for these files.

VPN configuration

Finally, we'll want to configure our VPN for an automatic secure connection upon boot. Once again, thanks to the James @ MakeUseOf who figured out this fast and easy way to get your VPN working without needing to manually start the VPN or enter credentials each time you restart your device. This is essential if you want to guarantee the encryption of your connection while using the seedbox. In this guide, we're using an OpenVPN connection, which depends on:

An OpenVPN configuration file
A certificate from your VPN provider
Your username and password for your VPN provider stored in a text file
Three special shell scripts to start our VPN automatically upon boot and route traffic

Storing a password in text is not ideal. However, it's the fastest and leanest way to get things going, and since we're only using this locally, having your VPN credentials stored on this device will not risk the files to those outside your home network. If you want to host your seedbox with port forwarding and essentially open to the Internet, I highly recommend either searching for an alternative method or considering these risks before proceeding.

Next, let's install the OpenVPN software:

```
$ sudo apt-get install openvpn resolvconf
```

We're going to leave everything in our home folder while doing this, which is /home/odroid in this case. It really can be anywhere, but these are not files you want in the same place where people on your home network will connect and download files from your seedbox.

First, get your OpenVPN configuration file (referred to as vpn-server.opvn in this guide) and place it in your home folder along with your certificate file (referred to as ca.crt in this guide). Now create a new text file (.txt) with two lines. The first line should be the username of your VPN connection service, and the second line should be your VPN connection service's password:

```
$ sudo nano /home/odroid/pass.txt
$ username
$ password
```

Next, we're going to open our configuration file, which in my case is "vpn-server.opvn":

```
$ sudo nano /home/odroid/vpn-server.opvn
```

We're going to add the following line at the bottom, which will let us connect to the VPN provider without entering the credentials manually:

```
$ auth-user-pass /mnt/torrents/openvpn/pass.txt
```

Then, we're going to add these three lines which refer to the shell scripts we're about to create:

```
$ route-up /home/odroid/route-up.sh
$ down-pre
$ down /home/odroid/down.sh
```

This will allow OpenVPN to connect to our VPN provider automatically. Once it connects, it will automatically route all traffic through these VPN connection, whether incoming (down.sh) or outgoing (route-up.sh). Next, we're going to create these two shell scripts:

```
$ sudo nano /home/odroid/route-up.sh
```

Enter the following for route-up.sh:

```
#!/bin/sh
iptables -t nat -I POSTROUTING -o tun0 -j MASQUERADE
```

Then, edit down.sh:

```
$ sudo nano /home/odroid/down.sh
```

Enter the following for down.sh:

```
$ #!/bin/sh
$ iptables -t nat -D POSTROUTING -o tun0 -j MASQUER-
ADE
```

Lastly, we need a shell script that we can launch when our OS boots and use to initiate the VPN connection automatically:

```
$ sudo nano /home/odroid/vpn.sh
```

Enter the following for vpn.sh:

```
$ sudo openvpn --client --config /home/odroid/vpn-
server.ovpn --ca /home/odroid/ca.crt --script-securi-
ty 2
```

Now, let's make all of these files executable:

```
$ sudo chmod +x /home/odroid/route-up.sh
$ sudo chmod +x /home/odroid/down.sh
$ sudo chmod +x /home/odroid/vpn.sh
```

Now we're going to open up our rc.local file, which runs scripts upon system startup:

```
$ sudo nano /etc/rc.local
[code]
```

Add this line right above the "exit 0" line:

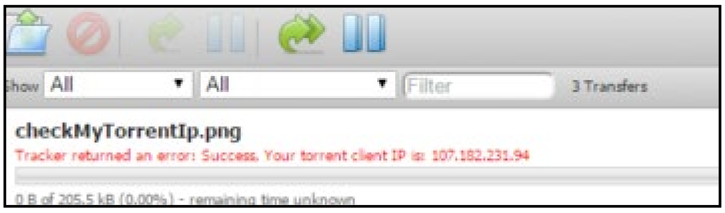
```
[code]
$ /home/odroid/vpn.sh
```

Your VPN is now configured, and your seedbox will download exclusively through an encrypted connection with your VPN provider. You can easily test this by using the TorGuard test file which checks your IP address when downloading the file.

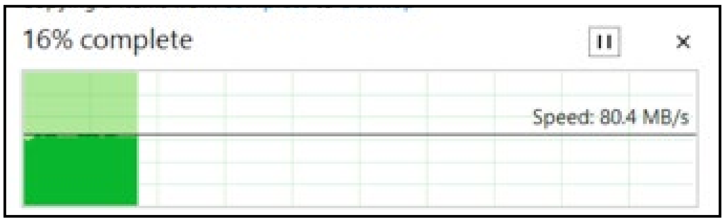
Connect to your Transmission web GUI at odroid:9091, or whatever your local IP address is.

Upload the TorGuard test file and check the IP address reported as "Success! Your torrent client IP is XX.XX.XX.XX" Check this IP address against what Google reports as your IP address by visiting www.whatismyip.com.

If the IP addresses are different, then your connection is secure. You can still use your local IP address to access and download files via Samba. My VPN provider allows me download at speeds around 75 Mbps, with the ODROID- XU4 peaking at around 22 percent CPU usage, which shows just how powerful the octa-core processor is for tasks such as these. However, the real glory is the transfer speeds from your ODROID to your personal computer. I saw 800 Mbps transfer speeds when moving some files back and forth, making this seedbox an extremely viable Network Attached Storage Device too.



checkMyTorrentIp.png
Tracker returned an error: Success. Your torrent client IP is: 107.182.231.94
0 B of 205.5 kB (0.00%) - remaining time unknown



The Transmission environment: look at those speeds!

Feel free to offer any comments or suggestions in the forum topic at http://bit.ly/2fPeek7. The setup is hardly perfect, but it's a surefire way for anyone to get started with their very own seedbox.

MEET AN ODROIDIAN

DANIEL HAZE (@FOURDEE)

edited by Rob Roy (@robroy)

Please tell us a little about yourself.

I live in Burton Latimer in England. It's one of those "ye-olde" village towns, and is a lovely place to live and bring up a family. I am currently a "stay-at-home Dad", since I am dealing with a long term gastrointestinal illness that affects and limits my daily life. Prior to that, I was an IT assistant for a drainage company near Cambridge. I have a 2 and a half year old son named Jack, who loves the television show "PAW Patrol", plays with dust bins (no idea why yet), and building things with Legos. Gemma, my fiance, is a full time mother and my rock. My educational background is in Information and Communications Technology. School wasn't really for me, since I am more of a "hands-on" person who enjoys taking things apart to see what makes them tick.

How did you get started with computers?

My first computer experience was with the Commodore 64 when I was around 5 or 6 years old. However, for some reason, the only memories I have are of the tape drive with a 2+ hour loading time. Even then, you were lucky if the game actually loaded. Those were the days! I was also fortunate enough to have an Amiga 500+ in my early years. For me, this was the start of my love for computers. I used MOD tracker to play the song Axel F, and remember Workbench, Elite Frontier and not having the 1MB RAM upgrade to play "Rise of the Robots".

What attracted you to the ODROID platform?

I read reviews of the C1 and benchmark comparisons with the Raspberry Pi 2. I snapped one up right away, and was simply blown away by how much faster it was in every area of performance. Ever since then, I've been hooked on ODROIDs. It's great to see Hardkernel continually innovate and push their products to new levels. I'd even go as far to say that ODROIDs are years ahead of anything that the Raspberry Pi Foundation currently offers.

How do you use your ODROIDs?

I have one ODROID-C2 which is used for daily testing with my DietPi image. This is mostly due to the C2's excellent I/O performance with the eMMC, allowing for an extremely fast testing environment, especially when it comes to automated software installation tests. I also have another C2 with the HiFi Shield 2 addon, which I use as a dedicated music system with Music Player Daemon and the YMPD interface. I also run an XU4 in a Cloudshell enclosure, which is mainly used for displaying statistics with DietPi-Cloudshell, network attached storage (NAS), backup mirroring with DietPi-Sync, which is based on rsync, and running PiHole to block advertisements on our local network.

Daniel and his wife Gemma

Which ODROID is your favorite and why?

Without a doubt, it's the ODROID-C2. For me, it is the best all-around single-board computer (SBC) on the market today. Coupled with 1GB Ethernet and an EMMC capable of 140Mb/s transfer rates, it's a performance beast capable of anything from a NextCloud server to a HiFi system. ODROIDs have come a long way since the original C1. All those small issues with unstable power supplies and SD card incompatibility have been ironed out. The C2 in my eyes is SBC perfection.

Your DietPi image is very popular. What was your motivation for developing it, and what improvements do you plan to make in the future?

Daniel's hometown of Burton Latimer in England

DietPi originally started as a minimal image. It was aimed at reducing the resource load of the bulky default images provided by the first Raspberry Pi model. We had these 700mhz devices using far more resources than was actually required by the user. Now DietPi has evolved to much more than a minimal image, offering automated installations of popular software, all of which are optimized and configure for you. With DietPi, you get the maximum performance from your device because only the software you need is installed.

As for the future, DietPi has always been shaped by our end user feedback, suggestions and support. We do have plans for a web interface in DietPi, ideally to replace the whiptail menu system, and to provide a modern GUI experience for users.

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

Personally, I would love to see a USB 3.0 port and bus on the next C-series board. DDR4 memory would be sublime, and boost overall performance, especially with the shared GPU memory. An energy-efficient onboard WiFi and bluetooth would also be a welcome addition to future boards.

What hobbies and interests do you have apart from computers?

Music has always been close to my heart. I'm a true audiophile with a love for all genres, but especially trance. I like spending time with family and friends. My son is now into Legos, and it's a real joy to build things together. Although, even when he's asleep, it's easy to get carried away and attempt to build the Millennium Falcon!

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

If you are new to programming, do not be put off by a lot of strange code on the screen. Programming isn't as hard as you think. Follow an online guide for "my first program" and stick with it. With practice and patience, all that strange code will gradually make sense and it's a great feeling.

For beginners, I'd highly recommend Python. Its simple to learn, with a vast community and many online guides to get started. It's also the leading programming language for GPIO projects on SBCs.

Finally, but most importantly, regardless of your skill level or chosen language, always make sure you code something that excites you, and enjoy every moment of it.

Daniel's photogenic son and master Lego builder, Jack