SMS Integration • Lakka • G Spot • Cubicle Commando • Pocket Putter

ODROID Magazine

Year Six
Issue #70
Oct 2019

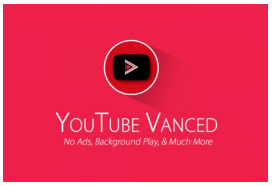
SPEED BEYOND LIMITS

BLASTING THROUGH THE 10000MBPS NETWORK SPEED LIMIT WITH THE ODROID-H2

YOUTUBE VANCED:
PLAY YOUTUBE VIDEOS LIKE A PRO

WATER COOLING:
GET THE MAXIMUM FROM YOUR ODROID

MONKU R3:
BUILDING THE ULTIMATE ODROID-XU4 / XU4Q GAMING CONSOLE



YouTube Vanced: Play YouTube Videos Like A Pro

October 1, 2019

YouTube Vanced is a modded version of YouTube for Android.

Lakka: Building The Ultimate ODROID-XU4 / XU4Q Gaming Console

October 1, 2019

Introduction and Tutorial Goals Hello and welcome to our ODROID-XU4 Lakka gaming console build tutorial. This review will show you in detail how to build a powerful Lakka based retro video game console from scratch. You will need some parts. I've listed the ones I used above and placed a ▶

Networking At Ludicrous Speed: Blasting Through The 10000Mbps Network Speed Limit With The ODROID-H2

October 1, 2019

I remember being an early adopter of 1 GbE networking in the 90s and many of my friends were wondering: who needs 1 GbE anyway? Twenty years later, I am an early adopter (but really not the first) of faster than 10 GbE at home and I hope that this ▶

Is That a Linux Computer in Your Pocket, or Are You Just Glad to See Me?: Build an ODROID Computer You Can Carry in Your Pocket

October 1, 2019

Fresh on the heels of the ODROID Tablet project, https://magazine.odroid.com/article/build-a-rootin-tootin-dual-bootin-odroid-tablet-using-the-odroid-c0-to-make-a-professional-grade-tablet-for-under-usd100/, comes an even more portable version of the ODROID-C0. Rather than sporting a large-scale HDMI-equipped LCD, this “pocket ‘puter” relies on a framebuffer-driven video output displayed on a 3.2-inch thin-film-transistor (TFT) touchscreen shield dubbed the C1. While this touchscreen shield ▶

The G Spot: Your Goto Destination for all Things That are Android Gaming

October 1, 2019

Sony Interactive Entertainment won the coveted Best of gamescom award for its creation of the sandbox game, Dreams.

Low Cost Water Cooling for your Single Board Computer: Get The Maximum Speed From Your ODROID

October 1, 2019

SBC water cooling is not new and others have implemented designs using off the shelf components for the ODROID-XU4 and other SBC. Some implementations have already been covered in ODROID Magazine in the past. December 2016 https://magazine.odroid.com/wp-content/uploads/ODROID-Magazine-201612.pdf July 2018 https://magazine.odroid.com/article/liquid-cooling-part-1-cluster/ https://magazine.odroid.com/article/liquid-cooling-part-2-server/ The focus of this project was initially to make ▶

Five Minute Fun with your Monku R1: Retro Cubicle Commando

October 1, 2019

This tutorial shows you how to convert your Monku Retro console into a Retro Cubicle Commando!

SMS Text Messaging Status Updates With Your CloudShell2: Remote Monitoring Made Simple

October 1, 2019

There is a new version of the Cloudshell2 Info and Monitoring Tool, Version 1.0.4-1. Now you can add GSM Shield support via USB2UART deviceand add new command line switch to identify a defective hard disc more easily (though it is still experimental)

Monku R3: Building The Ultimate ODROID-XU4 / XU4Q Gaming Console - Part 1

October 1, 2019

Requirements Tools: A small screwdriver set that contains a few small Phillips head screwdrivers. A clean static free work surface. Monitor or TV with HDMI support to test the device. Mac SD card image writing software. I use balenaEtcher, it is free and works great. Window SD card image writing ▶

YouTube Vanced: Play YouTube Videos Like A Pro

October 1, 2019 By @LazyBunny Android, ODROID-C2, Tutorial

YouTube Vanced is a modded version of YouTube for Android. It has Ad Blocking, and also lets you configure your default play resolution and auto-play and auto-repeat features. Visit ttps://vanced.app/. It is worth it, especially if you like to play youtube videos using Google Assistant.

We cannot install YouTube Vanced using TWRP, or Magisk. Also, Youtube must be installed as a system app, i.e., not from the play store. I talked with the Youtube Vanced developer and he told me of a much easier way of doing this. Hence this tutorial. Do not let the length of this tutorial overwhelm you. It is all pretty easy.

Things you will need

- A Fresh install of Android for the ODROID-C2. I used this during the making of this tutorial: viewtopic.php?f=137&t=19203#p266354
- OpenGAPPS ARM 6.0 STOCK. Not PICO or NANO, do not unzip it: **https://opengapps.org/**
- A gapps-config.txt that's been slimmed down **https://tinyurl.com/y2vrnb8v**. Read notes at the bottom regarding this
- A file browser with Root Access. I use MiXplorer v6.39.2, and will be using it throughout this tutorial **https://tinyurl.com/jmggvey**.
- YouTube Vanced Root armeabi-v7a version. Do not need the Installer. **https://vanced.app/APKs?type=ROOT**
- Optional but HIGHLY Recommended: An eMMC Module for your OS, and not a microSD card.

Process

There are many steps. However, they are quick, easy, and pretty much fool-proof. You are already using an ODROID, so you already have the skills needed for this.

Step 1: Install Android onto your eMMC or microSD card. It may work on an existing install. Just uninstall any and all google apps that you have installed through the play store, especially YouTube, and the

Google search app. I was able to upgrade a PICO Install to NANO for Google Assistant, but since my stock install removes stuff that pico/nano have by default, it may or may not work. Use at your own risk.

Step 2: Download OpenGAPPS ARM 6.0 STOCK. Do not unzip it. **https://opengapps.org**/. Not ARM64, or x86/x86_64. Do not install it yet, just move onto the next step for now.

Step 3: Download the gapps-config.txt. **https://tinyurl.com/y2vrnb8v**. This gapps-config.txt slims it down so it will fit. You must have YouTube installed as a system app.

Step 4: Download MiXplorer: **https://tinyurl.com/yxh6rr3l**.

Step 5: YouTube Vanced armeabi-v7a root version, of your choice. White/Dark, or White Black. If you have not already **https://vanced.app/APKs?type=ROOT**. Rename your download to Youtube.apk, and create a copy of it, named Youtube2.zip. Extract the ZIP version. Youtube_version_(armeabi-v7a)(nodpi)(vTheme-v2.0.9)-vanced.apk to Youtube.apk

Step 6: Copy the open_gapps-arm-6.0-stock-**********.zip, the gapps-config.txt, the MiXplorer APK, and Youtube.apk and the /lib/ from the extracted youtube2.zip onto a separate thumb drive, or microSD card if you are using an eMMC.

Step 7: Copy the open_gapps zip, the gapps-config.txt, and MiXplorer apk, but not the YouTube.apk or /lib/ folder, to your Internal Download folder. open_gapps-arm-6.0-stock-**********.zip and the gapps-config.txt must be in the same folder.

Step 8: Remove the Thumb Drive/MicroSD card, or whatever you used to transfer the files.

Make sure your Thumb Drive/MicroSD card is removed!

Step 9: Install open_gapps: Open the ODROID Utility, click the little icon in the top right, click Package Install from Storage, and navigate to the Download folder and select the open_gapps-arm-6.0-stock-numbers.zip. It will ask if you want to proceed. Then select Proceed. Let it do its thing. It will only install what it needs from the gapps-config.txt.

After open_gapps is installed, it will probably crash while trying to login, you will see that YouTube is now installed on your ODROID in the app drawer. As well as Google (the google app, this is good, it means Google Assistant will work right).

Step 10: Open the Google Play Store. After you log in, click the 3 bars in the top left, scroll down to Settings, and Click on Auto-Update apps. Then select "Do not auto-update apps." Just manually check for updates every so often, and do not update YouTube.

Step 11: Go into your Downloads and Install MiXplorer by its APK.

Step 12: Go to Settings -> Apps -> scroll down to YouTube and click Force Stop. Then Click Disable.

Step 13: Check to see that the youTube app is gone from the app drawer.

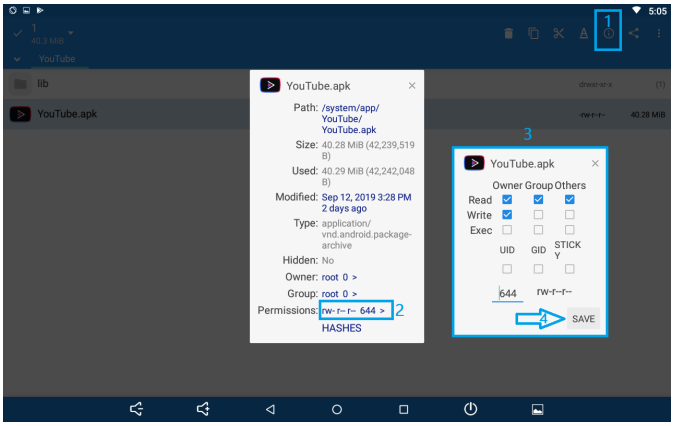
Step 14: Open MiXplorer, Click the 3 lines in the top left corner and click Root. Supersu will ask if you want to allow Root access. Click grant access forever, then go ahead and grant it access.

Step 15: Go to /System/app/Youtube/ and you should see a single Youtube.apk inside it. Press/hold your finger/mouse on the file until its selected, then press the Trash/Rubbish-Bin on the top right to delete it. Yes, delete the Youtube.apk

Step 16: Put the thumb drive/mSD back in, inside MiXplorer, click the 3 lines, select your device, open it, go to the youtube folder. Hold on the Youtube.apk, tap the lib folder so both are highlighted. Then tap the icon that looks like 2 pages next to the rubbish bin in the top right to copy files.

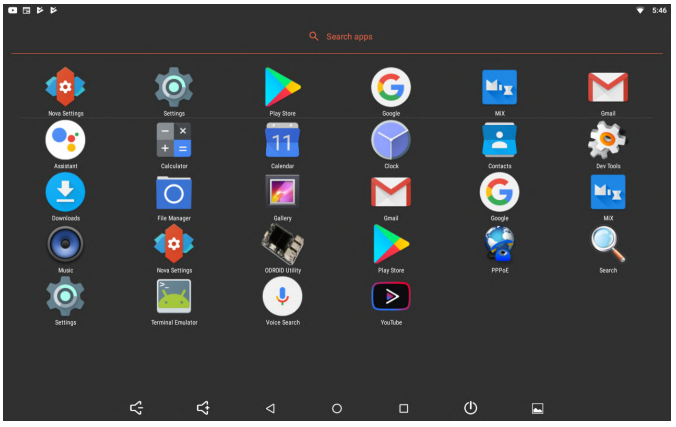
Step 17: Go back to /System/app/Youtube/ and Click the first icon in the top right, the Clipboard, and click Copy. And it will copy the Youtube.apk and the libfolder into /system/app/Youtube/. So it should look like /system/app/Youtube/Youtube.apk /System/app/Youtube/lib/armeabi-v7a/ bunch of .so files.

Step 18: After they are done copying, hold your pointer on Youtube.apk, click the 5th icon on the top right, the circle with an (i), and click where it lists the permissions. Top 3, first, middle, none bottom. Then save.



Step 19: Reboot your ODROID-C2.

Step 20: Go into your Android's Settings -> Apps -> Youtube, and Enable. Go into your app drawer, and you should see Youtube Vanced there, with a brand new icon.



Block Google Play Store notifications in Sound & Notification settings to block notifications about updating YouTube. Enjoy! Go into your YouTube Settings in the app and down to the Vanced settings. From here you can alter your stuff like Resolution, ads, and auto-plays.

Question & Answers

Why do we need to do it this way? We need YouTube installed as a system app to replace it properly with YouTube Vanced. However, since we do not have a recovery menu, we cannot use TWRP. This way is still pretty easy.

Why not just use the non-root version? The non-root version is treated as a separate app. It is not even named YouTube Vanced. So it does not work with Google Assistant.

Gapps config

The gapps-config.txt is a custom file. The whole gapps stock would not fit, so I had to pick and choose. Anything installed with gapps is installed as a system app. If you need some other stuff installed, for example the Google Dialer app normally in PICO, you can put it back. https://tinyurl.com/of6q2eq. However, if you get an out of space error while trying to install opengapps, you may need to remove something else, like Gmail. Then install it separately, later. Just do not remove "Search" the Google App listed under nano. or YouTube. If you keep the Google app, and have it installed as a system app, you can use Google Assistant with Activate on Voice Match. Due to this, I never recommend you install the PICO gapps. You should install nano, but use the gapps config to only include the Pico stuff, and Search, from the Nano section.

Using Google Assistant with YouTube Vanced is so nice. "Ok Google, Play Disturbed The sound of silence, on youtube." Without ads.

Can you do this on an older install instead of a fresh install, maybe an install that has Pico or Nano gapps? Maybe... Just maybe. Uninstall YouTube, if you installed it through the Play Store, and then download the stock gapps and the gapps-config.

I did not have any problems upgrading from pico to stock while testing. However, if it says you do not have enough space. Remove Gmail from the gapps config. and try again.

Reference

https://forum.odroid.com/viewtopic.php?f=137&t=36356

Lakka: Building The Ultimate ODROID-XU4 / XU4Q Gaming Console

October 1, 2019 By Brian Ree Gaming, ODROID-XU4

Introduction and Tutorial Goals

Hello and welcome to our ODROID-XU4 Lakka gaming console build tutorial. This review will show you in detail how to build a powerful Lakka based retro video game console from scratch. You will need some parts. I've listed the ones I used above and placed a link next to each one. These are the actual items I've used in the past and I find them to be reliable. SD cards do fail and sometimes with no warning but for the most part I've had no problems with the parts listed.

This tutorial will cover the setup, and construction of the game console from a hardware and software point of view. Now unlike the Monku Retro 1, 2 we won't be adding any special hardware buttons. The ODROID-XU4 comes with a hard reset button built in, so that's already done for us. As for the custom control button I have not found a good location for it using the current case. However, this device is much more powerful than the C1+ or even the C2 and it is also fairly more reliable so for now we'll not add one to the device. We will also cover all the software setup including installing and configuring Lakka, retroarch, and certain emulators. Most of the software configuration steps will be covered in part 2 of this tutorial. Let's take a look at some of the features of the device we're working on, wow look at that emulator list!

XU4 Features

- ODROID Goodness!
- Hardware Reset Button
- Support for Atari 2600, Atari 5200, Atari 7800, Atari Jaguar, Atari Lynx, ColecoVision, Commodore64, MSX-1, MSX-2, NES, GameBoy, GameBoy Color, Virtual Boy, SNES, N64, GameBoy Advance, WonderSwan Pocket/Color, NEO GEO Pocket/Color, Sega SG-1000, Sega Mark 3, Sega Master System, Sega Genesis, Sega

GameGear, Sega Dreamcast, NEC Turbo Graphics 16, NEC Super Graphics, PSP, and PS1 emulators configured and ready to go.
- Lakka and Retroarch with XBM.

Take a look at the performance specs of this device when compared to some other common devices.

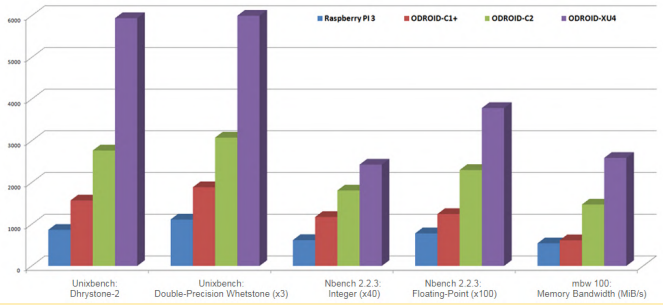


Figure 1 - ODROID performance comparison

Tools Needed

- A small screwdriver set that contains a few small Phillips head screwdrivers.
- A clean static free work surface.
- Monitor or TV with HDMI support to test the device.
- USB Keyboard

Parts Needed

- ODROID-XU4 / ODROID-XU4Q x1: $49.00 / $49.00 (On Sale normally $59)
- Case x1: $5.40
- 64GB Micro SD Card x2: $16.99
- HDMI Cable x1: $1.00
- Power Supply 5V/4A x1: $5.50
- GameSir Wired Controller x1: $17.00

Hardware

First thing's first, let's go over the tools and parts, lay them out, and get ready to build. We have an electronics screwdriver set. If you've built an ODROID-GO the same screw driver set should work fine here. Notice we have our device, an ODROID-XU4 is depicted below, this tutorial applies equally to the ODROID-XU4 or the ODROID-XU4Q version of this device. The device runs just about every emulator you can think of and it runs them wonderfully. We have our board, case, SD cards, and tools all ready to go.

Figure 2 - Parts needed for the build

Clear your workspace and grab the case, take it out of its plastic bag if need be, place it down in the center of the work space. There are two main clips on the case, all in all, it's easier to work with than the C1+ / C2 cases. The first main clip is on the left hand side of the case bottom near the top. The second main clip is on the top side of the case bottom near the right. You can see slight rectangles near these areas in the image below.

Figure 3 - ODROID-C2 case side and top view

Figure 4 - Case clip close up

To clear the first main clip give the case a slight skew as shown below. Ever so slightly pushing the bottom to the left while pushing the top to the right should do it.

Once it comes undone flip the case around so that the other main clip is in the position depicted below. Apply a similar set of forces until the clip separates. It should come apart easily once you get the right set of subtle forces on it. Notice we're using a similar technique for the second main clip as we used for the first.

Figure 5 - Case close up of clips along the side

Figure 6 - tools and part for case assembly

Once you have the case separated you'll find a surprise inside. A bag of tiny screws. If you have an ODROID-GO and you have a good amount of leftover screws I would recommend using them instead of the screws provided. Now this could have changed but at one time the default case screws were a bit smaller than the ODROID-GO screws and I found the extra screws in the ODROID-GO kit to be easier to work with. Let's layout the tools and parts we need to assemble the case with the XU4 board mounted inside. You won't have to worry about SD card access, if you're used to the C1+ or C2 case, the SD card is easily accessible by default.

Carefully open the antistatic bag that the XU4 comes in. Make sure you don't have a static charge by discharging yourself against something large and metal. Layout the case and the board, rest the board on top of the antistatic bag, we're going to place the board on the bottom half of the case and place and tighten the 2 internal screws. The XU4 case is similar to the C1+ / C2 case in that there are two internal screws and 2 external screws. Tip: Make sure the screws are tight but don't over tighten them, snug would be a good description of how much to tighten them.

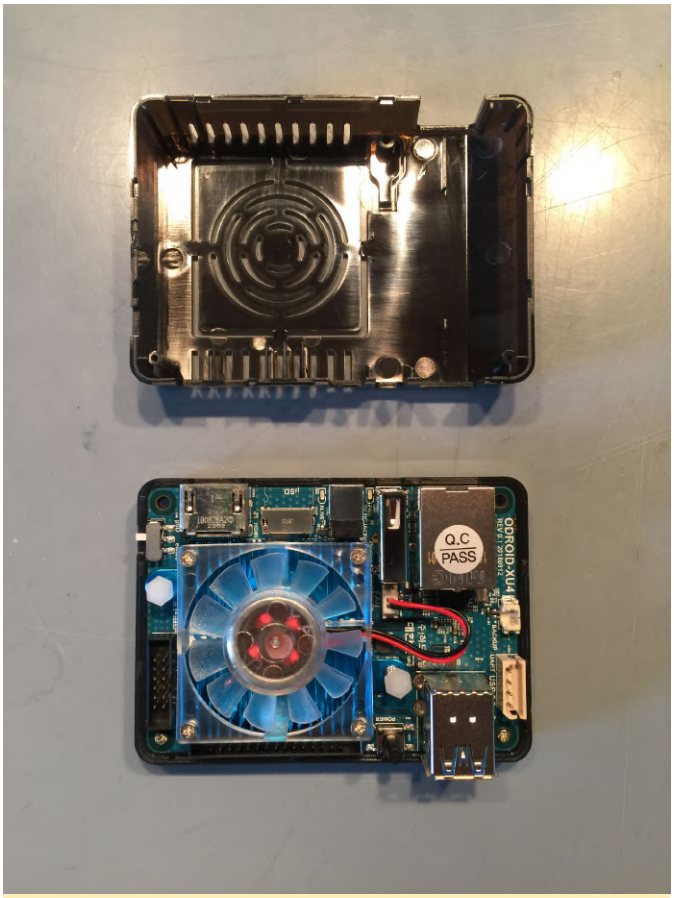


Figure 7 - ODROID-XU4 resting in the bottom of the case

Flip over the case and place and tighten the two external screws. With the back of the case facing towards you take notice of a small white switch. Flip the switch closer to the edge of the case for SD card use, flip the switch the other way, closer to the center of the case, for eMMC use. Bam! You're all done with the hardware construction. Next up we'll be working on the base SD card and OS image.

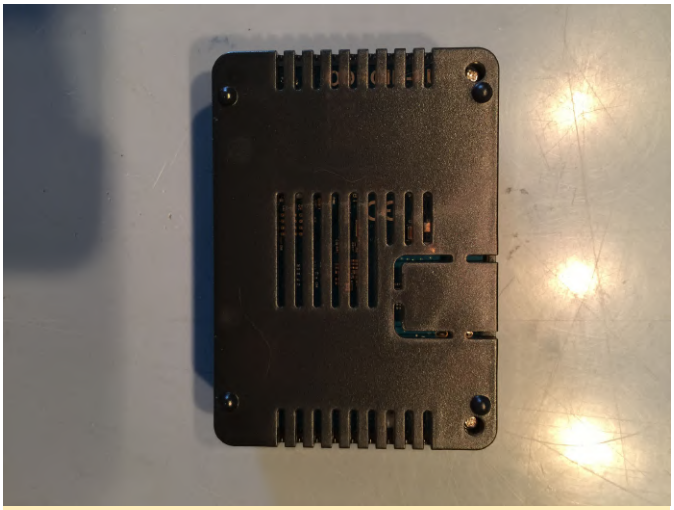


Figure 8 - Screw together the case

The Software

For the next step we're going to locate a specific Lakka OS image. Navigate your browser to lakka.tv. This is the place to go to find versions of Lakka configured for different embedded systems. Click the Get Lakka button on the landing page. You'll be brought to a disclaimer page, it's ok, it is just telling you the software is under development - nothing more than that. Click the Get Lakka button and then click on the Linux image in the subsequent page's OS choices.

You'll be brought to a screen that has a listing of Lakka images custom prepared for different embedded systems. Find the ODROID section depicted below.

Figure 9 - ODROID selection on Lakka download website)

Click on the ODROID XU3/4 entry then click on the Download Lakka button presented on the next page. The image file is around 300MB so it'll take a little while to download but not too long. In the meantime get your SD cards ready and test them out on your SD card reader or SD card USB adapter. Once the image is done downloading I'll show you how to write the image to your SD card. The filename at the time of this writing for the target SD card image is Lakka-OdroidXU3.arm-2.2.2.img. Don't worry that it says XU3 and not XU4 the image will work fine on your device.

Next let's get ready to write the image to an SD card. Etcher is a tool to easily flash SD card on macOS, Windows, and Linux.. Select the uncompressed OS image we just downloaded. Insert your micro SD card into your Mac either using a converter of some kind, link to one listed above, or using a native SD card drive. Make sure to select the proper target drive. You don't want to overwrite important data so make sure to double check the destination drive. Once you're sure everything is set correctly then flash the OS image to the SD card, this will only take a few minutes. The SD card will be unmounted and ready to remove at the end of the process.

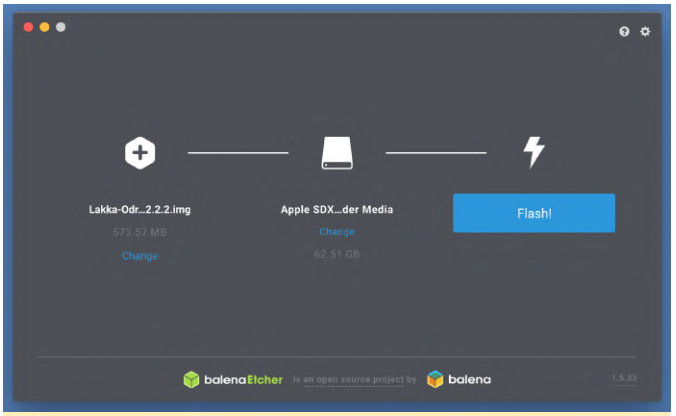


Figure 10 - Screenshot of etcher setup to flash an SD card

Let's get ready to configure Lakka and get our ROMs and Controller ready. Before we do so we have to set things up some we can control the device remotely. I like to use SSH to connect to my Lakka devices so that is the method I'll cover. Keep in mind the default login for Lakka OS is as follows.

User: root Password: root

I won't be covering how to sure up security on Lakka, keep in mind that it is not necessarily configured with security in mind. It's a good idea to turn off SSH once you're done configuring things and loading up ROMs. You'll have to turn on SSH by navigating to Settings -> Services and turn on SSH. First thing's first let's find out what IP address the XU4 Lakka device is running on. I use a cable connection as opposed to a USB WiFi device for simplicity sake. On the main menu category there is an entry called Information depicted below.

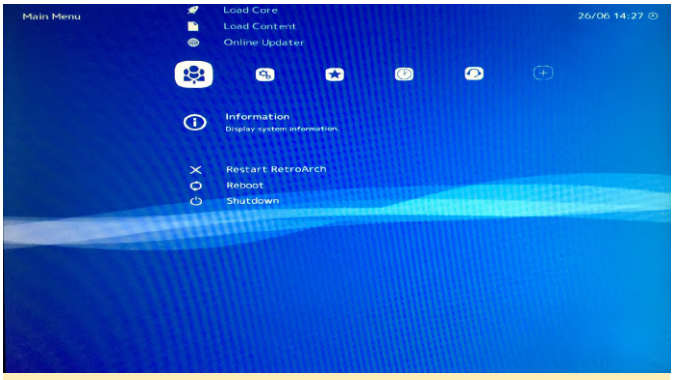


Figure 11 - Lakka Information entry

Select the Network Information entry and you'll be able to see the device's current IP address if the network connection is working.

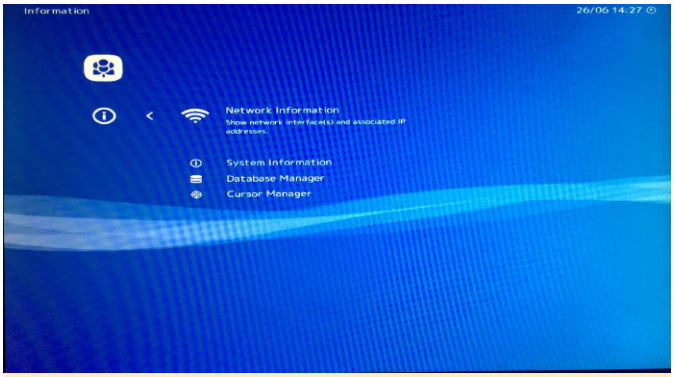


Figure 12 - Lakka network information

You can connect to the XU4 Lakka device via SSH on macOS and Linux, even from another ODROID device running Ubuntu if you have one...hint...hint. You'll have to substitute the IP address of your XU4 Lakka device for the one displayed in the following screenshots.

In windows you'll have to install putty, a free SSH client. You can find putty at this URL, https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html. Select the version you need for your system, 32bit or 64bit, and install it.

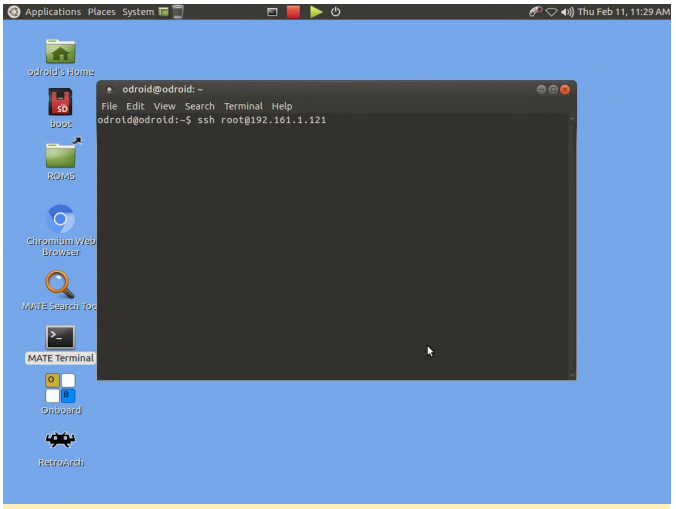


Figure 13 - SSH command from macOS or Linux

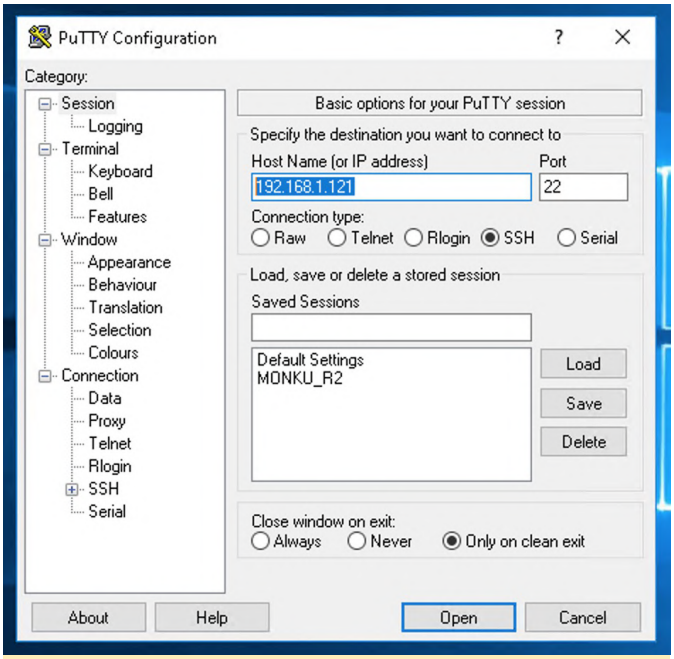


Figure 14 - Putty on Windows

That wraps up this part of the tutorial. Tune in to Part 2 for steps on how to configure retroarch, copy over your ROMs, and setup a controller.

This article was taken from middlemind.com, for more information please visit the original website or Lakka at the following links:

http://middlemind.com/tutorials/odroid_go/ra_lakka_cfg.html

www.lakka.tv

Networking At Ludicrous Speed: Blasting Through The 10000Mbps Network Speed Limit With The ODROID-H2

October 1, 2019 By Dominique Hermsdorff Linux, Tutorial

I remember being an early adopter of 1 GbE networking in the 90s and many of my friends were wondering: who needs 1 GbE anyway? Twenty years later, I am an early adopter (but really not the first) of faster than 10 GbE at home and I hope that this article will convince you to become one too.

As of today economical solutions for deploying a 10 GbE network at home or in a small office are still uncommon and costly. The 10 GbE consumer market is unjustifiably overpriced (10 GbE has been in use in data centers and large organizations for around 15 years. All the issues and glitches have been tamed for quite some time. 10 GbE is NOT a new thing in the enterprise world, it's more like "so passé". This will not prevent consumer market companies to present it as the new best thing you must have). RJ-45 based 10 GbE NICs can cost more than $400. Only cards based on the Tehuti or Aquantia chipsets provide sub-$100 solutions. On the switch side, 10 GbE switches cost $300-500 for a limited (let's say 4 or 5) number of ports (If you need at least 8 ports, the prices climb quickly; see, for instance, **https://www.newegg.com/p/N82E16833156580** or **https://www.newegg.com/p/N82E16833122848**). If you need 18 ports or more, a 10 GbE switch will burn around $1,500 of your hard won money. Microtik, a Latvian company, sells 10 GbE switches at more affordable prices (**sub-$200**), but they use SFP+ not RJ45 (SFP+ by the way is a less expensive solution as of this writing). You also need to update your cabling, CAT-5e won't carry 10 GbE, preferably you want to invest into CAT-6a. The main PC motherboards OEMs or NAS vendors are just finally proposing boards and appliances with 10 GbE NICs onboard.

Clearly the 10 GbE RJ-45 market is still nascent and prices still need to go down for it to become a mass market, as 1 GbE is today. I was tired of waiting for

affordable 10 GbE solutions so I started to look for alternatives or second-hand hardware on eBay.

Although at first glance you may think it sounds a little preposterous if not ridiculous to think you can communicate at 10,000+ Mbps with a small single board computer (SBC), it is exactly what this article is going to show. It will also explain how you can do it yourself. So no I didn't blow up a few MOSFETs in the upper layers of the gray matter that serve as my brain.

Intel was the initiator and king of the 1 GbE networking revolution back in the 90s. Most cards of the period were bearing the Intel chipsets. Not so with 10 GbE or higher. During the last 20 years successful start-ups like **Mellanox**, **QLogic**, and others targeted the data center market by offering 20 Gb/s, 40 Gb/s, 100 Gb/s, 200 Gb/s solutions (1 GbE means one gigabit over Ethernet per second. 1 Gb/s means one gigabit over a network the latter being Ethernet or something else). It does not stop at 200 Gb/s. If one doubles or triples the cabling then speeds of 400 Gb/s or 600 Gb/s can "easily" be achieved. As of this writing there is an AI company whose training and inference appliances have enough onboard RDMA-based NICs providing a whopping 2.4 Tb/s bandwidth. See interesting news report at **https://www.servethehome.com/favored-at-facebook-habana-labs-eyes-ai-training-and-inferencing/**. Nvidia has since then acquired Mellanox. Intel acquired QLogic. While acquisitions and consolidation have maintained a quick pace in the last 10 years, Mellanox was and is definitely the 800-pound gorilla of the fast interconnect industry.

Mellanox came up with an alternative to Ethernet: InfiniBand (aka **IB**) and **RDMA**. The latter means Remote Direct Memory Access. I will not spend time detailing the advantages of InfiniBand, just google it for more information. RDMA deserves some explanation. When you communicate over Ethernet using the IP (Internet Protocol) stack (i.e. TCP/IP) a lot of things happen on the local and destination computers. The data to be sent gets copied several times through the IP stack buffers on your local computer, then the packets transit as frames over the network, these packets are then reassembled on the destination computer where again the data gets copied multiple times through the IP stack buffers to finally end up in the destination application. Note that all these manipulations and data copying are performed by the CPU on each computer. That's a lot of CPU cycles!

RDMA works in a drastically different manner: the "client" computer runs the application that initiates a request with the "server" computer that runs the target/destination application. The data to be sent is already present in the client application memory space so RDMA copies it directly over the network (InfiniBand) to the server application memory space. Done! No CPU involved, the communication is entirely handled by the InfiniBand NIC. Obviously, I'm somewhat simplifying but this the essence of RDMA in a few sentences, so you can imagine how performing RDMA let's say over a 40 Gb/s network can be compared to IP over a 1 GbE Ethernet network.

Mellanox made a lot of $$$ and a name for itself by selling InfiniBand/RDMA hardware in huge numbers. Data centers and large organizations loved it. So much that they asked Mellanox if it could implement IP on top of it so that IP-based applications could run over InfiniBand/RDMA without code change. Mellanox obliged and came up with IP over IB (Internet Protocol over InfiniBand aka ipoib). Under competitors pressure and feature requests Mellanox then came up with RoCE (RDMA over Converged Ethernet). That's basically the reverse: running RDMA over existing good quality Ethernet cables (saving data centers and large organizations a re-cabling of their facilities.)

After 20 years of activity and multiple upgrades and re-provisioning of InfiniBand hardware in data centers and large organizations it is not surprising to find a huge stash of used IB hardware on eBay at incredible prices. You can get 40 Gb/s NICs at prices starting as low as $20. You can also get 18-port 40 Gb/s unmanaged switches costing as low at **$100~$125**.

So why paying several hundred dollars for 10 GbE brand new hardware when you can get 40 Gb/s hardware for an order of magnitude less! You start thinking "But I know nothing about InfiniBand and RDMA". Don't worry, neither I did less than one year

ago 🙂 You just have to be an avid reader of manuals and technical documentation.

Tips about eBay

- Check out the vendor before clicking on the Buy button.
- Good used enterprise computer hardware vendors include several pictures of the product, especially the one with the product label.
- If unsure contact the vendor for more details. The good ones do answer.
- Do not buy on impulse, do your homework and make a list of what you exactly want to buy.
- If you do not find a good deal, try again day after day for several weeks, good deals are constantly coming and going.
- One you received the items, test them right away. If you are sure they do not work contact the vendor, the good ones will accept returns or replacements.

Concise Minimal InfiniBand Vocabulary

VPI	Virtual Protocol Interconnect
VPI Card	A Mellanox network card that can be configured as InfiniBand card or as Ethernet. With dual cards, one port can be InfiniBand and the other port Ethernet.
EN Card	A Mellanox network card that is exclusively Ethernet.
RDMA	Remote Direct Memory Access
IPoIB	Internet Protocol over InfiniBand. Allows you to run IP-based applications on top of RDMA.
RoCE	RDMA over Converged Ethernet. If you are wondering what the "Converged" means just know it is just a marketing buzzword. From a technological viewpoint there is no converged Ethernet. There is just Ethernet. The converged qualifier was invented to sell the fact that most network protocols were now converging to run on top of Ethernet. It was part of the marketing war that occurred between Mellanox and Chelsio (iWarp) during which the latter was trying to disparage InfiniBand. Anyway, it's like saying that celebrities are converged people.
SFP	Small Form-factor Pluggable. A small transceiver that plugs into the SFP port of a network switch and connects to Fibre Channel and Gigabit Ethernet (GbE) optical fiber cables at the other end. There are multiple types. Usually 1 GbE.
SFP+	Enhanced small form-factor pluggable. Usually 10 GbE or 10 Gb/s.
QSFP, QSFP+	Quad Small Form-factor Pluggable (QSFP). 4-lane cable, up to 40 Gb/s.
QSFP14	Cables up to 56 Gb/s
QSFP28	Cables up to 100 Gb/s
DAC	Direct Attached Cable (almost a synonym of passive copper cable)
AOC	Active Optical Cable

Network Bandwidth

	Name	Line Encoding	Speed for 4 x lane cables
SDR	Single Data Rate	8b10b	10 Gb/s
DDR	Double Data Rate	8b10b	20 Gb/s
QDR	Quad Data Rate	8b10b	40 Gb/s
FDR10	Fourteen Data Rate	8b10b	56 Gb/s

FDR	Fourteen Data Rate	64b66b	56 Gb/s
EDR	Enhanced Data Rate	64b66b	100 Gb/s
HDR	High Data Rate	64b66b	200 Gb/s

All these speeds are available today. The vast majority of InfiniBand (or Ethernet SFP) cables are 4-lane cables. However there are 8-lane and 12-lane cables. With these cables the maximum theoretical speed is doubled or tripled.

So when you look at IB products on eBay, you will mostly see QDR or FDR cards, switches and cables for reasonable prices. The used EDR hardware is more expensive. Be careful with FDR and FDR10, many vendors do not distinguish although there is a real difference between the two.

This leads me to the "Line Encoding" column included in the table above. Transmitting bits over a line consists in the case of let's say a copper wire in squared signals of high and low voltage using a frequency based on the clock. If we want to send 1,000 bits equal to 1 as is the electrical would become a very long signal of high voltage. This bring multiple issues: first such a long signal will not work well, second the transceivers will have great pain to maintain the synchronization of their clocks: the square signal becomes basically MIA. So for physical reasons you want to limit the number of consecutive bits of the same value. To do so the transceivers scramble the data. The transmitter scrambles the data and add a few bits that will allow the receiver to descramble them. I'm simplifying here, but that's basically the story.

The 8b10b line encoding means each time you send 8 bits, the very low level of the NIC will scramble them and add 2 bits to allow the receiver to descramble them. This has an obvious effect on the bandwidth purely dedicated to your data aka the payload. If you feel a sudden angry pressure to start a "Save the Bandwidth!" online movement, don't do it. Line encoding is pervasive and you have been using it since you were born. Examples: PCI Express 3, SATA 3, DisplayPort 2.0 use 128b/130-132b, USB 3.1 Gen 1 uses 8b/10b while USB 3.1 Gen 2 128b/132b.

The difference between FDR10 and FDR is now easily explained. FDR10 provides 56 x 8 / 10 = 44.8 Gb/s max bandwidth. FDR provides 56 x 64 / 66 = 54.3 Gb/s. So before buying an FDR card, switch or cable refer to the model number and its documentation to be sure. Mellanox has switches that bear the same model number where only the sub-number tells you if it is FDR10 or FDR. In doubt, ask the vendor to confirm what the model is.

For QDR, the max bandwidth is 40 x 8 / 10 = 32 Gb/s.

After this introduction, let's now see how you can do 10,000+ Mbps with the ODROID-H2.

ODROID-H2

The ODROID-H2 is an SBC from Hard Kernel, a Korean based manufacturer of small boards. You probably have heard of Raspberry Pi the first series of products who popularized the usage of SBCs. Many agree that Hard Kernel is the number two of the SBC market. Hard Kernel makes more powerful SBCs than Raspberry, makes sure that the Linux and Android solutions are of good quality and supports the hardware well, sells plenty of accessories allowing you to extend the applicability of their boards, publishes a magazine, maintains a wiki and has a very active forum community with dedicated members who are glad to answer questions and provide help.

If you combine all of that think of Hard Kernel like the Ikea of the SBC world.

For a detailed reference about the H2, see:

- the product page at **https://www.hardkernel.com/shop/odroid-h2/**
- the wiki page at **https://wiki.odroid.com/odroid-h2/start**
- the forum pages at **https://forum.odroid.com/viewforum.php?f=167**
- the main Odroid distributor in the US is Ameridroid: **https://ameridroid.com/products/odroid-h2**

PCIe with H2

The Mellanox NICs are PCIe cards. The H2 has no PCIe slot. So what are we going to do? The H2 board has

on the back an NVMe slot PCIe 2 with 4 lanes (PCIe 2 x4). We can use an NVMe PCIe x4 SSD for very fast disk I/O... or we can use an NVMe to PCIe x4 adapter to get the PCIe slot we need. A good adapter is the ADT Link model or a similar alternative, as shown in Figures 1 and 2.

Figure 1 - R42SF available at ADT
http://www.adt.link/product/R42-Shop.html

Figure 2 - R42SR available at AliExpress
https://www.aliexpress.com/item/32833359557.html

In both cases, you can customize the adapter: its length, the orientation of the PCIe slot (forward, top, bottom). Choose the option the most convenient to your project. On one end you have an M.2 M key connector, on the other end an open ended PCIe x4 slot.

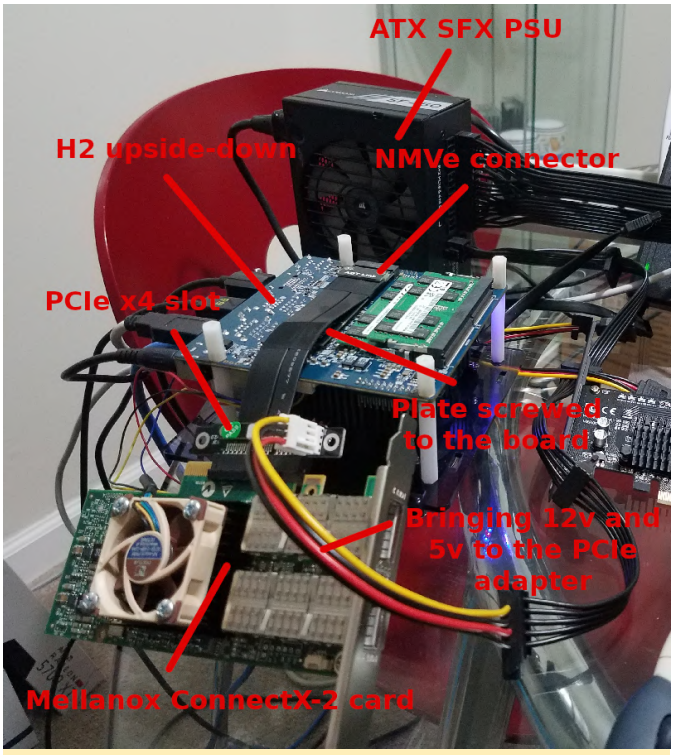


Figure 3 - The adapter installed on the ODROID-H2

Software Packages

Let's now install the minimal set of packages we need for going through this article.

Linux utilities

```
~$ sudo apt install vim htop iperf3 net-tools
parted inxi smartmontools ssh
```

InfiniBand packages

```
~$ sudo apt install rdma-core opensm ibutils
ibverbs-utils infiniband-diags perftest mstflint
```

Linux Utilities Packages Description

vim	Vim is an almost compatible version of the UNIX editor Vi. Many new features have been added: multi level undo, syntax highlighting, command line history, on-line help, filename completion, block operations, folding, Unicode support, etc. Note: many of you use nano anyway.

htop	htop is an ncurse-based process viewer similar to top, but it allows one to scroll the list vertically and horizontally to see all processes and their full command lines.
iperf3	iperf3 is a tool for performing network throughput measurements. It can test either TCP or UDP throughput. This is a new implementation that shares no code with the original iperf from NLANR/DAST and also is not backwards compatible. This package contains the command line utility.
net-tools	This package includes the important tools for controlling the network subsystem of the Linux kernel. This includes arp, ifconfig, netstat, rarp, nameif and route. Additionally, this package contains utilities relating to particular network hardware types (plipconfig, slattach, mii-tool) and advanced aspects of IP configuration (iptunnel, ipmaddr). In the upstream package 'hostname' and friends are included. Those are not installed by this package, since there is a special "hostname*.deb".
parted	GNU Parted is a program that allows you to create, destroy, resize, move, and copy disk partitions. This is useful for creating space for new operating systems, reorganizing disk usage, and copying data to new hard disks. Parted currently supports DOS, Mac, Sun, BSD, GPT, MIPS, and PC98 partitioning formats, as well as a "loop" (raw disk) type which allows use on RAID/LVM. It can detect and remove ASFS/AFFS/APFS, Btrfs, ext2/3/4, FAT16/32, HFS, JFS, linux-swap, UFS, XFS, and ZFS file systems. Parted also has the ability to create and modify file systems of some of these types, but using it to perform file system operations is now deprecated. Note: parted is the replacement for fdisk when you have to create partitions larger than 2TB. Once you get used to parted, you use it no matter the size of the partitions you want to create and stop using fdisk.
inxi	inxi is a system information script that can display various things about your hardware and software to users in an IRC chatroom or support forum. It runs with the /exec command in most IRC clients.
smartmontools	The smartmontools package contains two utility programs (smartctl and smartd) to control and monitor storage systems using the Self-Monitoring, Analysis and Reporting Technology System (S.M.A.R.T.) built into most modern ATA and SCSI hard disks. It is derived from the smartsuite package, and includes support for ATA/ATAPI-5 disks. It should run on any modern Linux system.
ssh	This metapackage is a convenient way to install both the OpenSSH Client and the OpenSSH Server.

	ssh (Secure Shell) is a program for logging into a remote machine and for executing commands on a remote machine. It provides secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel. It can be used to provide applications with a secure communication channel. OpenSSH Client This package provides the ssh, scp and sftp clients, the ssh-agent and ssh-add programs to make public key authentication more convenient, and the ssh-keygen, ssh-keyscan, ssh-copy-id and ssh-argv0 utilities. OpenSSH Server This package provides the sshd server.

InfiniBand Packages Description

rdma-core	RDMA core userspace infrastructure and documentation This package provides the basic boot time support for systems that use the Linux kernel's remote direct memory access (RDMA) sub-system which includes InfiniBand, iWARP, and RDMA over Converged Ethernet (RoCE). Several kernel RDMA support daemons are included: - The rdma-ndd daemon which watches for RDMA device changes and/or hostname changes and updates the Node Description of the RDMA devices based on those changes. - The iWARP Port Mapper Daemon (iwpmd) which provides a kernel support service in userspace for iWARP drivers to claim TCP ports through the standard socket interface.
opensm	InfiniBand subnet manager OpenSM provides an implementation of an InfiniBand Subnet Manager (SM) and Administrator (SA). One Subnet Manager is required to run on each InfiniBand subnet in order to initialize the InfiniBand hardware.
ibutils	InfiniBand network utilities This package contains a set of utilities useful for diagnosing and testing InfiniBand based networks.
ibverbs-utils	Examples for the libibverbs library libibverbs is a library that allows userspace processes to use RDMA "verbs" as described in the InfiniBand Architecture Specification and the RDMA Protocol Verbs Specification. iWARP ethernet NICs support RDMA over hardware-offloaded TCP/IP, while InfiniBand is a high-throughput, low-latency networking technology. InfiniBand host channel adapters (HCAs) and iWARP NICs commonly support direct hardware access from userspace (kernel bypass), and libibverbs supports this when available.

	This package contains useful libibverbs1 example programs such as ibv_devinfo, which displays information about InfiniBand devices.
infiniband-diags	InfiniBand diagnostic programs InfiniBand is a switched fabric communications link used in high-performance computing and enterprise data centers. Its features include high throughput, low latency, quality of service and failover, and it is designed to be scalable. This package provides diagnostic programs and scripts needed to diagnose an InfiniBand subnet.
perftest	InfiniBand verbs performance tests This is a collection of tests written using InfiniBand verbs intended for use as a performance micro-benchmark. The tests can measure the latency and bandwidth of InfiniBand fabrics.
mstflint	Mellanox firmware burning application and diagnostics tools This package contains a burning tool and diagnostic tools for Mellanox manufactured host channel adapters (HCA) and network interface cards (NIC). This burning tool should be used only with Mellanox manufactured HCA/NIC cards. Using it with cards manufactured by other vendors may be harmful to the cards (due to different configurations). Using the diagnostic tools is normally safe for all HCAs/NICs.

Mellanox InfiniBand Network Cards

At this point we assume the ADT link is securely connected to the board, the card inserted in the PCIe slot, the H2 has a working Ubuntu 18.04 OS. Let's start the H2 and open a terminal.

Identifying the Mellanox card on the PCIe bus

Let's look for the card using the lspci utility command. Type:

```
~$ lspci | grep Mellanox
```

This should return something similar to this:

```
01:00.0 InfiniBand: Mellanox Technologies
MT25408A0-FCC-QI ConnectX, Dual Port 40Gb/s
InfiniBand / 10GigE Adapter IC with PCIe 2.0 x8
5.0GT/s In... (rev b0)
```

To get the Link capabilities and status, type:

```
~$ sudo lspci -vv -s 01:00.0 | grep Width
```

This should return two lines similar to these:

```
LnkCap: Port #8, Speed 5GT/s, Width x8, ASPM L0s,
Exit Latency L0s unlimited, L1 unlimited
LnkSta: Speed 5GT/s, Width x4, TrErr- Train-
SlotClk- DLActive- BWMgmt- ABWMgmt-
```

LnkCap means link capabilities. See above (Speed 5GT/s, Width x8). This card is capable of handling 5 GT/s (GigaTransfers per second - see https://en.wikipedia.org/wiki/Transfer_(computing) and https://paolozaino.wordpress.com/2013/05/21/converting-gts-to-gbps/) over 8 PCIe lanes. LnkSta means link status. See above (Speed 5GT/S, Width x 4). The card will handle 5GT/s but only 4 PCIe lanes are available.

Due to this reduction in available PCIe lanes, we will lose more bandwidth.

What to do if the card does not show up?

1. Did you type lspci | grep Mellanox or lspci | grep mellanox? The "M" should be capitalized.

2. If the card is really not showing up

- reboot the H2,
- press the Delete key several times at boot time (when the Hard Kernel logo appears) so that you enter the BIOS,
- inside the BIOS, go to Chipset and then PCI Express Configuration
- inside PCI Express Configuration disable PCI Express Clock Gating:

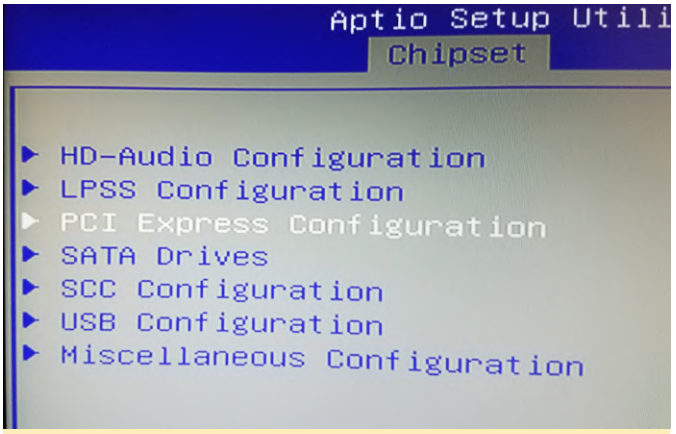


Figure 4 - Go to Chipset and then PCI Express Configuration

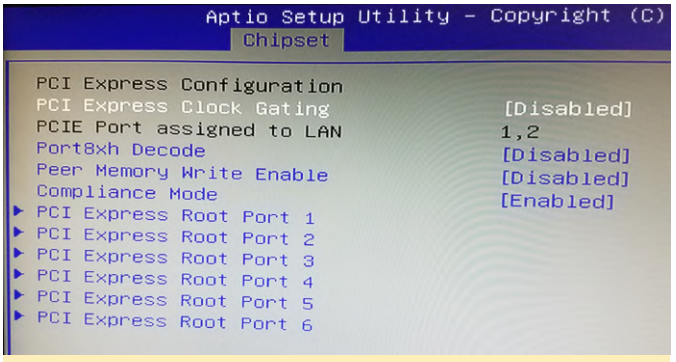


Figure 5 - Inside PCI Express Configuration disable PCI Express Clock Gating

Save the BIOS changes, wait for the H2 to reboot and try again.

From this point, we assume the card correctly shows up in lspci, inxi and any other Linux tools showing the hardware present on the board. To display more information about the Mellanox card, let's use the mstflint command:

```
~$ sudo mstflint -d 01:00.0 query
```

This should return something similar to this:

```
Image type:            FS2
FW Version:            2.9.1000
Device ID:             26428
Description:           Node             Port1
Port2            Sys image
GUIDs:                 0002c9030010ea44
0002c9030010ea45 0002c9030010ea46 0002c9030010ea47
MACs:
0002c910ea44     0002c910ea45
VSD:
PSID:                  MT_0FC0110009
```

The three fields that interests us at this point are FW Version, Device ID and PSID.

FW Version: it is the firmware version currently running on the card. For this particular card, 2.9.1000 is the current (and final) firmware. No update necessary.

PSID (Parameter-Set Identification): it is a 16-ascii character string embedded in the firmware image which provides a unique identification of the device. An intrinsically identical model (in terms of hardware) will have different PSID depending on who is distributing the product and under which brand.

Device ID: not sure, never got the gist of that one. Looks like a device ID that can be shared by several model revisions.

Which cards work on the ODROID-H2?

Here are 3 examples I validated:

Mellanox SeriesConnectX-2 VPI up max 20Gb/s

Label on the card	
lspci | grep Mellanox	InfiniBand: Mellanox Technologies MT25408A0-FCC-GI ConnectX, Dual Port 20Gb/s InfiniBand / 10GigE Adapter IC with PCIe 2.0 x8 5.0GT/s In... (rev b0)
sudo lspci -vv -s 01:00.0 | grep Width	LnkCap: Port #8, Speed 5GT/s, Width x8, ASPM L0s, Exit Latency L0s unlimited, L1 unlimited LnkSta: Speed 5GT/s, Width x4, TrErr-

	Train- SlotClk- DLActive- BWMgmt- ABWMgmt-
sudo mstflint -d 01:00.0 query	Image type: FS2 FW Version: 2.9.1000 Device ID: 26418 Description: Node Port1 Port2 Sys image GUIDs: 0002c903005a0aa0 5849560e65cf0d01 5849560e65cf0d02 0002c903005a0aa3 MACs: 0002c95a0aa0 0002c95a0aa1 VSD: PSID: MT_0F90120008
Sudo iblinkinfo	3 7[ ] ==(4X 5.0 Gbps Active/ LinkUp)==> 1 1[ ] "h2a mlx4_0" ( )

Mellanox SeriesConnectX-2 VPI up max 40Gb/s

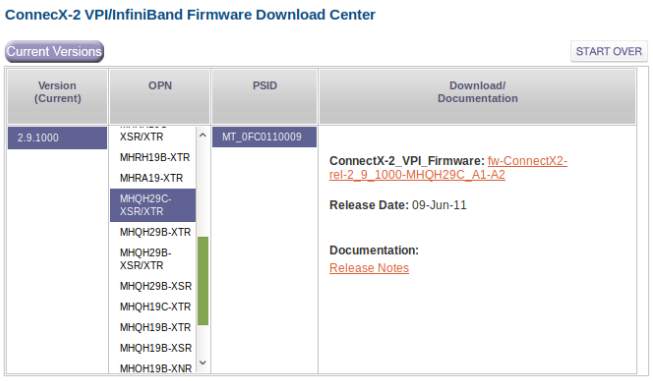
Label on the card	
lspci | grep Mellanox	InfiniBand: Mellanox Technologies MT25408A0-FCC-QI ConnectX, Dual Port 40Gb/s InfiniBand / 10GigE Adapter IC with PCIe 2.0 x8 5.0GT/s In... (rev b0)
sudo lspci -vv -s 01:00.0 | grep Width	LnkCap: Port #8, Speed 5GT/s, Width x8, ASPM L0s, Exit Latency L0s unlimited, L1 unlimited LnkSta: Speed 5GT/s, Width x4, TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgm
sudo mstflint -d 01:00.0 query	Image type: FS2 FW Version: 2.9.1000 Device ID: 26428 Description: Node Port1 Port2 Sys image GUIDs: 0002c9030010ea44 0002c9030010ea45 0002c9030010ea46 0002c9030010ea47 MACs: 0002c910ea44 0002c910ea45 VSD: PSID: MT_0FC0110009
Sudo iblinkinfo	3 7[ ] ==(4X 10.0 Gbps Active/ LinkUp)==> 1 1[ ] "h2a mlx4_0" ( )

You find the product brief of the ConnectX-2 VPI cards at **https://www.mellanox.com/related-docs/prod_adapter_cards/ConnectX-2_VPI_Card.pdf**. The cards listed at the bottom of this PDF should work fine on the ODROID-H2:

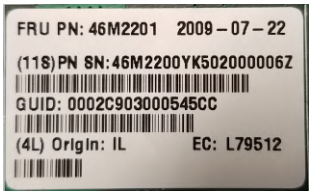
MHRH19B-XTR	Single 4X QSFP 20GbE/s InfiniBand	6.7W
MHQH19B-XTR	Single 4X QSFP 40GbE/s InfiniBand	7.0W
MHRH29B-XTR	Dual 4X QSFP 20GbE/s InfiniBand	8.1W (both ports)
MHQH29C-XTR	Dual 4X QSFP 40GbE/s InfiniBand	8.8W (both ports)
MHZH29B-XTR	4X QSFP 40GbE/s InfiniBand, SFP+ 10 GbE	8.0W (both ports)

You find these cards on eBay at **https://www.ebay.com/sch/i.html?_from=R40&_nkw=Mhqh29c&_sacat=0&_sop=15**, expect $20~$25 for the cheapest offers. Change the search keyword to look for another card model to find alternates. Note that some of these cards top at 20 Gb/s while the others top at 40 Gb/s. Given the almost no difference in pricing, go with the 40 Gb/s ones. You find the latest firmware of these cards at **https://www.mellanox.com/page/firmware_table_ConnectX2IB**.

Use the card label, the lspci description and the mstflint DeviceID and PSID to select the right firmware to download and then install. The ConnectX2 cards are no longer officially supported by Mellanox. The latest firmware is 2.9.1000 (from 2011). If you card is already running this firmware, you have nothing to do.



Mellanox SeriesConnectX IB max 40Gb/s

Label on the card	
lspci | grep Mellanox	InfiniBand: Mellanox Technologies MT25408A0-FCC-QI ConnectX, Dual Port 40Gb/s InfiniBand / 10GigE Adapter IC with PCIe 2.0 x8 5.0GT/s In... (rev a0)
sudo lspci -vv -s 01:00.0 | grep Width	LnkCap: Port #8, Speed 5GT/s, Width x8, ASPM L0s, Exit Latency L0s unlimited, L1 unlimited LnkSta: Speed 5GT/s, Width x4, TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgmt
sudo mstflint -d 01:00.0 query	FATAL - Can't find device id. -E- Cannot open Device: 01:00.0. File exists. MFE_UNSUPPORTED_DEVICE
Sudo iblinkinfo	3 7[ ] ==(4X 10.0 Gbps Active/ LinkUp)==> 6 1[ ] "h2a mlx4_0" ( )

You find the product brief of the ConnectX-IB at https://www.mellanox.com/pdf/products/silicon/ConnectX_IB_Silicon.pdf. The cards listed at the bottom of this PDF should work fine on the ODROID-H2.

MT25408A0-FCC-SI	Dual 4X (10Gb/s)	PCIe 2.0 2.5GT/s	8.1W
MT25408A0-FCC-DI	Dual 4X (10, 20Gb/s)	PCIe 2.0 2.5GT/s	8.6W
MT25408A0-FCC-GI	Dual 4X (10, 20Gb/s)	PCIe 2.0 5.0GT/s	9.1W
MT25408A0-FCC-QI	Dual 4X (10, 20, 40Gb/s)	PCIe 2.0 5.0GT/s	9.7W

These ConnectX-IB cards are now kind of difficult to find on eBay. They are very old and often appears in the seller description under a totally different name. These cards are pre-ConnectX-2 era. You better go with a ConnectX-2. But if you have one (I have a MT25408A0-FCC-QI) it will work. Note that only the QI goes up to 40 Gb/s anyway. You find the latest firmware of these cards at https://www.mellanox.com/page/firmware_table_ConnectXIB. IMHO, the ConnectX-2 VPI 40 Gb/s is your best bet for usage on the ODROID-H2. Do you need to update the firmware right away? No. If it works, get the time to get acquainted with the thing before updating its firmware.

If the card is actually a Mellanox one, not a distributor, mstflint can grab the appropriate most recent firmware upgrade online and do the upgrade for you. You don't have to play archaeologist in the archives pages of the Mellanox web site. If the card is from a distributor and not a Mellanox one, you will have to.

What to do when mstflint does not recognize the card?

The version of mstflint that comes from the APT repositories is 4.11.0 which is a relatively recent one. The recent versions no longer support the "old" ConnectX IB or ConnectX-2. In the examples above, it did not recognize the MT25408A0-FCC-QI. Because mstflint is the tool you use to update the firmware of the cards, that is a problem.

In addition, you may end up buying a non-Mellanox "flavor" of a card. IBM, Sun, Oracle, HP, Dell, EMC (non-exhaustive list) all distributed Mellanox cards under their brand because at the time it was the only show in town for 10, 20 or 40 Gb/s. In these cases, the PSID on the card will not be a Mellanox one. Mellanox did it this way so that customers of these major brands would download the new firmware and get their support from the distributors and not Mellanox. You will find tons of IBM, Oracle, HP cards on eBay.

But while the brand and the PSID of the cards are not Mellanox it is bona fide regular Mellanox technology on these cards.

Because the "modern" versions of mstflint do not allow you to overwrite the PSID of a card, you have to go back in time and find a version of MST that does. But not too old, otherwise it would not recognize more modern cards. One of the "optimal" versions of MST in this matter is version 4.0.0. It's a modern enough and it still include the old flint tool which has no qualms in overriding the PSID of a card. Hence you can update the firmware.

To download this version, go to: https://www.mellanox.com/page/management_tools and scroll to the bottom of the page (after obviously reading what the page is saying). You see:

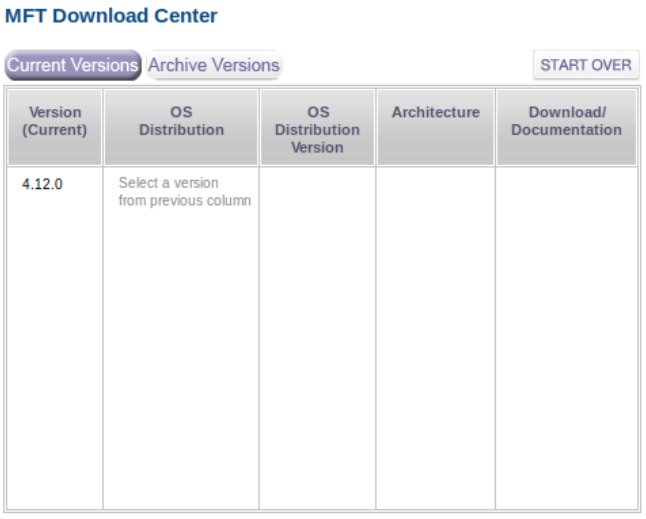


Figure 10 - The bottom of the Mellanox web page

Click on Archive Versions and select the appropriate values in the columns on the right, as shown in Figure 11.

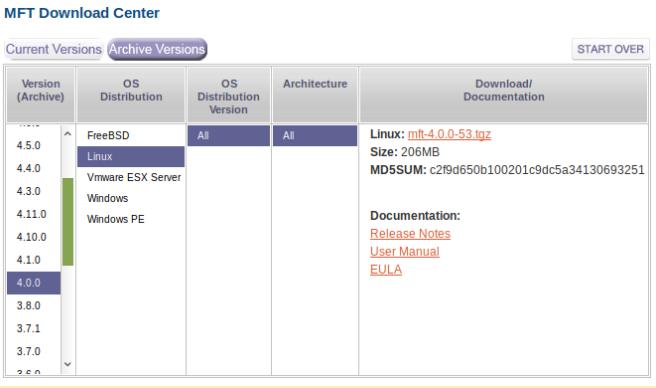


Figure 11 - Archive Versions

Here is an example of updating a card to the new firmware

```
# Become root
~$ sudo su -
# Start the firmware manager
~$ mft start
# Query the card to find out the current firmware
~$ flint -d 02:00.0 query
# If not the latest firmware, update it with the
one you downloaded
# Note the use of –allow_psid_change to deal with
non-Mellanox versions of the card:
~$ flint -d 02:00.0 -i fw-25408-2_9_1000-MHQH29-
XSC_A1-A5.bin --allow_psid_change burn
```

Warning

If you use the --allow_psid_change option, the flint tool will gladly attempt to burn whatever firmware bin file you passed as parameter with zero checking. Using a wrong firmware bin file will obviously and simply brick the card. So check, check and check again you know what you are doing. Otherwise you'll win a free visit to eBay to buy another card. Was I anxious the first time I burnt the firmware with that option? Yep.

My ODROID-H2 is going to talk to modern PCs. Should I use a ConnectX-2 on my PCs?

The answer is a very direct and simple: no. ConnectX-2 card are no longer supported by Mellanox, they are PCIe 2 (which is great for embedded boards like the ODROID-H2). Chance is that your "modern" PCs are probably providing PCIe 3 or even PCIe 4 with the Ryzen 3000 series and chipset X570. For your PCs, use ConnectX-3 Mellanox cards. You can find the product brief of these cards at https://www.mellanox.com/related-docs/prod_adapter_cards/ConnectX3_VPI_Card.pdf. My favorite is the MCX354A-FCBT Dual FDR 56 Gb/s or 40/56 GbE because it has two ports, supports QDR, FDR10 and FDR.

On eBay, see https://www.ebay.com/sch/i.html?_from=R40&_nkw=MCX354A-FCBT&_sacat=0&_sop=15. Expect around $50~$75 for a dual card. Better deals show up on a regular basis. You will also find non-Mellanox ones (mostly HP or

Oracle) for $30. Upgrading the firmware on ConnectX-3 cards works like on ConnectX-2 cards. So even though you buy an HP or Oracle one, you can update its firmware. Upgrading the firmware of the ConnectX-3 to the most recent version is necessary if you are planning to use the Mellanox OFED packages. In this article we use the packages from the Ubuntu APT repository, so there is no immediate need to update the firmware.

Note: do PCIe 3 Mellanox ConnectX-3 cards work on PCIe 4? The ones I tried do.

For your PCs, you can also use ConnectX-3 Pro Mellanox cards. On eBay you find them for about $100~$120 with a better deal from time to time. Again, think about the fact that you want a VPI (that supports both IB and Ethernet) and do not forget the top speed: QDR, FDR10 or FDR.

You find the product brief of the Pro cards there: **https://www.mellanox.com/related-docs/prod_adapter_cards/PB_ConnectX-3_Pro_Card_VPI.pdf**. ConnectX-3 and ConnectX3 Pro are still supported by Mellanox. This translates into support in the Mellanox OFED packages. For Linux, you can download the Mellanox OFED software from **https://www.mellanox.com/page/products_dyn?product_family=26**. Scroll down the page, click on the Download tab and select the flavor for your Linux distribution.

Note: do Mellanox ConnectX-3 cards also work on the ODROID-H2? As of this writing only the Pro version works. The non-pro versions do not. Know anyway that in terms of max speed achievable on the H2, the ConnectX-3 will provide the same result as the ConnectX-2. The ConnectX-2 provide 40 Gb/s in IB mode but only 10 GbE in Ethernet mode. The connectX-3 provide 40 Gb/s in IB mode and 40 GbE in Ethernet mode. However IP over IB provides a speed as good (if not a little bit better) with a port configured for IB as IP on a port configured for Ethernet. So really, no need to run a ConnectX3 Pro on the H2 (unless you already have a 20 or more GbE network running at home based on Ethernet).

Warning

The latest Ubuntu 18.04 apt update brings in kernel 5.x. The current Mellanox OFED package for Ubuntu 18.04 as of this writing (**MLNX_OFED_LINUX-4.6-1.0.1.1-ubuntu18.04-x86_64.tgz**) will not build correctly (probably the KDMS). This is the reason why in this article we use the packages from the Ubuntu APT repository.

Note: since then, Mellanox has released the Mellanox OFED package version 4.7-1.0.0.1 available for Ubuntu 18.04 as well as Ubuntu 19.04.

Is there a difference between Mellanox OFED and your distribution packages for InfiniBand? Yes, there is but you won't see the difference at the beginning. The Linux distribution packages for InfiniBand are mostly written by the Mellanox engineers who also write the Mellanox OFED software. What the latter brings is more tools, more configuration, more docs, more late bugs fixes, more "candies". You really do not need to worry about that.

For Windows, you can download the Mellanox OFED software there: **https://www.mellanox.com/page/products_dyn?product_family=32** To support the ConnectX-3 series, use the Windows – WinOF versions. Mellanox OFED for Windows is the only "simple" way to install the InfiniBand on Windows. This is definitely what you want to use if want to use SMB/CIFS full speed between a Windows Server and Windows 10. For more information start there: **https://docs.microsoft.com/en-us/windows-server/storage/file-server/smb-direct**

As of this writing, Linux Samba does not support RDMA yet. So if you are running Linux on your ODROID-H2 with Samba, the only way your Windows machine can access it is to use regular SMB/CIFS with IP over IB. From this point, we will not talk too much about Windows.

Warning

When buying Mellanox network cards on eBay, make sure you are buying what you really want. VPI cards both support InfiniBand and Ethernet. EN cards only support Ethernet. Dual port cards are usually the best solution because the two ports can be configured to InfiniBand and you can have 3 PCs connected without

the need for a switch (meaning PC A is connected directly to PC B and PC C, when B and C want to talk, they go through the daisy chaining via PC A). Two port cards also allow a PC to be on an InfiniBand subnet via port 1 and (usually a 10 GbE) Ethernet subnet via port 2 (as long as you connect it to another PC or switch with SFP+ connectors, not RJ-45).

Why do I look insisting in using an InfiniBand network rather than Ethernet one? In other words, let's talk about switches.

It is pretty easy to find a used 18, 24 or 36-port QDR, FDR10 or FDR InfiniBand switch on eBay in $125 to $250 price range. On the other hand, a similar switch but Ethernet will deeply hurt the feelings of your bank account (like $1,500 ~ $2,500 hurt.)

Mellanox IS5023 18-port 40Gbps **https://www.ebay.com/sch/i.html?_nkw=Mellanox+IS5023** (this is the one I'm currently using)

Mellanox SX6015 18-port 56Gbps (FDR10) **https://www.ebay.com/sch/i.html?_nkw=Mellanox+SX6015** (I tested OK this one, but it still has to be put it in "production")

Conclusion: using IP over IB with an IB QDR, FDR10 or FDR switch costs much much less while not limited to 10 GbE.

Do you need a switch? Not right away. With 1-port cards you can connect the ODROID-H2 to one PC. With a 2-port card you can connect 2 PCs. If you have more devices you can still daisy chain 2-port cards. Bandwidth will suffer when traffic has to go through multiple cards between device X and Y tough. I did not try the daisy chaining myself I went straight for a switch when I saw the low prices.

The Mellanox switches are built like trucks. The case is quasi unbreakable with thick metal plates. But the switches are also very NOISY beast. The IS5023 has four 40mm fans running at 15K rpm, the SX6015 has six. Remember these are data center models and nobody squats inside a data center so noise is not and issue. For home usage this is an entirely different story. So unless you have a home with a garage attached to the house or a sound-proof attic or basement the other family members will kick the switch out of the home or the apartment probably in the next 5 minutes you turn it on. I'm currently living in an apartment, no attached garage, no attic, no basement, no sound proof room. To solve that, I customized the switch to replace the fans, as shown in Figure 12.

Figure 12 - Customized Mellanox IS5023 switch

Gone the four noisy 15K rpm gremlins, four gentle Noctua 80mm redux took their place. The top plate is gone. Obviously the Noctua fans do not run at the same speed so the switch LED for the fan status is shining bright red. Not a problem. The switch has software thermal protection anyway. Never saw it being triggered. Not placing back any top plate solves multiple issues: (a) I would have have to design one to adapt to the 80mm fans (b) The 80mm fans easily push away the heat emanating from the main heat sink (c) The heat from the AOC transceiver electronics can get away by convection. So far so good. Even by night, the thing is barely audible, just a gentle hum because you know it's there, otherwise you would not notice.

Figure 13 - Below is a wider field view of the installation showing my QDR switch in my little living room data center

Which cables should I use?

You can use passive copper cables (DAC) for short distances (i.e. from 1 to 5 meters) or active optical cables (AOC) for longer distances (from 1 to 100 meters). You can use a mix of both.

The AOC are more “fragile” than the copper ones. Mellanox advise to not bend AOC cables under a radius of 17 cm (when in use). A badly bent AOC will trigger a lot of retries at the RDMA level.

Tips about AOC cables

a) AOC cables can get damaged in various ways, as shown in Figures 14 and 15.

In Figure 14 someone bent the cable very badly. The best way to check if the cable is still good is to run iperf3 for 10 minutes using that cable. If you see too many retries, the cable is dead and irreparable.

Figure 14 - Badly bent cable

Figure 15 - Someone pulled on the cable to disconnect it instead of using the pull tab. This cable is dead and is irreparable

b) Use copper cables to connect computers close to a switch and where space is reduced: AOC cables would start to bend too much and get entangled. The more bulky copper cables won’t.

c) Keep them in a box when not in use. To avoid the spaghettification of multiple cables and the risk of damaging one when disentangling them, roll them around spools. A simple solution is to use **kite winding plastic hoop spool**, as shown in Figure 16.

Figure 16 - Using plastic spool to store AOC cables

For more tips, see: https://www.mellanox.com/products/interconnect/pdf/Mellanox_Cable_Management_Guidelines_and_FAQs_Application_Note.pdf

Next you have to decide what eventual theoretical maximal speed you want to have right away or use in the future: 40 Gb/s (QDR), 56 Gb/s (FDR10) or 56 Gb/s (FDR). As you may guess a 40 Gb/s cable will NOT go faster when you connect it to a network card or switch capable of FDR10 or FDR. On the other hand, an FDR cable will happily work with a QDR or FDR10 network card or switch.

Last but not least, you are buying these cables used or refurbished on eBay. The prices for brand new cables are crazy expensive. Short length DAC coppers cables on eBay are numerous, the longer cables are rarer. For AOC it is basically the reverse: 3m, 5m, 10m are common on eBay, good luck finding a 1 meter.

Tip: when you receive a used DAC or AOC cable from an eBay vendor inspect it and try it for 10 minutes right away. If the cable is faulty, contact the vendor for return or replacement.

Be nice to your hardware and your hardware will be nice to you - especially with the cards. These used cards probably already have 10+ years of activity under the belt. As you may have noticed they come with a heat sink and I can tell you they easily reach 50+ C. A heat sink implies some thermal compound between the heat sink and the chip. That compound has already cooked for 10+ years. Time for a complementary refresh!

The heat sink is attached to the card with two spring rivet fastener push pin. On older cards they were using brass (easy to remove), the more modern cards use plastic ones (a pain in the neck).

Figure 17 - Heatsink with brass spring rivets

Push down on the rivets front side if they are a little bit sticky to the back side. Then use a pair of tweezers (or hemostat) to compress the back pins while gently pulling on the rivet from the front side. If everything goes well, the rivet will go through the board and heat sink holes. The keyword here is “gently” so that you do no break the pins (if plastic). Your mileage will vary, mine is 50% success. If you break the rivet, it is not the end of the world. Either you have bolts and nuts in you stash that will work either you go to https://www.ebay.com/sch/i.html?_nkw=Heatsink+Cooler+Cooling+Fan+Spring+Rivet

Once separated, use isopropyl rubbing alcohol 70% (or similar) to entirely remove the dead compound. You get a very clean new chip and heat sink, as shown in Figure 18. Give the time to the alcohol traces to fully evaporate. Apply some brand new compound (I’m using the extra compound I got with a Noctua CPU cooler. Otherwise you can use a compound such

as Artic MX, see Figure 19), then put back the heat sink and plug back the rivets.

Figure 18 - Heatsink and IC cleaned up

Figure 19 - Adding a drop of thermal compound, enough but not too much

Finally, if you have the funds, add a Noctua 40x10mm to pull the heat from the heat sink as shown in Figure 20. The screws to use are: pan head Phillips 6 x 3/4, they bite exactly the right way between the fins of the heat sink. You card will reward you by not dying in 2 or 3 years.

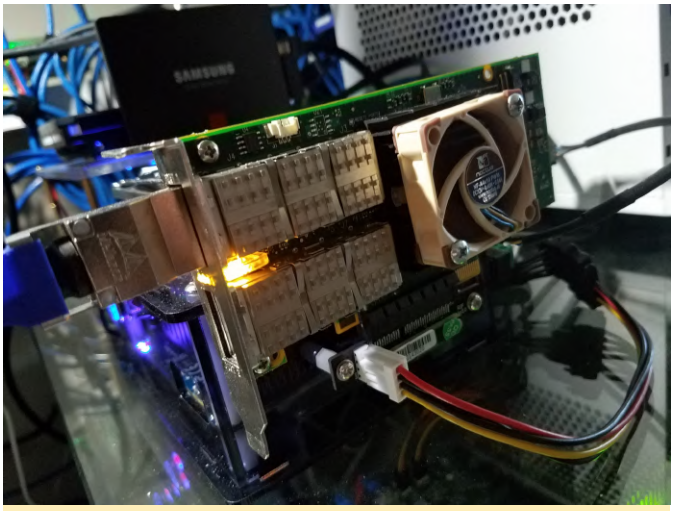


Figure 20 - Adding a 40x10mm fan is a plus

Powering the whole system

The official ODROID-H2 power brick provides 15V x 4A = max 60W. The Hard Kernel documentation (Wiki) indicates that the H2 can be powered by a DC 14V-20V (up to 60W) power supply.

The initial testing of the system (H2 + 10TB WD Red + 1 TB SSD + eMMC + 32Gb memory + Mellanox network card + card fan + chassis fan) showed a burst power consumption of 53W, then 35W to 45W during the various phases of booting to finally land at around 34.5W when idle, 38.5W with the IB card link up and 49.5W when running iperf3 network tests (where disks were not involved.) These measurements were made using the H2 power brick powering everything except the PCIe bus/IB card powered by a separate SFX PSU, which is the small format version of the regular ATX PSU.

This is too close to comfort to the max 60W supported by the lines of the H2 board, for we need some room space for additional devices connecting to the USB ports, and the chassis fan going full speed. So using a separate SFX PSU is warranted. Now it would be nice to solve the simultaneous usage of the two power supplies, this means powering the H2 with the SFX PSU and eliminate the H2 power brick from the equation.

But the SFX PSU can only provide 12V, not in the range of the14V-20V expected by the H2. A DC converter from 12V to 19V (easy to find on Aliexpress.com) could solve the issue. However, the 14V-20V requirement is not an absolute one. The H2

will be a perfectly happy camper with 12V as long as you do not ask him to **power the hard disk and SSD**. So the SFX PSU will power everything including the H2 itself and the SATA disks will get their power from the SFX PSU, not the H2.

Eliminating one power supply led to a significant reduction in power consumption: the system now idles at 24W (with a burst at 38W during boot), which is very manageable from the monthly electric bill viewpoint. Power consumption only climbs to 32W when running iperf3 network tests and max at 35W while copying files back and forth between the H2 and its clients over NFS.

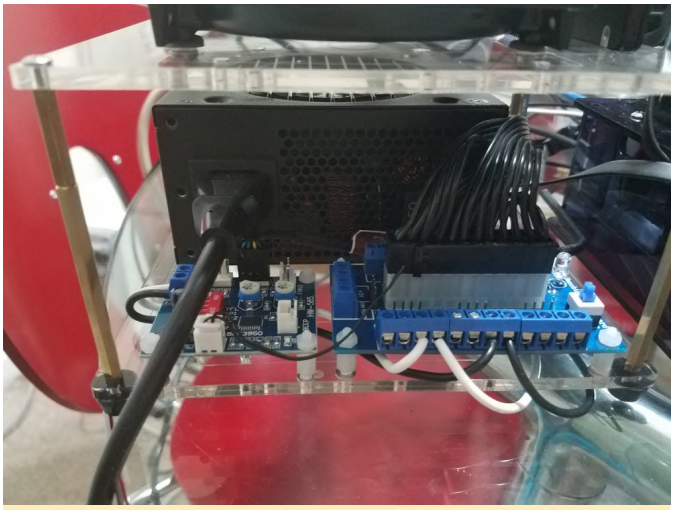


Figure 21 - The final configuration, with the SFX PSU visible in the background

The blue **ATX breakout board** is in the foreground on the right. It includes (on the right) a power button to power the ATX PSU on and off. A blue light LED (behind the button turns on). The ATX power cable plugs in the middle of the board. Such a board makes it easy to get the 12V, 5V and 3.3V power lines without cutting cables, soldering, etc. In other words, the SFX PSU and its cables are left intact for reuse in the future if necessary.

Note: with this configuration, powering the H2 therefore consists in two steps. First, power on the SFX PSU, then power on the H2. And once you shutdown the H2 you have to manually turn off the SFX PSU so that the network card and disks do not consume power for nothing. The white (+12V) and black (GND) wires connected to the board bring power to the small fan PWM controller board (foreground left) and the H2 itself. To plug the power into the H2 you just need **one male 2.1x5.5mm DC power jack plug adapter**. The fan PWM controller board permits to control the 40x10mm fan fixed on the Mellanox network card heat sink (basically bringing it down to sufficient speed while being silent.)

The complete assembled system is shown in Figures 22 and 23.

Figure 22 - Last day on the testing table

Figure 23 - First day in the little living room data center

At this point, we are done with the hardware side of the story. Let's switch to the software side.

Setting Up the IP addresses for InfinBand IP over IB

First check that your Ubuntu 18.04 or 19.04 have its netplan delegating all network configuration to the good old Network Manager:

```
~$ ls /etc/netplan
01-network-manager-all.yaml
~$ cat /etc/netplan/01-network-manager-all.yaml
# Let NetworkManager manage all devices on this
system
network:
  version: 2
  renderer: NetworkManager
```

In the rest of this article, we will use the Network Manager. If you prefer netplan because it is the new best thing on the planet and are a netplan expert guru, you will have no problem following what's below and translating it to the appropriate YAML configuration. At this point, the PCIe IB network card is installed and we successful established that it is recognized. So we should see the two ports available for configuration and activation. Let's see:

```
~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc
noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd
00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc fq_codel state UP group default qlen
1000
    link/ether 00:1e:06:45:0d:47 brd
ff:ff:ff:ff:ff:ff
    inet 192.168.1.70/24 brd 192.168.1.255 scope
global noprefixroute enp2s0
       valid_lft forever preferred_lft forever
    inet6 fe80::bdf4:34b7:f1a3:eb1/64 scope link
noprefixroute
       valid_lft forever preferred_lft forever
3: enp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu
1500 qdisc fq_codel state DOWN group default qlen
1000
    link/ether 00:1e:06:45:0d:48 brd
ff:ff:ff:ff:ff:ff
4: ibp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
65520 qdisc fq_codel state UP group default qlen
256
    link/infiniband
80:00:02:08:fe:80:00:00:00:00:00:00:00:02:c9:03:00
:10:ea:45 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff
:ff:ff:ff
5: ibp1s0d1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
65520 qdisc fq_codel state UP group default qlen
256
    link/infiniband
80:00:02:09:fe:80:00:00:00:00:00:00:00:02:c9:03:00
:10:ea:46 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff
:ff:ff:ff
```

The labels enp2s0 and enp3s0 represent the two ODROID-H2 onboard 1 Gbe ports and yes (!) we have ibp1s0 and ibp1s0d1 (4: and 5: as shown above), the two ports of the ConnectX-2 card. Note that on your ODROID-H2 or other PCs they may show up as ib0 and ib1. It depends on how the naming scheme the NICs has been configured.

What to do if you do not see the IB ports?

If you just installed the IB packages as described earlier a complementary reboot will help a lot. If, after rebooting, you still do not see the network IB ports, you have to dig in to find out what is going on. Is the ib_ipoib kernel module loaded?

```
~$ sudo lsmod | grep ib_ipoib
Should show something like:
ib_ipoib              110592  0
ib_cm                  57344  2 rdma_cm,ib_ipoib
ib_core               249856  10
rdma_cm,ib_ipoib,rpcrdma,mlx4_ib,iw_cm,ib_iser,ib_
umad,rdma_ucm,ib_uverbs,ib_cm
```

If it does not show up, it means the ib_ipoib Kernel module was not loaded. Try:

```
~$ sudo modprobe ib_ipobib
```

If still not successful and depending the error message, your best option is to apt remove the IB packages and reinstall them and reboot. If by any chance you previously had the Linux Mellanox OFED installed, proceed as follows:

```
# Become root
~$ sudo su -
# Go to the Mellanox OFED installation kit
~$ cd MLNX_OFED_LINUX-4.6-1.0.1.1-ubuntu18.04-
x86_64/
# Uninstall it
~$ ./uninstall.sh
# Get rid of it
~$ cd ..
~$ rm -fr MLNX_OFED_LINUX-4.6-1.0.1.1-ubuntu19.04-
x86_64
~$ rm -f MLNX_OFED_LINUX-4.6-1.0.1.1-ubuntu18.04-
x86_64 MLNX_OFED_LINUX-4.6-1.0.1.1-ubuntu18.04-
x86_64.tgz
~$ exit
# Install the apt repositories versions
~$ sudo apt install rdma-core opensm ibutils
ibverbs-utils infiniband-diags perftest mstflint
# Reboot
~$ sudo shutdown -r now
```

If you still do not see the IB ports, go to the ODROID-H2 forum and ask for help from a good Samaritan, and/or google until exhaustion for solutions from other lost souls who had the same issue(s).

Configuring the IB ports for IPoIB use

At this point, we assume that you do see the IP ports. Let's configure and activate them using the Network Manager nmtui command

```
~$ sudo nmtui
```

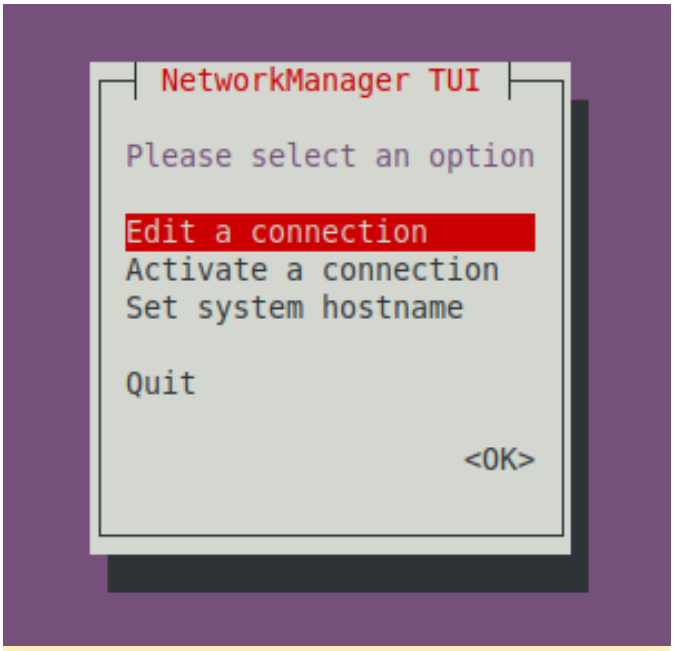


Figure 24 - The nmtui ncurse dialog appears

Select Edit a connection in the ncurse dialog, then Select Add and choose InfiniBand and select Create:

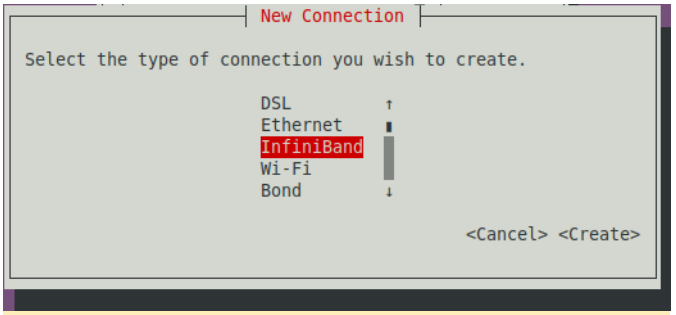


Figure 25 - Choose InfiniBand and select Create

The Edit connection dialog appears, as shown in 26.

Figure 26 - Edit connection dialog

Select or enter the following options:

```
- Transport Mode: Connected,
- MTU: 65520,
- Manual IPv4: 10.10.10.70/8
```

The IP is whatever subnet and mask you prefer as long as it is not the subnet of you regular 1 Gbe network. There is no need for Gateway, DNS servers or Search Domains, the H2 (and PCs) will find the IP addresses of the regular domain names via the 1 Gbe network or wireless. Finally select OK. After exiting this screen, select Back and select Activate a Connection in the main dialog to verify that your new NIC port is active. If not, activate it, as shown in Figure 27.

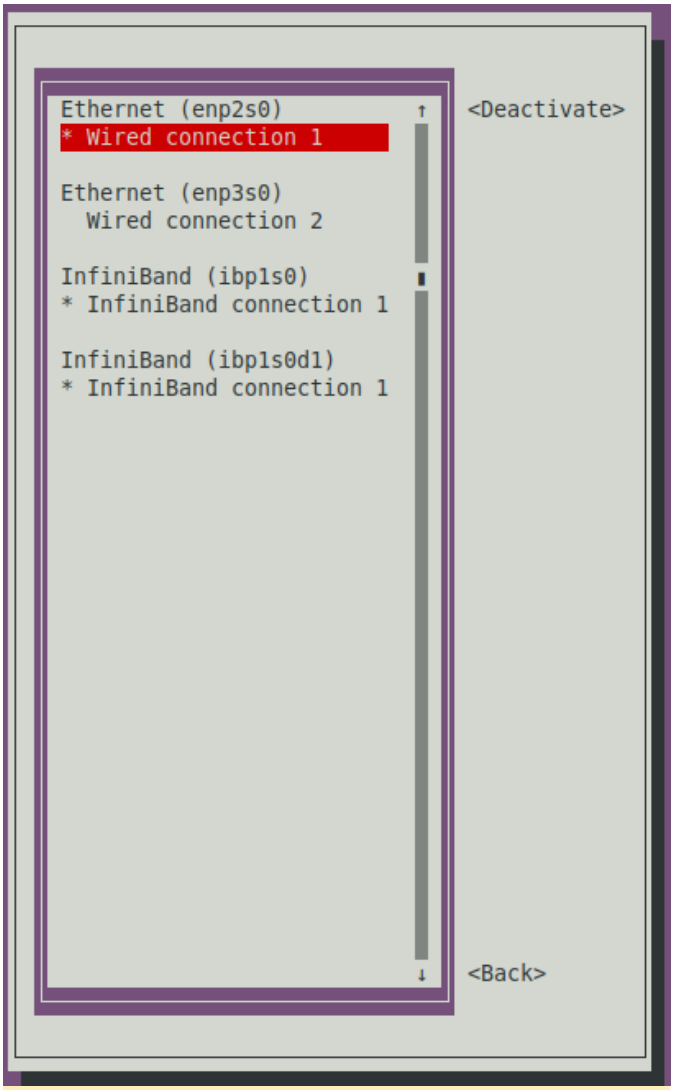


Figure 27 – The nmtui GUI may be confusing because it shows the connection on both card's ports. If I had also configured the second port, it would also show Infiniband connection #2 on both card's ports

Do not forget to add the IP addresses of your IB ports and host names to the host files of your H2 and PCs, or if you are running your own local DNS to add them to it.

Let's verify that our connection is working and the link is up:

```
~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc
noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd
00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
.../...
4: ibp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
65520 qdisc fq_codel state UP group default qlen
256
    link/infiniband
80:00:02:08:fe:80:00:00:00:00:00:00:02:c9:03:00
:10:ea:45 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff
:ff:ff:ff
    inet 10.10.10.70/8 brd 10.255.255.255 scope
global noprefixroute ibp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::76fa:9754:1f17:3ca7/64 scope link
noprefixroute
       valid_lft forever preferred_lft forever
```

Testing connection and bandwidth with iperf3

Note: If iperf3 does not connect between the server and the client, double-check your nmtui configuration then if the problem persists, double check that the kernel module ib_ipoib is loaded. It may also be because the IB network is not yet managed, execute the following command on the H2 or the PC:

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

Once OpenSM is started, execute the following command on the H2 or the PC:

```
$ sudo iblinkinfo
```

It should return something similar to this (LIDs are marked with **):

```
CA: h2a mlx4_0:
0x0002c9030010ea45 1** 1[ ] ==( 4X 10.0 Gbps
Active/ LinkUp)==>
3 6[ ] "Infiniscale-IV Mellanox Technologies" ( )
Switch: 0x0002c902004a3e78 Infiniscale-IV Mellanox
Technologies:
3 1[ ] ==( Down/PortConfigurationTraining)==> [ ]
"" ( )
3 2[ ] ==( 4X 10.0 Gbps Active/ LinkUp)==> 9** 1[
] "ripper
HCA-1" ( )
3 3[ ] ==( Down/ Polling)==> [ ] "" ( )
3 4[ ] ==( Down/ Polling)==> [ ] "" ( )
3 5[ ] ==( Down/ Polling)==> [ ] "" ( )
3 6[ ] ==( 4X 10.0 Gbps Active/ LinkUp)==> 1** 1[
] "h2a
mlx4_0" ( )
3 7[ ] ==( Down/ Polling)==> [ ] "" ( )
3 8[ ] ==( Down/ Polling)==> [ ] "" ( )
```

```
3 9[ ] ==( Down/ Polling)==> [ ] "" ( )
3 10[ ] ==( Down/ Polling)==> [ ] "" ( )
3 11[ ] ==( Down/ Polling)==> [ ] "" ( )
3 12[ ] ==( Down/ Polling)==> [ ] "" ( )
3 13[ ] ==( Down/ Polling)==> [ ] "" ( )
3 14[ ] ==( Down/ Polling)==> [ ] "" ( )
3 15[ ] ==( Down/ Polling)==> [ ] "" ( )
3 16[ ] ==( Down/ Polling)==> [ ] "" ( )
3 17[ ] ==( Down/ Polling)==> [ ] "" ( )
3 18[ ] ==( Down/ Polling)==> [ ] "" ( )
CA: ripper HCA-1:
0xf452140300346ea1 9** 1[ ] ==( 4X 10.0 Gbps
Active/ LinkUp)==>
3 2[ ] "Infiniscale-IV Mellanox Technologies" ( )
```

Check that both devices are listed. In the example above, there are "ripper HCA-1" and "h2a mlx4_0" within the Switch list as well as in independent CA entries. Without a switch it will look the same without the Switch list.

Finally, you can check pinging over InfiniBand, to do so see this **man page**. Note that ibping does not work like ping, and you have to start it in server mode on one machine and then only you can target it from another IB connected machine. Note also that ibping expects a LID as target, not an IP address.

In the example above, the h2a has a LID of 1 and the PC has a LID of 9. So to check that the two devices can talk to each other over IB, you would execute:

```
# On the PC
sudo ibping -S -d -v
# On the H2
sudo ibping 9
# OR
# On the H2
sudo ibping -S -d -v
# On the PC
sudo ibping 1
```

Once you have established that the two devices can talk over IB, you can then further troubleshoot at the IP level with ping and other IP tools over IPoIB. Let's use an i5 9600K PC to talk with the H2 using IP over OB. First let's start iperf3 in server mode on the i5:

```
~$ iperf3 -s --bind 10.10.10.21
-----------------------------------------------
---------
Server listening on 5201
-----------------------------------------------
---------
```

Note: press Ctrl-C to exit the server mode once you're done testing.

Then connect to it as client from the H2:

```
~$ iperf3 -c 10.10.10.21 --bind 10.10.10.70 -t 10
Connecting to host 10.10.10.21, port 5201
[  5] local 10.10.10.70 port 54865 connected to
10.10.10.21 port 5201
[ ID] Interval           Transfer     Bitrate
Retr  Cwnd
[  5]   0.00-1.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   1.00-2.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   2.00-3.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   3.00-4.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   4.00-5.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   5.00-6.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   6.00-7.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   7.00-8.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   8.00-9.00   sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
[  5]   9.00-10.00  sec  1.29 GBytes  11.1
Gbits/sec    0   3.18 MBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
Retr
[  5]   0.00-10.00  sec  12.9 GBytes  11.1
Gbits/sec    0             sender
[  5]   0.00-10.00  sec  12.9 GBytes  11.1
Gbits/sec                  receiver
```

Let's now do the reverse, let's start iperf3 in server mode on the H2:

```
~$ iperf3 -s --bind 10.10.10.70
-----------------------------------------------
---------
Server listening on 5201
-----------------------------------------------
---------
```

Then connect to it as client from the i5:

```
~$ iperf3 -c 10.10.10.70 --bind 10.10.10.21 -t 10
Connecting to host 10.10.10.70, port 5201
[  4] local 10.10.10.21 port 49411 connected to
10.10.10.70 port 5201
[ ID] Interval           Transfer     Bandwidth
Retr  Cwnd
[  4]   0.00-1.00   sec  1.67 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   1.00-2.00   sec  1.67 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   2.00-3.00   sec  1.68 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   3.00-4.00   sec  1.67 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   4.00-5.00   sec  1.68 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   5.00-6.00   sec  1.67 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   6.00-7.00   sec  1.68 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   7.00-8.00   sec  1.67 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   8.00-9.00   sec  1.68 GBytes  14.4
Gbits/sec    0   3.12 MBytes
[  4]   9.00-10.00  sec  1.67 GBytes  14.4
Gbits/sec    0   3.12 MBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
Retr
[  4]   0.00-10.00  sec  16.7 GBytes  14.4
Gbits/sec    0             sender
[  4]   0.00-10.00  sec  16.7 GBytes  14.4
Gbits/sec                  receiver
```

With 11.1 GbE in one direction and 14.4 GbE in the other we passed the 10 GbE mark, hence the eye catching title of this article! At this point you may start thinking, but wait a minute! 11.1 and 14.4 is way below the 40 Gb/s of the InfiniBand network. And you would be entirely right.

However, let's take a closer look at what is going on here:

- We start at 40 Gbe max.
- Remember the 8b10b line encoding? So we are now at 32 Gbe max.
- Remember the card is a PCIe 2 x8, but we only have a PCIe x4 slot, so the traffic on the PCIe bus get reduced by a factor of 2. So we are now at 16 Gb/s max (PCIe 2.0 uses an 8b/10b encoding scheme, therefore delivering, per-lane, an effective 4 Gbit/s max transfer rate from its 5 GT/s raw data rate, and 4 Gbit/s x 4 lanes = 16 Gbit/s)
- Add the processing of the IP stack, the processing of iperf3 itself over the speed of the Celeron J4105 (which is not an i9 9900k or Ryzen 3900x) and yes, we are in the neighborhood of the witnessed 11.1 GbE and 14.4GbE.

Another way to test the connection if you do not have (yet) a second PC with an IB card, and if you have a 2-port card on the H2 is to configure and activate both ports and then connect them to each other:

```
# In one terminal window:
~$ iperf3 -s --bind 10.10.10.70
# In a second terminal window:
~$ iperf3 -c 10.10.10.70 --bind 10.10.10.71 -t 300
Connecting to host 10.10.10.70, port 5201
[  4] local 10.10.10.71 port 49363 connected to
10.10.10.70 port 5201
[ ID] Interval           Transfer     Bandwidth
Retr  Cwnd
[  4]   0.00-1.00   sec  3.18 GBytes  27.3
Gbits/sec    0   1.75 MBytes
[  4]   1.00-2.00   sec  3.14 GBytes  27.0
Gbits/sec    0   1.75 MBytes
.../...
[  4] 299.00-300.00 sec  3.08 GBytes  26.5
Gbits/sec    0   3.12 MBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
Retr
[  4]   0.00-300.00 sec   895 GBytes  25.6
Gbits/sec    0             sender
[  4]   0.00-300.00 sec   895 GBytes  25.6
Gbits/sec                  receiver
```

Because the server and client are on the same machine iperf3 sums the input and output traffic. Not too surprisingly 25.6 ~= 14.4 + 11.1. For a comparison with more powerful PCs and CPUs, here are additional results.

```
# In one terminal window:
~$ iperf3 -s --bind 10.10.10.24
# In a second terminal window:
~$ iperf3 -c 10.10.10.24 --bind 10.10.10.25
Connecting to host 10.10.10.24, port 5201
[  4] local 10.10.10.25 port 38783 connected to
10.10.10.24 port 5201
[ ID] Interval           Transfer     Bandwidth
Retr  Cwnd
```

```
[  4]   0.00-1.00   sec  5.56 GBytes  47.7
Gbits/sec    0   1.37 MBytes
[  4]   1.00-2.00   sec  5.61 GBytes  48.2
Gbits/sec    0   1.37 MBytes
[  4]   2.00-3.00   sec  5.64 GBytes  48.5
Gbits/sec    0   1.37 MBytes
[  4]   3.00-4.00   sec  5.60 GBytes  48.1
Gbits/sec    0   1.44 MBytes
[  4]   4.00-5.00   sec  5.65 GBytes  48.5
Gbits/sec    0   1.44 MBytes
[  4]   5.00-6.00   sec  5.55 GBytes  47.7
Gbits/sec    0   1.44 MBytes
[  4]   6.00-7.00   sec  5.61 GBytes  48.2
Gbits/sec    0   1.44 MBytes
[  4]   7.00-8.00   sec  5.58 GBytes  48.0
Gbits/sec    0   1.44 MBytes
[  4]   8.00-9.00   sec  5.52 GBytes  47.5
Gbits/sec    0   1.44 MBytes
[  4]   9.00-10.00  sec  5.66 GBytes  48.6
Gbits/sec    0   1.44 MBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
Retr
[  4]   0.00-10.00  sec  56.0 GBytes  48.1
Gbits/sec    0             sender
[  4]   0.00-10.00  sec  56.0 GBytes  48.1
Gbits/sec                  receiver
```

Note: this was performed with an industrial board running an AMD v1605b.

```
# In one terminal window:
~$ iperf3 -s --bind 10.10.10.20
# In a second terminal window:
~$ iperf3 -c 10.10.10.20 --bind 10.10.10.21
Connecting to host 10.10.10.20, port 5201
[  4] local 10.10.10.21 port 57059 connected to
10.10.10.20 port 5201
[ ID] Interval           Transfer     Bandwidth
Retr  Cwnd
[  4]   0.00-1.00   sec  11.4 GBytes  98.1
Gbits/sec    0   1.56 MBytes
[  4]   1.00-2.00   sec  11.6 GBytes  99.4
Gbits/sec    0   1.56 MBytes
[  4]   2.00-3.00   sec  11.4 GBytes  98.3
Gbits/sec    0   1.56 MBytes
[  4]   3.00-4.00   sec  11.7 GBytes   100
Gbits/sec    0   1.56 MBytes
[  4]   4.00-5.00   sec  11.5 GBytes  98.7
Gbits/sec    0   1.56 MBytes
[  4]   5.00-6.00   sec  11.5 GBytes  98.4
Gbits/sec    0   1.56 MBytes
[  4]   6.00-7.00   sec  11.5 GBytes  98.4
Gbits/sec    0   1.56 MBytes
[  4]   7.00-8.00   sec  11.8 GBytes   101
Gbits/sec    0   1.56 MBytes
[  4]   8.00-9.00   sec  11.4 GBytes  98.1
Gbits/sec    0   1.56 MBytes
[  4]   9.00-10.00  sec  11.5 GBytes  98.7
Gbits/sec    0   1.56 MBytes
- - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
Retr
[  4]   0.00-10.00  sec   115 GBytes  98.9
Gbits/sec    0             sender
[  4]   0.00-10.00  sec   115 GBytes  98.9
Gbits/sec                  receiver
```

Note: this was performed with a host running an i5 9660K (the motherboard is an ASRock Z390M-ITX/ac, memory is at 3200MT/s) and over FDR.

Not too surprisingly, the more money you throw at something, the best result you get… usually. But the point here is that the ODROID-H2 can do up to 14.4Gbe. Not bad for a $111 embedded board. Let's now see it behaves as an NSF server.

Using NFS over InfiniBand

To prepare the volumes to be shared, let's partition and format the 1 TB SSD (/dev/sda).

```
~$ sudo parted /dev/sda
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list
of commands.
(parted) mklabel gpt
(parted) unit tb
(parted) mkpart primary 0.00 1.00
(parted) align-check optimal 1
1 aligned
(parted) q
Information: You may need to update /etc/fstab.

~$ sudo mkfs -t ext4 /dev/sda1
mke2fs 1.44.6 (5-Mar-2019)
Discarding device blocks: done
Creating filesystem with 250050816 4k blocks and
62513152 inodes
Filesystem UUID: 48956888-5174-45bb-89d7-
c287615650b8
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200,
884736, 1605632, 2654208,
```

```
  4096000, 7962624, 11239424, 20480000, 23887872,
71663616, 78675968,
  102400000, 214990848

Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting
information: done
```

Let's partition and format the 10 TB WD Red (/dev/sdb).

```
~$ sudo parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list
of commands.
(parted) mklabel gpt
(parted) unit tb
(parted) mkpart primary 0.00 10.00
(parted) align-check optimal 1
1 aligned
(parted) q
Information: You may need to update /etc/fstab.

~$ sudo mkfs -t ext4 /dev/sdb1
mke2fs 1.44.6 (5-Mar-2019)
Creating filesystem with 2441608704 4k blocks and
305201152 inodes
Filesystem UUID: 5f3bb840-7d13-4052-8444-
42f9c55c9abf
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200,
884736, 1605632, 2654208,
  4096000, 7962624, 11239424, 20480000, 23887872,
71663616, 78675968,
  102400000, 214990848, 512000000, 550731776,
644972544, 1934917632

Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting
information: done
```

Check whether or not these new volumes can be mounted without problem:

```
# Create the mount points
~$ sudo mkdir /mnt/ssd
~$ sudo mkdir /mnt/hdb

# Mount the volumes
~$ sudo mount /dev/sda1 /mnt/ssd
~$ sudo mount /dev/sdb1 /mnt/hdb

# Are they mounted?
~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
.../...
/dev/sda1       938G   77M  891G   1% /mnt/ssd
/dev/sdb1       9.1T   80M  8.6T   1% /mnt/hdb
# Yep!
```

Add these volumes to /etc/fstab using their UUID:

```
# Fetch the UUID of /dev/sda1 and /dev/sdb1
~$ sudo blkid
.../...
/dev/sda1: UUID="48956888-5174-45bb-89d7-
c287615650b8" TYPE="ext4" PARTLABEL="primary"
PARTUUID="b660dfe2-1a2f-4238-a460-3eabbcb39c23"
/dev/sdb1: UUID="5f3bb840-7d13-4052-8444-
42f9c55c9abf" TYPE="ext4" PARTLABEL="primary"
PARTUUID="3b46613a-302f-4cd0-91bc-80cdd6f81a41"

# Edit /etc/fstab
~$ sudo vi /etc/fstab
# To add these two lines:
.../...
UUID=48956888-5174-45bb-89d7-c287615650b8
/mnt/ssd        ext4    defaults  0  0
UUID=5f3bb840-7d13-4052-8444-42f9c55c9abf
/mnt/hdb        ext4    defaults  0  0
.../...

# Unmount the volumes previously manually mounted
~$ sudo umount /mnt/ssd
~$ sudo umount /mnt/hdb

# Mount all volumes according to /etc/fstab
~$ sudo mount -a

# Are our two new volumes mounted?
~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
.../...
/dev/sda1       938G   77M  891G   1% /mnt/ssd
/dev/sdb1       9.1T   80M  8.6T   1% /mnt/hdb
# Yep!
```

Setting up NFS (server side)

Let's install the NFS server package on the H2:

```
~$ sudo apt update
~$ sudo apt upgrade
~$ sudo apt install nfs-common nfs-kernel-server
```

Configure the exports (adding the two lines shown below):

```
~$ sudo vi /etc/exports
.../...
/mnt/ssd/nfs    *(rw,sync,no_subtree_check)
/mnt/hdb/nfs    *(rw,sync,no_subtree_check)
.../...
```

Let's set the permissions without bothering at all for security at this point (we're testing!):

```
~$ sudo chown -R nobody:nogroup /mnt/ssd/nfs
~$ sudo chmod -R 0777 /mnt/ssd/nfs
~$ sudo chown -R nobody:nogroup /mnt/hdb/nfs
~$ sudo chmod -R 0777 /mnt/hdb/nfs
```

Load the RDMA kernel module for NFS server over RDMA:

```
~$ sudo modprobe svcrdma
```

Restart the NFS service:

```
~$ sudo systemctl restart nfs-kernel-server
```

Once restarted, tell the NFS service to also listen on the **RDMA ports**:

```
~$ echo rdma 20049 | sudo tee
/proc/fs/nfsd/portlist
```

Note: you have to do this each time you restart the NFS service. Eventually you will want to put it all into a script.

Check whether or not it is correctly set up:

```
~$ sudo systemctl status nfs-kernel-server
  nfs-server.service - NFS server and services
   Loaded: loaded (/lib/systemd/system/nfs-
server.service; enabled; vendor preset: enabled)
   Active: active (exited) since Sun 2019-09-15
14:47:12 PDT; 18s ago
  Process: 2386 ExecStartPre=/usr/sbin/exportfs -r
(code=exited, status=0/SUCCESS)
  Process: 2387 ExecStart=/usr/sbin/rpc.nfsd
$RPCNFSDARGS (code=exited, status=0/SUCCESS)
 Main PID: 2387 (code=exited, status=0/SUCCESS)


Sep 15 14:47:12 h2a systemd[1]: Starting NFS
server and services...
Sep 15 14:47:12 h2a systemd[1]: Started NFS server
and services.

~$ sudo cat /proc/fs/nfsd/portlist
rdma 20049
rdma 20049
udp 2049
tcp 2049
udp 2049
tcp 2049
```

Setting up NFS (client side)

Let's use again an i5 9600K PC as testing client. In the text below, the name "h2a" is the host name of the ODROID-H2 over the 1 GbE network and the name "h2a.ib" is the host name of the H2 over the InfiniBand network. These names get their associated IP addresses via the /etc/hosts file setup on both machines (or a local DNS if you run one.)

Let's install the NFS client package on the i5:

```
~$ sudo apt update
~$ sudo apt upgrade
~$ sudo apt install nfs-common
```

Load the RDMA kernel module for NFS client over RDMA:

```
~$ sudo modprobe xprtrdma
```

Testing

For testing, we copied various files from the i5 to the H2:

- CentOS-7-x86_64-DVD-1804.iso, the ISO image of the CentOS7 64-bit install DVD. What matters here is the file size: 4.5GB
- Lunar_LRO_WAC_GLD100_DTM_79S79N_100m_v1.1.tif, a TIFF image of the Moon. File size: 10.5GB
- VMware-image-folder, a directory containing a Vmware image for a total size of 41GB with 41 files.

On the i5, we proceed as follows:

```
# Create the mount points:
~$ sudo mkdri h2a:/mnt/nfs/h2a/ssd
~$ sudo mkdri h2a:/mnt/nfs/h2a/hdb
# Mount the SSD remote volume from the H2 over the
```

```
1 GbE network:
~$ sudo mount h2a:/mnt/ssd/nfs /mnt/nfs/h2a/ssd
# Measure the time it takes to copy a file:
~$ time cp /home/domih/Downloads/CentOS-7-x86_64-DVD-1804.iso /mnt/nfs/h2a/ssd
# Mount the hard disk remote volume from the H2 over the 1 GbE network:
~$ sudo mount h2a:/mnt/hdb/nfs /mnt/nfs/h2a/hdb
# Measure the time it takes to copy a file:
~$ time cp /home/domih/Downloads/CentOS-7-x86_64-DVD-1804.iso /mnt/nfs/h2a/hdb
# Unmount the volumes:
sudo umount /mnt/nfs/h2a/ssd
sudo umount /mnt/nfs/h2a/hdb
# Mount again but this time over NFS RDMA
sudo mount -o rdma,port=20049 h2a.ib:/mnt/ssd/nfs /mnt/nfs/h2a/ssd
# Measure the file copy again.
sudo mount -o rdma,port=20049 h2a.ib:/mnt/hdb/nfs /mnt/nfs/h2a/hdb
# Measure the file copy again.
```

Tests for the 4.5GB file were made 3 times over NFS in sync mode and then in async mode. The test were made only once for the bigger file and directory over NFS in async mode.

To switch the NFS server from sync to async mode, we just change the shares in the /etc/exports file and restart the NFS server (without forgetting the echo rdma…):

```
~$ sudo vi /etc/exports
.../...
/mnt/ssd/nfs    *(rw,async,no_subtree_check)
/mnt/hdb/nfs    *(rw,async,no_subtree_check)
.../...
~$ sudo systemctl restart nfs-kernel-server
~$ echo rdma 20049 | sudo tee
/proc/fs/nfsd/portlist
```

Here are the results:

Copied file size: 4.5 GB — 4,470,079,488.00 bytes

	1Gbe NFS TCP/IP	NFS RDMA	1Gbe NFS TCP/IP (in seconds)	NFS RDMA (in seconds)	1Gbe NFS TCP/IP (speed in Gbe)	NFS RDMA (in Gbe)	Acceleration (x times faster)
NFS in Sync mode							
ssd	0m40.176s	0m11.629s	40.18	11.63	0.89	3.08	3.45
ssd	0m40.401s	0m11.318s	40.40	11.31	0.89	3.16	3.57
ssd	0m40.343s	0m11.627s	40.34	11.63	0.89	3.08	3.47
hard disk	0m43.555s	0m24.475s	43.56	24.48	0.82	1.46	1.78
hard disk	0m45.222s	0m25.480s	45.22	25.48	0.79	1.40	1.77
hard disk	0m42.873s	0m24.445s	42.87	24.45	0.83	1.46	1.75
NFS in Async mode							
ssd	0m39.859s	0m5.698s	39.86	5.70	0.90	6.28	7.00
ssd	0m39.855s	0m5.978s	39.86	5.98	0.90	5.98	6.67
ssd	0m39.873s	0m5.972s	39.87	5.97	0.90	5.99	6.68
hard disk	0m39.722s	0m5.825s	39.72	5.83	0.90	6.14	6.82
hard disk	0m39.895s	0m5.961s	39.90	5.96	0.90	6.00	6.69
hard disk	0m39.975s	0m5.840s	39.98	5.84	0.89	6.12	6.85

Copied file size: 10.5 GB (10461394351 bytes) — 10,461,394,351.00 bytes

	1Gbe NFS TCP/IP	NFS RDMA	1Gbe NFS TCP/IP (in seconds)	NFS RDMA (in seconds)	1Gbe NFS TCP/IP (speed in Gbe)	NFS RDMA (in Gbe)	Acceleration (x times faster)
NFS in Async mode							
ssd	1m31.047s	0m11.561s	91.05	11.561	0.92	7.24	7.88
ssd	1m32.195s	0m12.317s	92.20	12.317	0.91	6.79	7.49
ssd	1m32.165s	0m12.742s	92.17	12.742	0.91	6.57	7.23
hard disk	1m30.509s	0m24.418s	90.51	24.418	0.92	3.43	3.71
hard disk	1m32.240s	0m26.146s	92.24	26.146	0.91	3.20	3.53
hard disk	1m32.201s	0m27.365s	92.20	27.365	0.91	3.06	3.37

Copied directory size: 41GB — 43,812,100,842.00 bytes

	1Gbe NFS TCP/IP	NFS RDMA	1Gbe NFS TCP/IP (in seconds)	NFS RDMA (in seconds)	1Gbe NFS TCP/IP (speed in Gbe)	NFS RDMA (in Gbe)	Acceleration (x times faster)
NFS in Async mode							
ssd	6m32.864s	1m16.595s	392.864	76.595	0.89	4.58	5.13
hard disk	6m33.159s	3m15.189s	393.159	195.189	0.89	1.80	2.01

Figure 28 - Testing results

The formula to compute the speed in Gbe is:

```
File Size in Bytes x 8 / Time in secs / 1000000000
= Speed in Gbe
```

What to conclude from these numbers

- The resources (CPU) and time consumed writing to the SSD or hard disk do not affect the speed of the 1 GbE Ethernet because the network speed is slow enough to cover it up. On the other hand they do affect the 10+ GbE interconnect. You particularly see this comparing the NFS RDMA results in sync mode between the SSD and the hard disk. This effect disappears after switching NFS RDMA to async mode. Is there a "danger" penalty to switch to async? Not really as long as your server is behind a UPS and does not fail, meaning I/O error on the disk or crash. Both Windows and Linux in a certain extent cache disk writing anyway so in case of failure, the story is the same. For a longer discussion, see **https://www.google.com/search?q=nfs+sync+vs+async**.
- The acceleration varies from a little less than x2 to close to x8. The acceleration varies depending on what you copy and where you copy it. In any case, not bad at all for an embedded board that costs slightly more than $100.
- Beyond the numbers the human psychological effect is more about how long you wait for an operation: 1m16 is definitely much faster than 6m32! With the former you wait, with the latter you go grab a coffee.
- These are realistic numbers without time spent on optimizing. See Postscript below.

To play with unrealistic numbers, let's use a RAM disk on one of the machines (here the i5) and copy from the H2 to the I5:

```
sudo mkdir -p /mnt/ramdisk
sudo mount -t tmpfs -o size=20480M tmpfs
```

```
/mnt/ramdisk

time cp /mnt/nfs/h2a/ssd/CentOS-7-x86_64-DVD-
1804.iso /mnt/ramdisk
4470079488*8/3.338/1000/1000/1000 = 10.71 Gbe

time cp
/mnt/nfs/h2a/ssd/Lunar_LRO_WAC_GLD100_DTM_79S79N_1
00m_v1.1.tif /mnt/ramdisk
10461394351*8/7.385/1000/1000/1000 = 11.33 Gbe
```

Yeah, we are back above the 10Gbe mark! So if you really want to pump up the numbers, use a RAM disk (on the server and/or the client), the amount of data you can transfer will be limited to the size of the RAM disk (here 20GB) and you will have to write a CRON job or something like that to eventually copy the RAM disk to the physical disk on the server and/or the client for actual long term persistent storage.

Postscript

We hope you enjoyed reading this article. As mentioned earlier, remember that I am not an embedded computing nor and InfiniBand expert. If you are, do not hesitate to send your feedback to the Odroid Magazine or forum because I and others like to learn more about new things. Due to time and resource restrictions as well as reasonable article length for a magazine publication, I did not address the following topics:

- Possible CPU affinity, IRQ affinity and TCP/IP optimizations,
- Use LVM2 to create a cached logical volume for optimizing disk I/O,
- Samba usage over this type of interconnect,
- Windows Server 2012 (or later) and Windows 10 integration with SMB direct,
- Alternate solutions using 10 GbE Ethernet SFP+ cards and switches (SFP+ is significantly less expensive than their RJ-45 cousins as of this writing),
- Other technologies leveraging RDMA, such as distributed file systems (Hadoop, Lustre, Ceph, GlusterFS...), distributed databases, etc.

Depending on interest, there might be follow-up articles by me or others. Do not hesitate to express your interest to the Odroid Magazine.

About the author

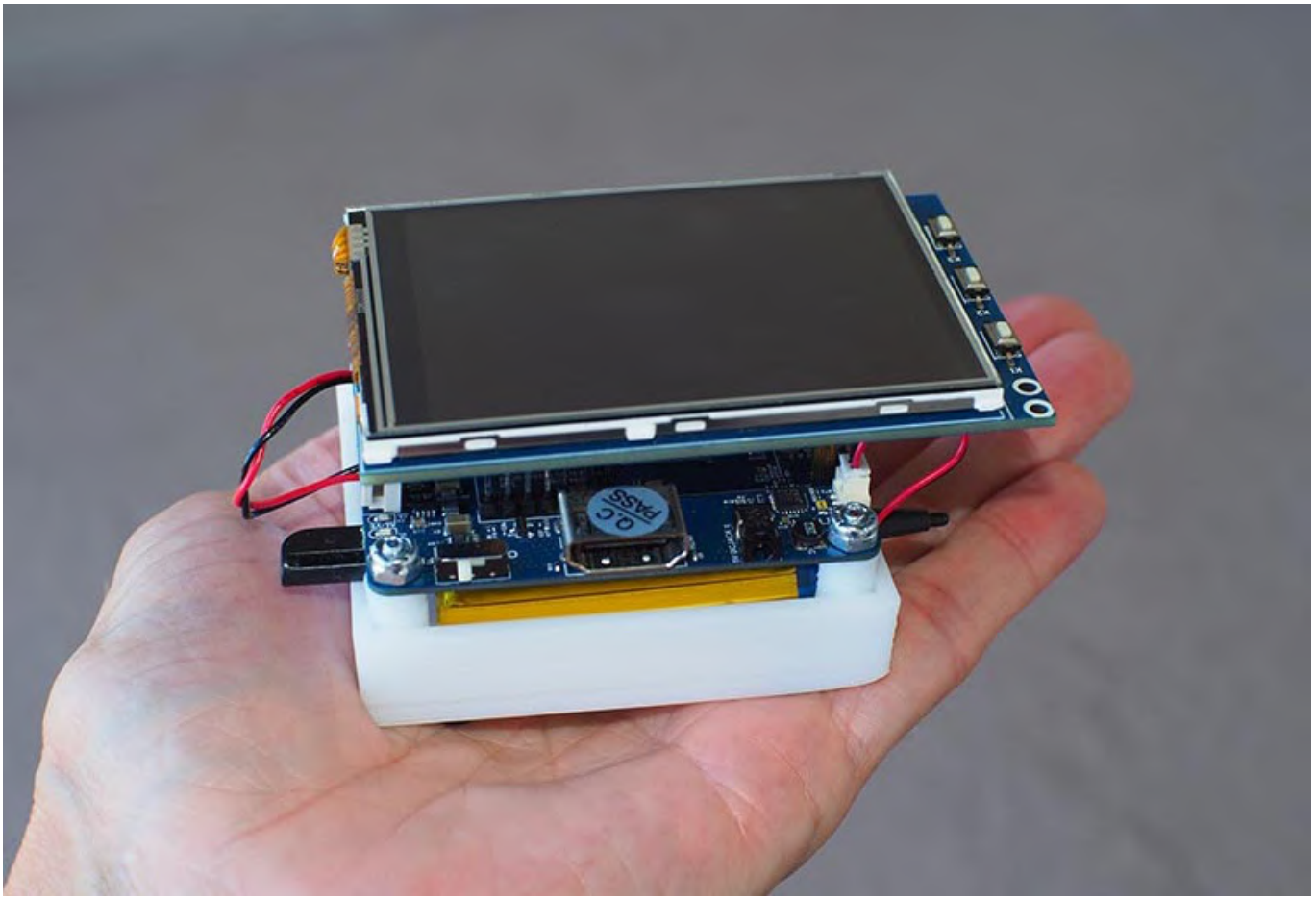
Figure 29 - Dominique when he was younger. We used this image as the opportunity to show how cool the fashion for little boys in France early 60s was

Dominique has grown up since the above image was taken, and has been working as CTO or SVP of Engineering in multiple start-ups located around the San Francisco bay area (Silicon Valley) for the last 30 years. Dominique is currently SVP of Engineering at bonafeyed.com a start-up just getting out from under the radar. Its main product, Cy4Secure, is a data security solution that protects data or information when it is shared with others, through or in the cloud or an application, keeping the data always encrypted while on the move as well as at rest. Cy4Secure represents the last line of data protection after a

security breach. Dominique is also consulting for mkrypt.com which is working on a Bluetooth low-energy fob providing true voice encryption for mobile and smartphones. You can reach Dominique on the ODROID forum by messaging @domih.

Is That a Linux Computer in Your Pocket, or Are You Just Glad to See Me?: Build an ODROID Computer You Can Carry in Your Pocket

October 1, 2019 By Dave Prochnow ODROID-C0, Tinkering



Fresh on the heels of the ODROID Tablet project, https://magazine.odroid.com/article/build-a-rootin-tootin-dual-bootin-odroid-tablet-using-the-odroid-c0-to-make-a-professional-grade-tablet-for-under-usd100/, comes an even more portable version of the ODROID-C0. Rather than sporting a large-scale HDMI-equipped LCD, this "pocket 'puter" relies on a framebuffer-driven video output displayed on a 3.2-inch thin-film-transistor (TFT) touchscreen shield dubbed the C1.

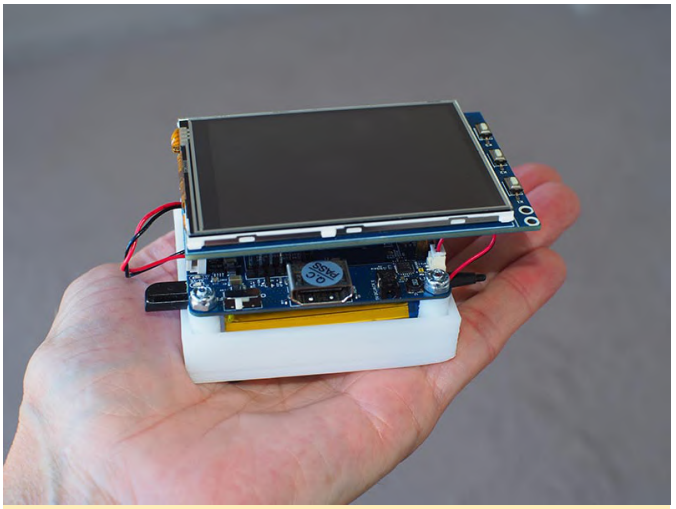


Figure 1 – A "pocketable" palm-sized Linux computer.

While this touchscreen shield is designed as a simple "plug-n-play" peripheral, the software setup for enabling the ODROID-C0 to use this display can be a daunting task. Luckily, Hardkernel has created a

series of articles for enabling you to setup this display quickly and easily. Just religiously follow the Hardkernel wiki steps, while adhering to the following assembly steps, and you will have your very own pocket full of computing pleasure for less than $65.

Parts

- ODROID-C0 $28.00
- C1 3.2-inch TFT + Touchscreen Shield $25.00
- Connector Pack for ODROID-C0 $1.80
- 3000 mAh Battery $1.00
- RTC Backup Battery $2.50 [Optional]
- microSD Card 32GB $5
- Also, you will temporarily need access to an HDMI monitor and USB keyboard and mouse for programming the ODROID-C0.

Step-by-Step

1. Prepare the ODROID-C0 PCB with several components from inside the Connector Pack for using the C1 shield. Solder the dual stacked USB connector to the PCB. Snap off 26 pins, two rows of 13 pins, from the ODROID-C0 dual row header. Solder this header onto the ODROID-C0 GPIO connector starting from pin 1.

(

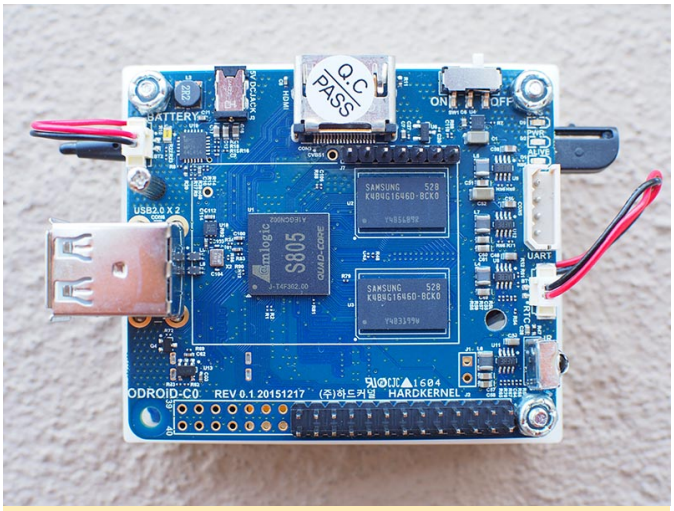


Figure 2 - All of the soldering is now complete. Notice that the dual row header has only been soldered to 26 GPIO pins.

2. Connect the 3000 mAh Battery and the optional RTC Backup Battery to the ODROID-C0.

3. Slide the C1 Shield female headers onto the ODROID-C0 headers you soldered in Step 1.

4. Temporarily connect the ODROID-C0 to an HDMI monitor and USB keyboard and mouse.

5. Ensure that the ODROID-C0 is being powered by a 5V 2A power supply. The PCB's green battery charging LED may or may not be lit during the programming of the ODROID-C0.

6. Follow the Hardkernel Wiki steps for programming the ODROID-C0:

http://wiki.odroid.com/accessory/display/3.2inch_tft_touchscreen_shield/start

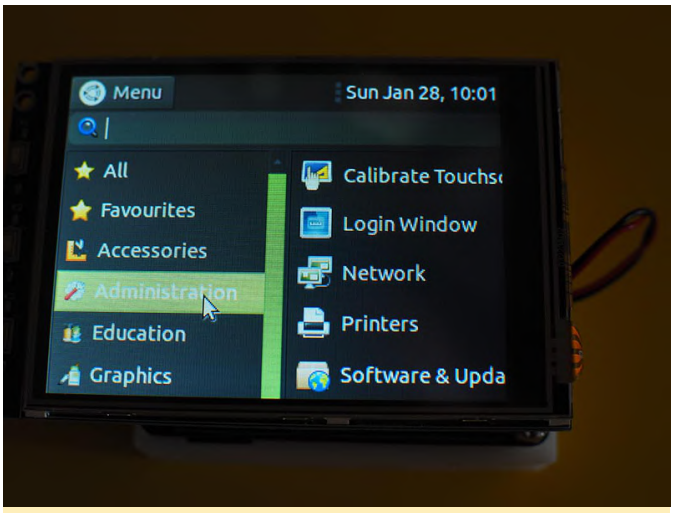


Figure 3 - After you've completed the Hardkernel C1 shield wiki, look for the touchscreen calibration app inside the Ubuntu menu. Run this app for finalizing the touchscreen shield setup.

7. Disconnect the HDMI monitor and reboot the ODROID-C0. Keep the USB keyboard and mouse plugged into the ODROID-C0. After a one to two minute delay, the mouse cursor will appear on the Touchscreen Shield followed by the Log-in screen.

Figure 4a - The Ubuntu Log-in screen—just click it and you're ready to go.

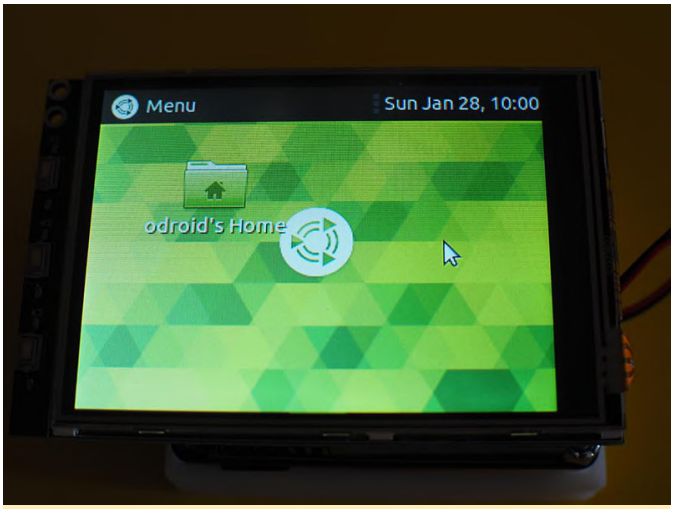


Figure 4b - There isn't a lot of screen real estate on the touchscreen shield. Use the Notes (i.e., numbers 3 and 4) below for maximizing your little screen's potential.

8. Use the USB keyboard and mouse for configuring your new pocket-friendly ODROID. Possible uses for this Pocket 'Puter are a portable event timer/stopwatch/clock, a family calendar, a mobile download device, and a personal media player.

Figure 5 - Either ready for the local coffee shop WiFi hotspot or resting attentively at your local family household wifi hotspot, this petite 'puter powerhouse can handle it.

Notes

1. The large ground, GND, plane on the ODROID-C0 PCB can make soldering extremely difficult. Try using an adjustable-temperature soldering station with a temperature setting of about 380-degrees C.

2. Here's a Pro Tip for beginning DIYers: never solder more male header pins on a PCB than you plan on using. For example, in this project only two rows of 13 pins were used. If all 40 pins of the ODROID-C0 connector were soldered onto the PCB, some power pins will be exposed (e.g., pin 38 1.8V and pin 39 GND) which could result in a short circuit and damage your beloved ODROID.

3. Add a shutdown button to the top panel of the Desktop window. You can add this button by performing a right click with your mouse while the cursor is inside the top panel.

4. Make two modifications to the Preferences of the File Management app: first, change the behavior of the user interface to using "one click" for accessing files and folders. Second, decrease the magnification of the standard view to 50%.

5. The title for this article is a paraphrase quote of Mae West from the movie "Sextette" (1978).

The G Spot: Your Goto Destination for all Things That are Android Gaming

October 1, 2019 By Dave Prochnow Android, Gaming

Based on the game releases announced at gamescom 2019, the future for Android gaming is very bright. As you'll recall, gamescom, organized by Koelnmesse and game – the German Games Industry Association, is the self-proclaimed 'world's largest trade fair and event highlight for interactive games and entertainment'. This year it was held in Cologne, Germany between August 20 and August 24, 2019. There were games, video highlights, attendants in cosplay, and something big, very big. That "something" was Sony.

Figure 1 - gamescom 2019 finally pulled the curtain back on Google's Stadia streaming game subscription service.

Other than the raft of future game titles that were announced, the absolutely biggest eyebrow-raising occurrence had to be Sony Interactive Entertainment winning the coveted Best of gamescom award for its creation of the sandbox game, Dreams. You'll remember that Sony was a glaring "no-show" at E3 Expo 2019. That snub of the Los Angeles event didn't stop them from hogging Hall 7 in Cologne and walking away with a fist-full of awards including the mentioned "Best of gamescom" along with "Best VR/AR Game" (Marvel's Iron Man), "Best Family Game" (Concrete Genie), "Most Original Game" (Dreams, again!), and "Best Sony PlayStation 4 Game" (you guessed it, Dreams).

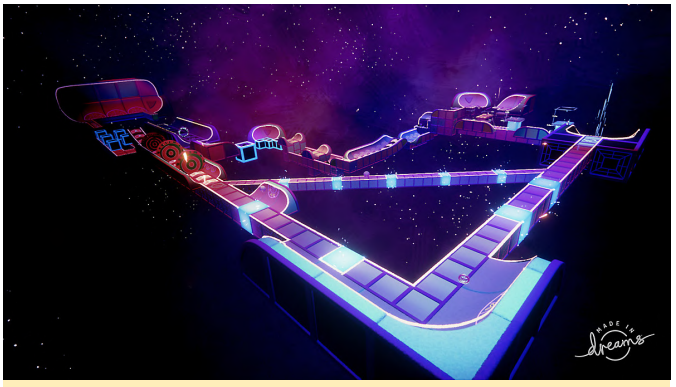


Figure 2 - No, you're not dreaming, Sony Interactive Entertainment Dreams title is a worthy winner of the "Best of gamescom" award. Image courtesy of Sony Interactive Entertainment.

Sony wasn't the only big news at gamescom, there was also that little bit of news about Android gaming. Actually, this news isn't truly about Android games, per se, it's ALL about Google Stadia! Stadia is for Android as well as iOS, PC Windows, MacOS, and, can you believe it, Linux. If your platform of choice has Google Chrome or Chromium, then you can be a player. YAY!

Costing $9.99 per month, Stadia Pro,i.e., the version included in the $129 Founder's Edition AND including regular FREE game releases, is scheduled for launch in November of this year. According to a little rumor I heard, Google Stadia will come in a FREE version (no monthly fee) sometime in 2020. There are no FREE games, however this version is limited to 1080p.

So what games can we expect from Google Stadia? At gamescom, a complete roster of titles was released. Here is a large snippet from that long, long roster:

Bethesda – DOOM Eternal Bungie – Destiny 2 [this title could be a FREE release in November for Founder's Edition members] CD PROJEKT RED – Cyberpunk 2077 [rave reviews from its release at E3 Expo 2019] Dotemu – Windjammers 2 Larian Studios – Baldur's Gate 3 Pandemic Studios – Destroy All Humans! Robot Entertainment – Orcs Must Die 3 SEGA – Football Manager Square Enix – Final Fantasy XV Square Enix – Marvel's Avengers 2K – NBA 2K Warner Bros. - Mortal Kombat 11 Ubisoft – Tom Clancy's Ghost Recon Breakpoint Ubisoft – Just Dance [an ODROID Magazine staff favorite]

The Google Stadia Twitter account has a video of gamers playing Stadia:

https://t.co/WeBuVEUYVb

If you're like me, I'm sure that you are now saying, 'November can't get here fast enough.' And, 'where's my Founder's Edition subscription?'

Breaking News

Just as this issue was going into production, two news items landed on my desk. First, in honor of the id Software 25th anniversary of the release of DOOM (in 1993), Bethesda has launched both DOOM and DOOM II on Google Play for $4.99 each. My how times change, right? Remember back at the time of the 20th anniversary, the DOOM app release was FREE!

Second, Nintendo just announced that Mario Kart Tour will debut for Android on September 25. That's right; this title was "supposed" to launch in March 2019. After some reworking of the code, however, this

mobile edition should finally be release-worthy by the time you read this article.

Some of you will no doubt be shocked to learn that Mario Kart Tour will be a FREE app download. But don't hold your breath too long, the game will be supported through in-game purchases or microtransactions. It remains to be seen how well this payment system will coexist with the game's play mechanics.

Low Cost Water Cooling for your Single Board Computer: Get The Maximum Speed From Your ODROID

October 1, 2019 By Edward Kisiel ODROID-XU4, Tinkering

SBC water cooling is not new and others have implemented designs using off the shelf components for the ODROID-XU4 and other SBC. Some implementations have already been covered in ODROID Magazine in the past.

December 2016 **https://magazine.odroid.com/wp-content/uploads/ODROID-Magazine-201612.pdf**

July 2018 **https://magazine.odroid.com/article/liquid-cooling-part-1-cluster/** **https://magazine.odroid.com/article/liquid-cooling-part-2-server/**

The focus of this project was initially to make water cooling low cost and economical for the ODROID-XU4. After working on the water block design, I discovered a way to support a wide range of other SBCs regardless of their ability to support a proper heatsink. This development was the auspice for the SBC Model Framework that I completed earlier. **https://forum.odroid.com/viewtopic.php?f=53&t=33823** Now any SBC supported by the SBC Model Framework can utilize this design to provide a universal low cost water block for a liquid cooled system.

The past water cooled implementations that I have seen used components from the INTEL/AMD desktop arena that were larger and had more capacity than necessary for SBC's and, therefore, typically had a relatively high cost.

Cooling systems that cost more than the SBC it is cooling are not cost effective or economically justifiable. They are fine for research and special one-off uses but not for widespread adoption. So the first question I had to answer when I started this project was how much cooling cost is economically justifiable?

A lot can be said and has been said on this subject, but for me I arrived at a value of no more than 20% of the total system cost or about the cost of a reasonable heatsink and fan. When I built my ODROID-XU4 cluster, the cost per node, including power supply, SD card and networking, was approximately $73. I wondered if it was possible to build a water cooled system for $14 or less. I already knew it was not possible, if using off the shelf components, so I had to get creative designing new components while looking to re-purpose components from other industries.

There are a host of problems that have to be solved in order to reach all of the goals set out for this project. This article deals specifically with the first phase of the project, a universal water block and attachment method with low cost fittings and pump. The heat exchanger and final packaging of the complete cooling system will be addressed in the second phase. This project is also the first step toward my ultimate goal to design a low cost, scaleable water cooled ODROID-XU4 cluster. I'm using a single SBC to work through many of the design issues prior to a cluster implementation.

After experimenting with several approaches for the water block design, and in consideration that some SBC's do not have a practical way to attach a heatsink, I started to experiment with a water cooled case that could accommodate different size water blocks and a universal attachment method.

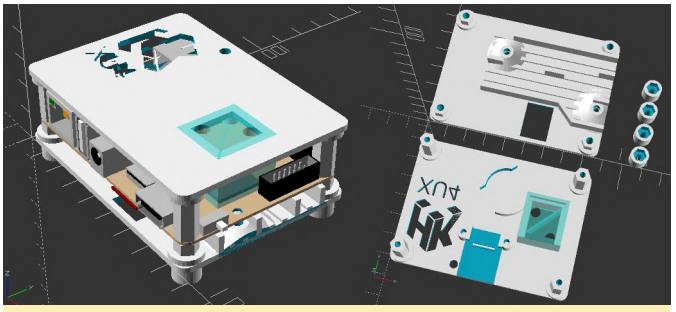


Figure 1 - ODROID-XU4 SBC Water Cooled Case Design with RTC and UART holder

By integrating the water block directly into the case, this approach provided both a way to accommodate most SBC layouts and served as a solid attachment for the water block. The water block could easily be made to any size, shape and allow multiple water blocks on either side of the PCB. With the increasing trend of additional hardware on SBC's to handle A.I., networking or other specialized processing, I felt it would be beneficial to provide the means to water cool them, as well. It could also provide a way to cool multiple memory chips. A 1/8"(3.19mm) thick piece of copper inserted into the water block is used to transfer heat from the SOC to the cooling media.

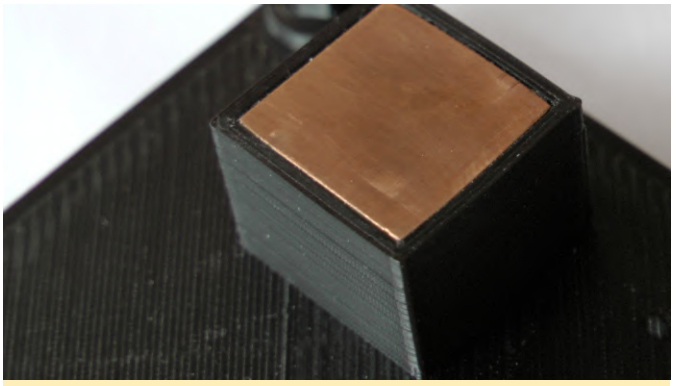


Figure 2 - ODROID-XU4 Water Block

Water Cooled Case Design

Some of the additional features of the SBC Water Cooled Case design include:

- Universal SBC Support
- Up to 4 water blocks, top and/or bottom
- Integrated and/or user provided standoffs
- Blind or thru bolted case top with or without countersink
- Predefined accessories (UART holder, RTC holder, ODROID-XU4 case bottom support, artwork, multi-shape fan or cable holes, etc)
- User defined additive or subtractive accessories

Issues and Tips

There have been 3 main issues that I have had to work on to bring this design forward. The first was finding fittings that were inexpensive, small and strong enough to work adequately. Fittings from INTEL/AMD liquid cooled systems are too large for most SBC SOC. Two of them will not fit within the size of many SOCs. Being familiar with drip irrigation systems, I decided to use drip irrigation barbs that are both strong, small, can be easily glued in place, and come in straight and 90 degree variations.

https://www.digcorp.com/homeowner-drip-irrigation-products/1-4-barbed-fittings

For testing I needed a strong but reversible method to attach them for reuse. After trying several glues, Cyanoacrylate(Super Glue) seemed to work best. I cut the barb off one end and glued them in place. It was strong enough to hold under considerable stress and the barbs could still be freed due to the glue's brittleness. Another stronger and permanent glue or ABS acetone weld may be used for the final production. It may even be possible to tap one end of the barb for a threaded solution.

The second issue was finding a suitable pump that was inexpensive, was rated for continuous duty, had an adequate flow rate, and ran on 5 volts. After searching in both the medical and food industries I was able to acquire for $5(delivered), an ET-Tech series 23 5v 1.5w micro pump rated for continuous duty. It also had the benefit of being very quiet. http://www.et-pump.com/brushless_23.html

The third issue to solve had to do with producing the case. The weak point of the design has to do with the manufacturing of the water block using 3D printing technology. Each layer of the water block was a potential water leak. On top of this, the ODROID-XU4 SOC location provided a unique challenge due to its close proximity to the 12 pin GPIO header. Because of the required water block wall thickness, the water block did not have the proper clearance for the connector. None of the other SBCs I tested had this issue, it was specific to the ODROID-XU4. To make things worse, it was close to working but if you forced the assembly of the case, the pressure bowed the top of the case slightly and ultimately created enough pressure to open micro cracks in the print layers that eventually leaked.

Over several months of working on this issue I could not solve it and set the design aside. The design still worked for other SBC but the ODROID-XU4 was the main reason I wanted this solution. After many more attempts to address what I considered a major design issue, I realized that the housing for the 12 pin GPIO header could easily be slide off the pins which allowed enough room for the water block to make proper contact with the SOC. Generally speaking, my overall design approach is never to make permanent modifications to the SBC. I felt this solution was a reasonable compromise since the housing could be easily re-installed. Please note, due to the minimization of solder used in the manufacturing of the PCB and lack of support with the housing off, it is very easy to pop a pin loose, so be careful if you remove the GPIO header housing.

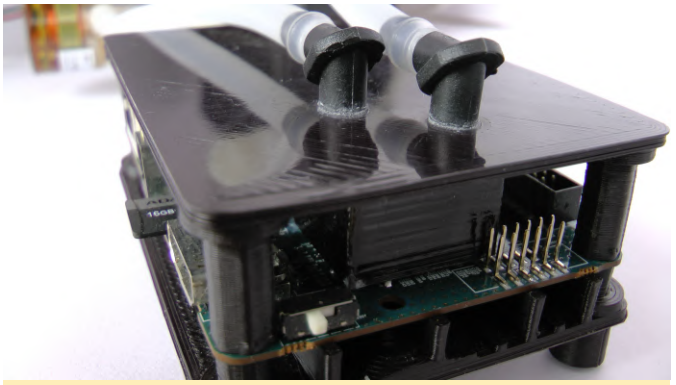


Figure 3 - ODROID-XU4 case closeup

To add some assurance that micro leaks would not be an ongoing problem, I also developed a technique to strengthen the whole water block. Using a cotton swab and acetone, I dipped the swab in the acetone and then rubbed it back in forth across each side of the water block and corners. I repeated this process 2-3 times per side so that the ABS melted and formed a continuous bond across the face, significantly minimizing the chance of any micro cracks forming in the water block. I have not had any leaks using this method. If the case was manufactured with an injection mold, this problem would not exist. It is specifically due to the layered manufacturing process of 3D printing that this issue had to be solved.

One other tip worth mentioning is to dip the 1/4" tubing in boiling water to form any needed curves so that connections are not under stress once assembled. It only takes a few seconds exposure to soften the plastic enough to form any shape you might need and the shape is permanent once the plastic cools. It also makes it easier to fit the tubing over the barbs.

Beta Release of Design

I'm at a point where I can make a beta release of the design even though the heat exchanger is not complete. A lot of radiators incorporate a pump in the radiator so anyone that has an old one from an

INTEL/AMD system could create an inexpensive water cooled SBC using this current beta design.

It is not practical to test every SBC and case configuration; it can also include air cooled versions. It has been several months since I did multiple SBC test prints and it was restricted to the ODROID SBC that I owned. No operational testing was conducted at that time.

Figure 4 - ODROID Water Cooled Cases (MC1, N1, C2 and XU4)

Since then, I have completely rewritten the OpenSCAD case algorithm and added new features. The ODROID-N2 was released and the ODROID-H2 became available again so be aware and please provide any feedback you have if you tried one of these SBC, not that the ODROID-N2 needs water cooling, or a new one.

SBC Water Cooled Case Design Files

The SBC Model Framework is needed and its directory should be installed in the same parent directory as the SBC Water Cooled Case directory. If you would like to use a different directory structure the include and use statements in sbc_water_cooled_case.scad need to be changed accordingly.

```
use <../sbc_models/sbc_models.scad>
include <../sbc_models/sbc_models.cfg>
```

The SBC Model Framework can be acquired in the ODROID forum at
https://forum.odroid.com/viewtopic.php?f=53&t=33823

Initial Water Block Testing

I did not know how a reduced size water block was going to perform. I do know the ODROID-XU4 has a relatively small SOC, is one of the most challenging SBC to cool when running at 2Ghz, and if It worked. This success bodes well for adoption with most other SBCs, too. After over a year and a half of working on this project I finally arrived at a point to do the initial testing and find out for sure if I was on a workable design. It is not comprehensive testing, that portion of this project will come after the heat exchanger is incorporated into the system.

Test Bed and Operational Parameters

120mm copper radiator and 120mm fan was used for the water block testing using the HK Minimal Ubuntu 18.04 image. All tests were conducted with the A15@2ghz, A7@1.5ghz, memory at 933mhz with Performance CPU and GPU governor settings.

```
$ uname -a
Linux c2n1 4.14.127-164 #1 SMP PREEMPT Wed Jun 19
17:28:22 -03 2019 armv7l armv7l armv7l GNU/Linux
```

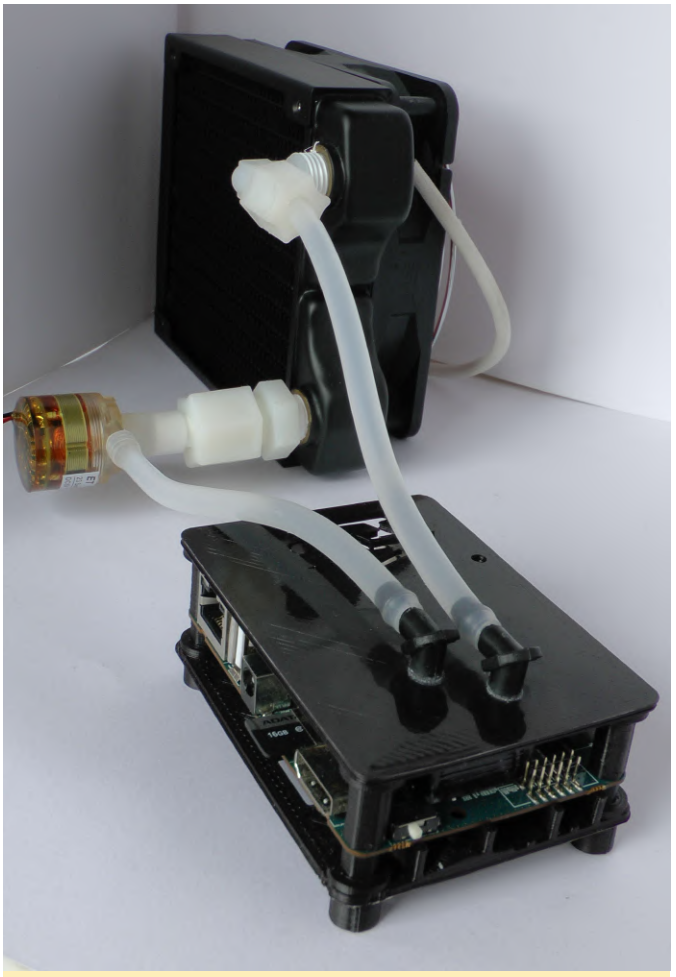


Figure 5 - Waterblock Test Bed

```
Kernel compile at 2ghz @ 71F(21.66c) Ambient
Temperature
real 25m12.282s
```

```
user 172m3.503s
sys 16m48.678s
```

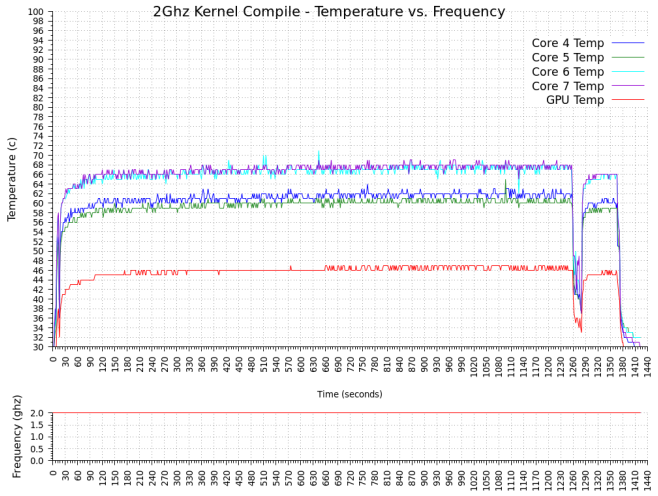


Figure 6 -Kernel Compile Test 2ghz

I found it interesting that the compile time was slightly better than the ODROID-H2 cross compile time demonstrated during a recent review at cnx-software.com. https://www.cnx-software.com/2019/07/14/odroid-h2-review-ubuntu-19-04/ The author compared the ODROID-H2 cross compiling of a ODROID-XU4 kernel to an ODROID-XU4Q native compile time. The ambient temperature during that test was higher and unfortunately I don't have an ODROID-H2 so I cannot duplicate the test to compare the performance with the same ambient temperature. One other note regarding the kernel compile test; after the test completed, I realized that I had been running on a stock kernel that had transparent huge pages enabled. I'm not sure if this helped or hurt the completion time but since it was the thermal characteristics I was interested in I didn't worry about it. The BOINC test used the same kernel.

Boinc Test 8 threads @ 72(22.22c) Ambient Temperature

BOINC Universe@home 8 thread workload was used for the following tests at 2ghz, 1.9ghz and 1.8ghz for approximately 1 hour each. The same workload was used by pausing BOINC processing until the SOC cooled down, the clock frequency was adjusted and then the BOINC workload was resumed. As evident from the chart below, two threads finished approximately 10 minutes early during the 1.8ghz test. I believe the results are still valid when considering there was no change in the last 10 minutes of the 1.9ghz or 2.0ghz tests. Looking at the drastic temperature difference once the 2 threads completed, I initially assumed they were both running on A15 cores but after examining the test data closer I couldn't verify that conclusion because the temperature across all the A15 cores dropped proportionally. This would seem to indicate that it could have been two A7 cores that completed. I would not have expected such a decrease and uniformity in A15 core temperatures if that was the case. I have not done any other analysis or testing to confirm which was the case so it is still undetermined. For simplicity, the chart is of the hottest A15 core for each frequency tested.

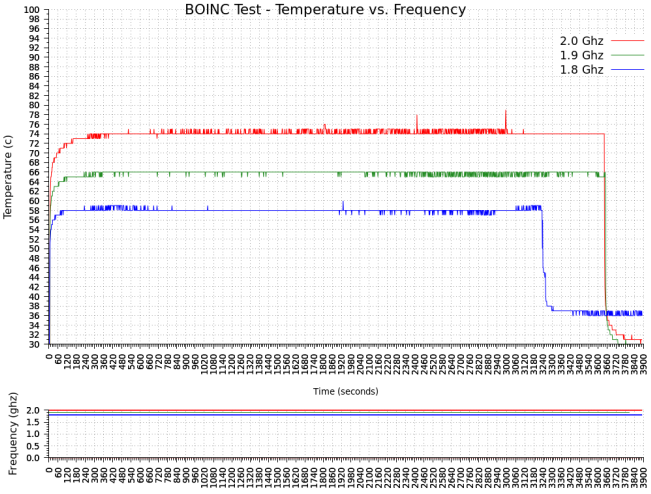


Figure 7 - BOINC Test at 2ghz, 1.9ghz and 1.8ghz

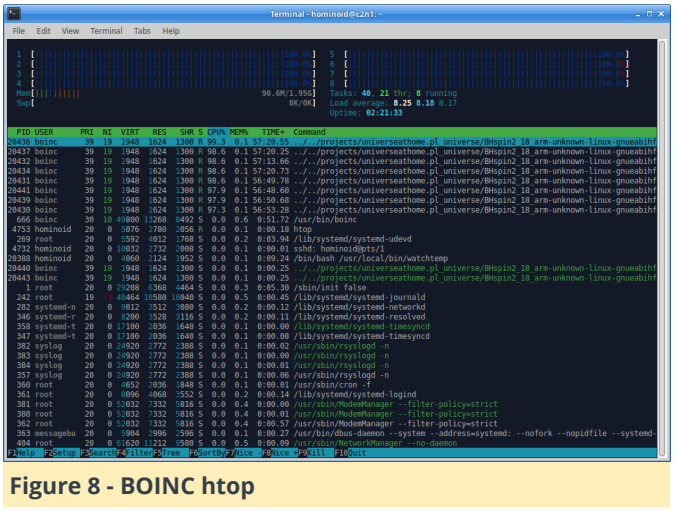


Figure 8 - BOINC htop

Summary

The kernel compile test was able to maintain approximately 68C and the BOINC loads also performed well at the test frequencies. Some small improvements in thermal performance may be possible with the addition of a fluid with better

thermal absorption properties, varying flow rate or by fine tuning flow patterns through the water block. I look forward to finishing the heat exchanger design so more comprehensive testing can be done with the complete system.

XU4 Current BOM	Cost (USD)
1/4" Barbs, ABS $.09 x 2	$0.18
1/4" Tubing 1'	$0.15
1/8" Copper 15.5mm x 16.75mm	$0.38
Pump 5v 1.5w .5 L/min	$5.00
ABS Plastic Filament, 30g	$0.34
Glue and Electricity	$0.25
Sub-Total	$6.30

Of the $14 budget, $7.70 remains for the heat exchanger. I have several designs I'm currently considering. Even though the current pump has performed well, I have also been exploring other options to reduce its size and cost. If this project is successful, SBC water cooling may become economical and expand the thermal envelope by allowing an SBC to run unthrottled in higher ambient temperatures while increasing real world performance of SOC's using older fabrication processes.

The OpenSCAD design files, test data and gplot scripts are available in the forum at **https://forum.odroid.com/viewtopic.php?f=98&t=35751**.

Five Minute Fun with your Monku R1: Retro Cubicle Commando

October 1, 2019 By Brian Ree Gaming, ODROID-C1+, ODROID-C2, ODROID-XU4

Requirements

- A Monku Retro 1,2,3 / ODROID-C1+,2,XU4 (It is expected these devices are configured with Ubuntu and MATE. Check the references below for R1, R2 devices and R3 devices.)
- A USB Audio Card x1
- A Wireless GameSir Controller x1
- An ODROID USB WiFi Adapter x1

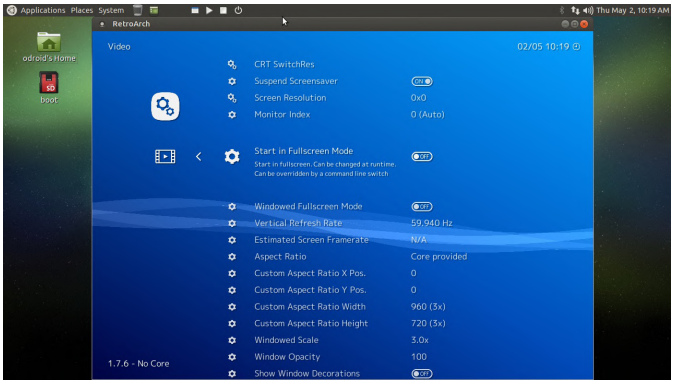
Introduction and Tutorial Goals

We all get bored at work and sometimes we have an opportunity to kill some time. This tutorial shows you how to convert your Monku Retro console into a Retro Cubicle Commando! The perfect console for gaming discreetly on a VGA screen. The steps below will show you how to adjust the scripts on your Monku R1, R2, or R3 (ODROID-C1+, -C2, -XU4) so that using the custom control button to switch to VGA mode automatically adjusts the retroarch config file to use the USB audio card. The R3, ODROID-XU4, does not use custom control scripts but you can just run the scripts that alter retroarch's audio configuration by hand. You will need a Monku Retro device, like the one we showed you how to build R1, R2 & R3 devices, a USB audio card, a USB WiFi adapter, and a wireless gamepad for stealth - links provided earlier. We will use earbuds for audio and wireless wherever possible to keep a low profile when gaming during cubicle downtime. Follow the rest of the steps to create a Retro Cubicle Commando of your own. First, a little bit about the parts needed. The wireless GameSir gamepad is not really required. You can get away with using the wired version for about half the price. Hardkernel has a very good price on that model so be sure to check it out when you're shopping for parts. We recommended getting the wireless version because it is easier to hide and less noticeable--no long black cord; which are all things that we need for our Retro Cubicle Commando. You can also avoid the USB audio card cost. But who wants to play games

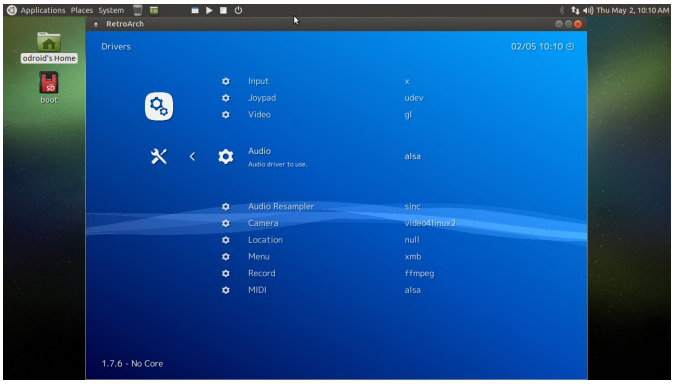
with no sound? The WiFi adapter is also optional if you do not plan to do any web surfing on the nearby cafe's customer WiFi network, otherwise, it is a perfect addition to our stealthy retro gaming console. Alright, let us get started.

Setting Up the Audio

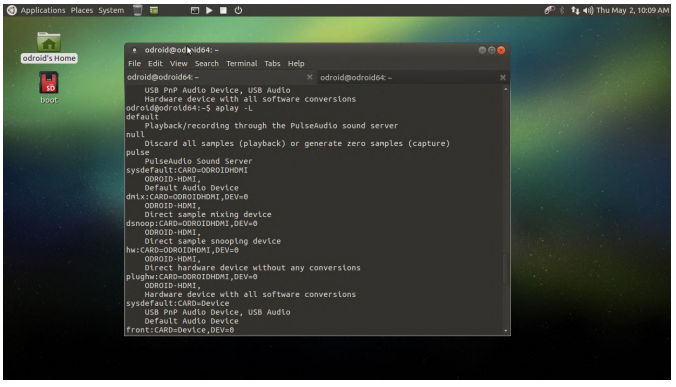
First things first, let's turn off full screen mode in retroarch. Launch retroarch, if needed, navigate to the Drivers section and scroll down to the Video option, scroll down to the Start in Fullscreen Mode option, and set it to off. Use this same process and turn it back on when we are done adjusting retroarch. The screenshot below depicts the setting that needs to be adjusted.



Next, navigate to the Drivers section of the menu system and select the Audio option. Change the Audio Driver option to alsa. Scroll further down and look for the Device option. Make sure to leave the setting on its original value. Use the left and right navigation to flip through the available audio devices. Note: If you are unable to select any devices, exit retroarch, restart it, and try to flip through the available audio devices again. We will be adding the config file changes by hand so this part is not the most important.



If you are having trouble listing the audio hardware through retroarch, exit retroarch and open up a terminal at the menu location Applications -> System Tools -> MATE Terminal. Type the following command, aplay -L you should see output similar to that shown below.



You should see an entry that is similar to this one, hw:CARD=ODROIDHDMI,DEV=0, and after plugging in the USB audio adapter one that is similar to this one, front:CARD=Device,DEV=0. Make a copy of these strings and save them in a temporary file. Run the following commands from the terminal in the odroid home directory, i.e. the default directory.

```
$ cp .config/retroarch/retroarch.cfg
.config/retroarch/retroarch.cfg.Orig
$ cp .config/retroarch/retroarch.cfg
.config/retroarch/retroarch.cfg.VgaAudio
$ cp .config/retroarch/retroarch.cfg
.config/retroarch/retroarch.cfg.HdmiAudio
```

We are making a backup of the current retroarch config file and two copies we can use with scripts to alter the audio configuration. Once those commands are done running, open up retroarch.cfg.VgaAudio using your favorite CLI editor, I will use nano.

```
$ nano .config/retroarch/retroarch.cfg.VgaAudio
```

Scroll down to the line that has the audio device setting. Change the value of this setting to front:CARD=Device,DEV=0. Repeat these editor selection steps for the retroarch.cfg.HdmiAudio file except for this file enter hw:CARD=ODROIDHDMI,DEV=0 for the audio device value.

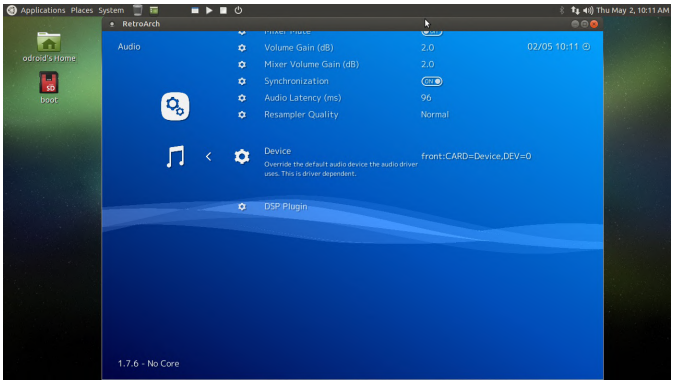
Once these changes have been made, write some scripts that utilize the files we have just created. Use your favorite editor to create the following files.

```
#!/bin/bash
#set_hdmi_audio
$ cp
/home/odroid/.config/retroarch/reroarch.cfg.HdmiAu
dio
/home/odroid/.config/retroarch/reroarch.cfg
#!/bin/bash
#set_vga_audio
$ cp
/home/odroid/.config/retroarch/reroarch.cfg.VgaAud
io
/home/odroid/.config/retroarch/reroarch.cfg
```

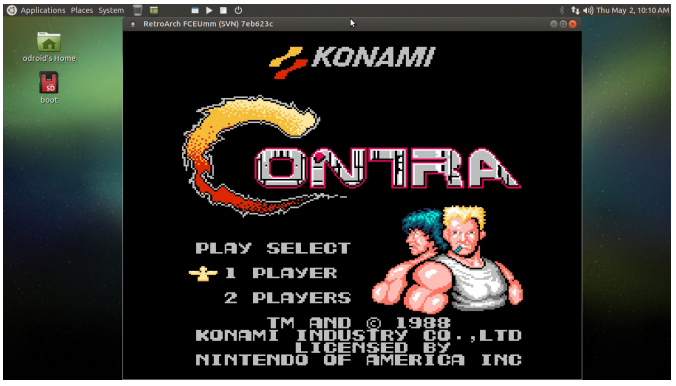
Let's set the proper permissions for these files. Run the following commands at the terminal.

```
$ sudo chmod 755 set_vga_audio set_hdmi_audio
$ sudo chmod +x set_vga_audio set_hdmi_audio
```

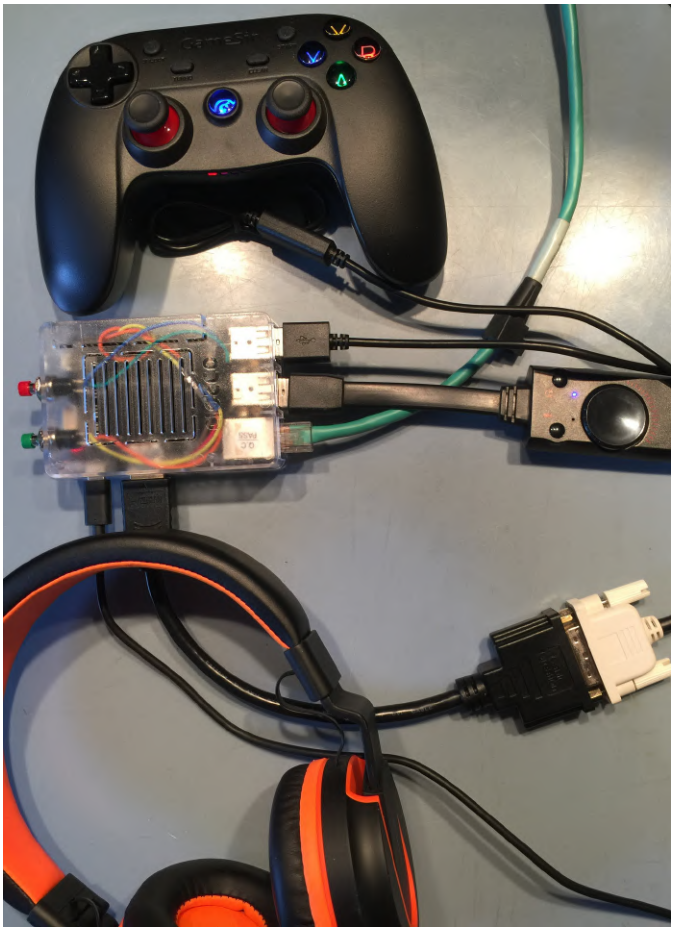
Run the set_vga_audio script you just created and get ready to test out your retro gaming console on a VGA screen with USB audio adapter. Fire up retroarch and verify the Device setting.



Head over and try out a game with your headset on and experience stealth retro gaming you can pull off at your desk! Note: Do not forget to restore the full screen setting to all copies of the retroarch config files, even the new ones you have made! Just use Ctrl + W and type in fullscreen to locate the entry, set its value to "on" if it is "off." Do this for all copies of the config file.

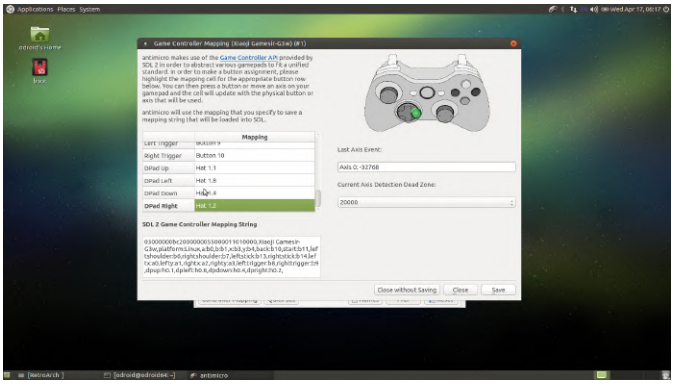


A slightly less stealthy Cubicle Commando setup but it gets the job done.

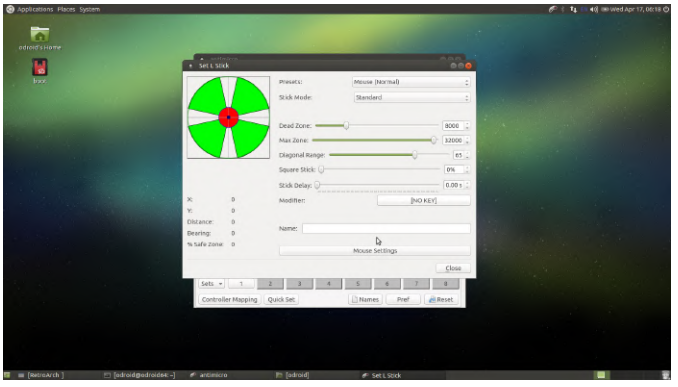


Setting Up the Controller

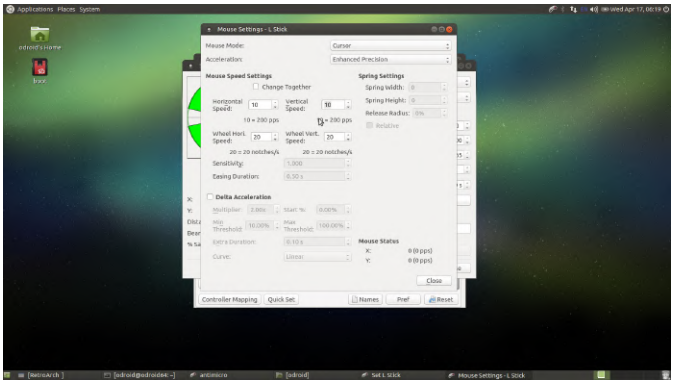
I will assume that the GameSir wireless controller has not been previously configured for your retro gaming console. This next part will walk you through configuring a new controller for a Monku Retro 1, 2, 3 (ODROID-C1+, -C2, -XU4) device. Let us get antimicro configured so we can start controlling the desktop environment with the gamepad. Open up a terminal, I will not list the menu path for it from this point forward. Type antimicro in the terminal and wait for the app to launch. Connect your linux supported controller and make sure that antimicro recognizes it. If it doesn' work, you will need to try another controller. Click the Controller Mapping button on the bottom left hand corner of the UI. This is where you tell antimicro about the base functionality of your controller. If you do not have a button for a specific position in the list, for instance Linux seems to ignore the blue central button on the GameSir controllers, use your mouse to click down to the next viable option. Match up the buttons on the gamepad with the controller graphic's green button indicator. Note: Some buttons, like triggers, fire multiple times and you will have to use the mouse to back up the position of the mapping and fix the double entry. Click save when you are done and return to the main antimicro UI.



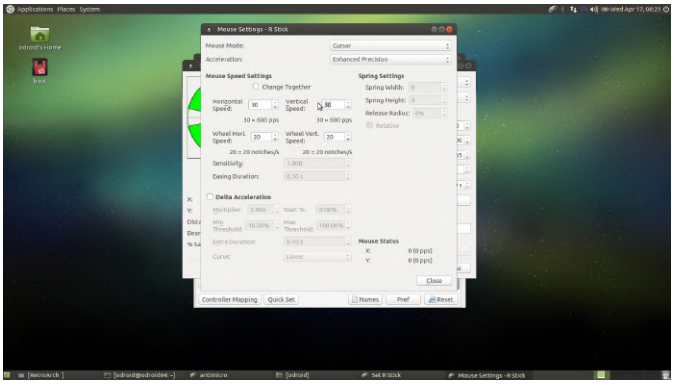
You will see a new mapping on the main antimicro UI that contains buttons for all the new mappings you just made. What we are going to do here is setup mouse support so that you can control the desktop environment from the gamepad when retroarch is not running. We will use the left thumbstick for fine-grained, slower, mouse control and the right thumbstick for faster mouse control. The A and B buttons will serve as the left and right mouse buttons. Right click on the left thumbstick area and select mouse normal from the option list.



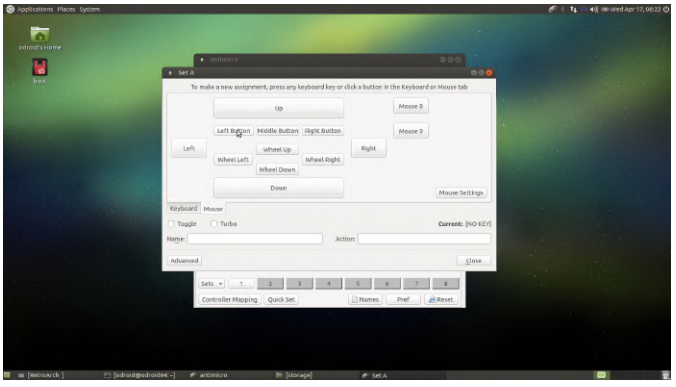
Click on the left thumbstick buttons again and find the Mouse Settings button at the bottom of the window. The image above shows the button we are looking for. In the mouse settings window, set the Horizontal Speed and Vertical Speed to 10 for the left thumbstick as depicted below.

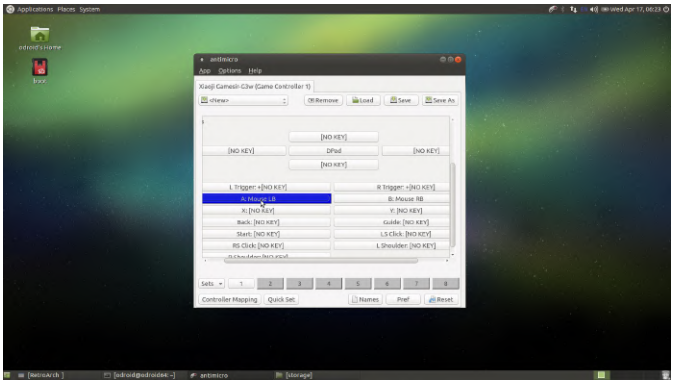


Do the same thing for the right thumbstick except set the Horizontal Speed and Vertical Speed to 30 as depicted below.

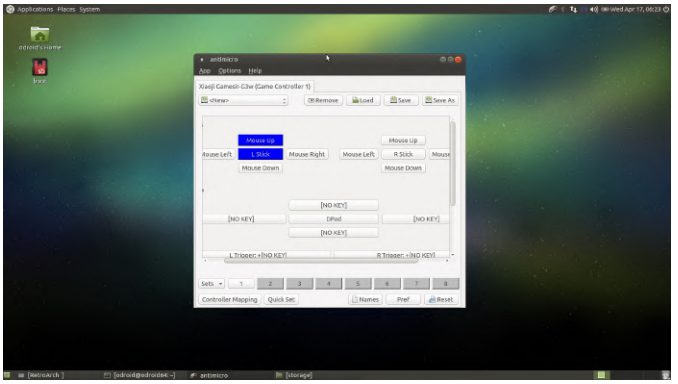


Now let us map the mouse buttons, close all dialogs and get back to the main antimicro UI. Find the A button in the button list below the thumbstick and dpad listing. Click on it then click on the Mouse tab. Select the left mouse button. Do the same thing for the B button except choose the right mouse button for that mapping. Below is a screen shot depicting the left mouse button mapping in action.

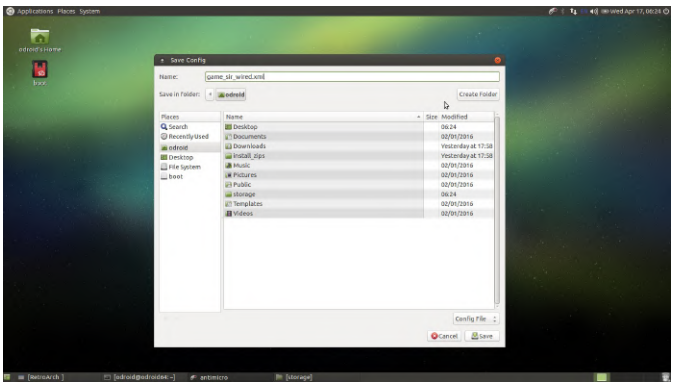




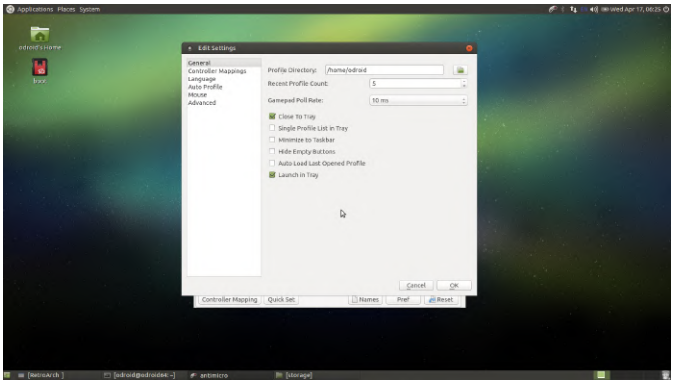
Take it for a spin while the main antimicro UI is open. You should see the mouse move around the screen as the button listings in the antimicro UI turn blue to indicate they are active. Test how it feels, adjust the speeds of the mouse controls as you see fit.



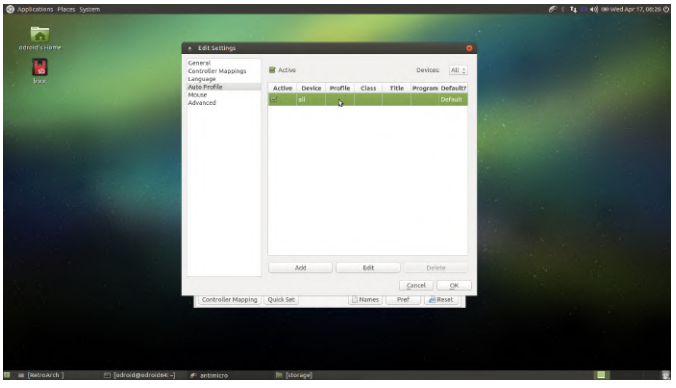
When you are all set to go back to the antimicro main UI, click the Save As button at the top right hand side of the screen. Save the controller configuration as game_sir_wired.xml or whatever you want to name your controller in the ODROID home directory as shown below. I will provide a copy of my XML file (**http://middlemind.net/images/products/monku_r1_build/gs3w.xml**) if you are using a GameSir controller you can just use it and save yourself some time. If you are using an Easy SMX controller use this (**http://middlemind.net/images/products/monku_r1_build/easySmx.xml**) file.

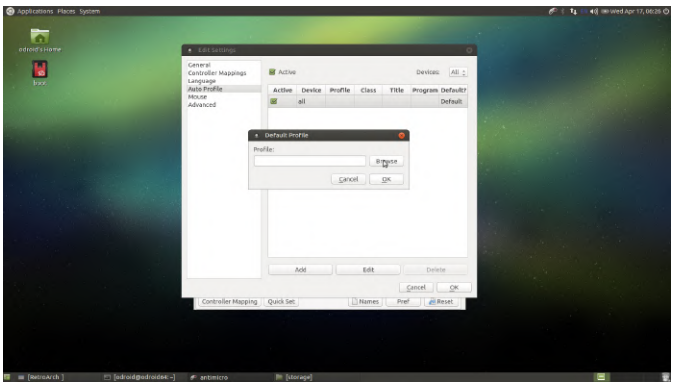


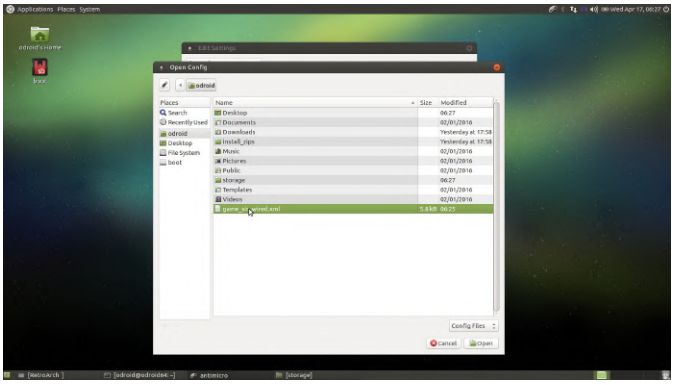
Click on Options -> Settings in the antimicro menu and make sure only Close To Tray and Launch In Tray are checked. This will ensure antimicro lives in the app tray and does not clutter up our screen. We have one more setting to adjust and then we will be done with antimicro and on to retroarch!



While still on the antimicro setting window, click on the Auto Profile option on the left. This will determine what profile will automatically be associated with the attached gamepad. You only get one mapping. It would be cool if it had different options for different hardware but as far as I can tell you are setting it up for the controller you have. Click the Active checkbox at the top of the window. Then select the Default row in the table. Click the Edit button and browse to the controller mapping XML file you saved just a few steps back. Click Ok, then quit antimicro. If it appears in the system tray, click the controller icon in the system tray and quit the app. Nice! We are done with the antimicro configuration!!







Next up let us whip retroarch into shape. Fire up retroarch from the menu system, I will not list the menu path for it from this point forward. First, let's get the gamepad working in retroarch. In retroarch, you can use the keyboard arrow keys, enter, and backspace to navigate the menu system without the gamepad. Make sure you have a mouse, keyboard, and game controller connected to your ODROID. First thing we will do is get the controller working. Use the arrows on the keyboard to navigate right to the Settings section, then move down to the Input section as shown below.

Adjust the settings on this screen as you see below. I usually set the max number of controllers to 4 since there are 4 USB ports. And I like the "L1 + R1 + Start +

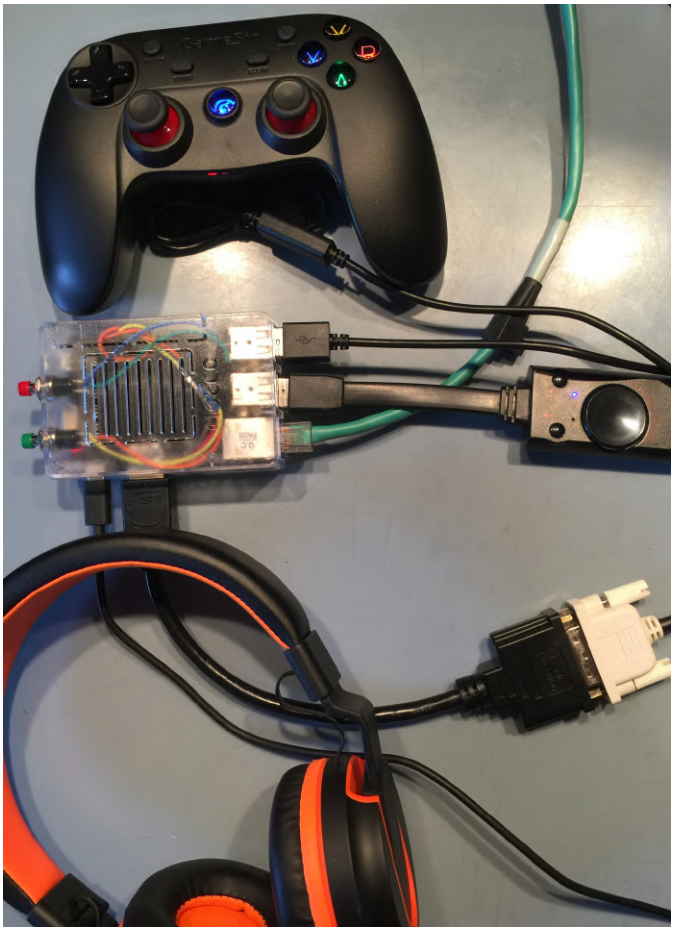
Select" Menu Toggle Gamepad Combo setting. Let's face it, if you're accidentally hitting this combination during game play something isn't right. Leave the remaining settings and scroll down to the User 1 Binds. You will have to setup each user input in this way it is not too bad and only takes a minute. Tip: Map the A and B buttons by name not position. If you're using a GameSir controller, the colors green and red map to positive/select, negative/back button usage. It's just what I like to do, but you can map 'em anyway you like!

Perfect, that should be all you need to do to use your new wireless controller. Stealth gameplay is just around the corner.

Setting Up the WiFi USB Adapter

Plug the WiFi USB adapter into your ODROID device configured as a Monku Retro gaming console or, otherwise, running Ubuntu and MATE. At the top right hand corner of the desktop screen, after closing retroarch, if need be, you should see a network icon, click on it. Use the menu options to add a new wireless network so you can surf the web when your not gaming.

Well that wraps up this tutorial. I hope you enjoyed it. Now go get some stealth retro gaming hours inside your cubicle!



References

http://middlemind.net/tutorials/odroid_go/mr1_build.html
http://middlemind.net/tutorials/odroid_go/mr3_build.html
https://amzn.to/2m3QWii
https://amzn.to/2m8BnWs
https://bit.ly/2kWBbcP
https://bit.ly/2JxEu4V
http://middlemind.net/tutorials/odroid_go/5mf_mr1_cc.html

SMS Text Messaging Status Updates With Your CloudShell2: Remote Monitoring Made Simple

October 1, 2019 By AreaScout CloudShell, Linux, Tutorial

There is a new version of the Cloudshell2 Info and Monitoring Tool, Version 1.0.4-1. Following are a couple of new features:

- Add GSM Shield support via USB2UART device
- Add new command line switch to identify a defective hard disc more easily (experimental)

GSM shield support

Figure 01

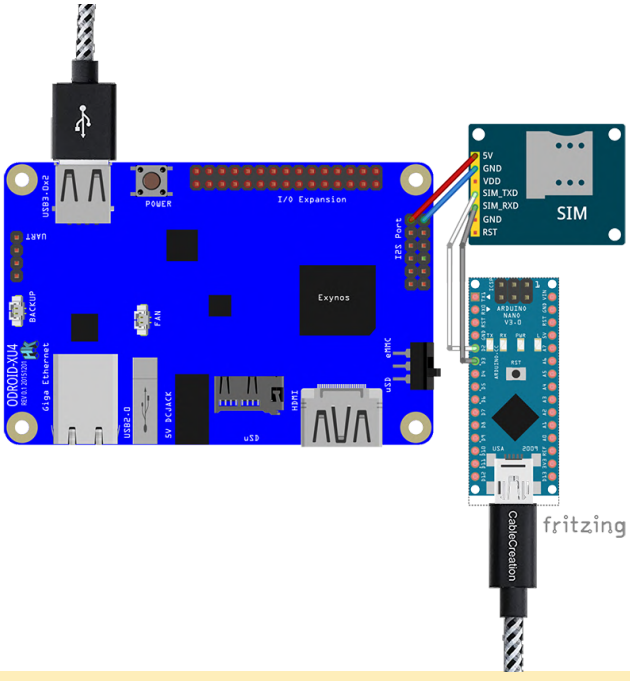


Figure 02

The picture above shows how I connected the GSM shield to my ODROID-XU4, the picture also shows an Arduino Nano which is only a suggestion as they are very cheap and it's successor (Arduino Nano Every) is only 8€ with a stronger microprocessor, of course, you could use any USB2UART device (see link below) but a lot of you guys own maybe already an Arduino so you could also use it, each Arduino should also have a tristate mode in which you start the Arduino with RST and GND connected to each other. It's like a bridge from USB to pin 0 and 1. If this is not working for you, like it didn't work for me, you have to use a sketch/source like this example (**https://bit.ly/2kxFxaf**). Skip the text output, use the same baud rate for hardware und software serial and it should work.

Here is a link with further instructions for which device you should use to communicate with the GSM Shield: **https://bit.ly/2kXcxZD**. By the way, I used this **https://bit.ly/2kqkbLI** to connect the 5V to my GSM shield, depending on what GSM shield you are using this could be entirely different, external power supply plug, etc.

Before you connect anything you should setup the GSM shield with your PC/Laptop/ODROID with minicom minicom -b 115200 -D /dev/ttyUSB0 or if you want to do that on Windows, you can use my tool (SerialLogger) which I wrote back in 2013. It is shipped with the debian package and you can find it inside the /usr/share/doc/cloudshell2-monitoring/ folder. It is a 64-bit application, if you want a 32-bit or WindowsXP build, please ask.

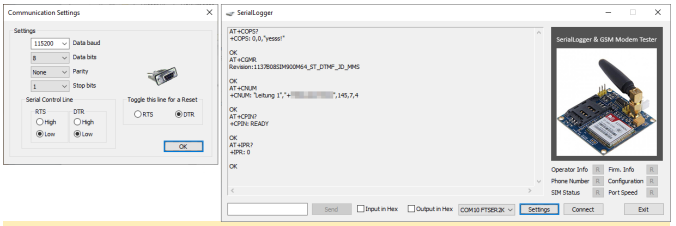


Figure 03 - Serial Logger

If the communication is working, you must disable the PIN of your SIM card with this command AT+CLCK="SC",0,"1234" (1234 should be exchanged with your PIN) and configure other settings about your network band. You should already be able to send and receive SMS with your shield (Fixed baud rate setup is also recommended instead of auto on your GSM shield).

Now you can setup the GSM shield inside the /etc/cloudshell2-monitoring/cloudshell2-monitoring.conf config file, with your phone number to send SMS, nickname, port name, and baud rate.

Note: I do not check signal strength nor do I check if the GSM shield is connected to a network provider, you have to have a good cell reception and the right frequency band must selected and auto select network must be enabled.

If everything is working and your PIN is disabled and your GSM shield is connected to your ODROID-XU4, you have to restart cloudshell2-monitoring tool sudo systemctl restart cloudshell2-monitoring.service and you can check if the GSM shield was detected with:

```
odroid@odroidxu4:~/CloudShell2$ sudo systemctl
status cloudshell2-monitoring.service
● cloudshell2-monitoring.service - "ODROID
Cloudshell2 Monitoring"
   Loaded: loaded
(/lib/systemd/system/cloudshell2-
monitoring.service; enabled; vendor preset:
enabled)
   Active: active (running) since Thu 2019-09-12
18:05:17 CEST; 5s ago
 Main PID: 6137 (start-lcd)
   CGroup: /system.slice/cloudshell2-
monitoring.service
           ├─6137 /bin/bash /usr/bin/start-lcd
           └─6141 /usr/bin/cloudshell2-monitoring

Sep 12 18:05:17 odroidxu4 systemd[1]: Started
"ODROID Cloudshell2 Monitoring".
Sep 12 18:05:17 odroidxu4 start-lcd[6137]: Found
GSM Shield
Sep 12 18:05:17 odroidxu4 start-lcd[6137]: GSM
disable echo
Sep 12 18:05:17 odroidxu4 start-lcd[6137]: GSM
shield command executed OK
Sep 12 18:05:17 odroidxu4 start-lcd[6137]: GSM
enable live SMS
Sep 12 18:05:17 odroidxu4 start-lcd[6137]: GSM
shield command executed OK
Sep 12 18:05:17 odroidxu4 start-lcd[6137]: GSM set
SMS text mode
Sep 12 18:05:17 odroidxu4 start-lcd[6137]: GSM
shield command executed OK
```

If this is OK, you can now send SMS messages to your CloudShell2 and your CloudShell2 will automatically inform you if something goes wrong (RAID error, over temperature, etc.)

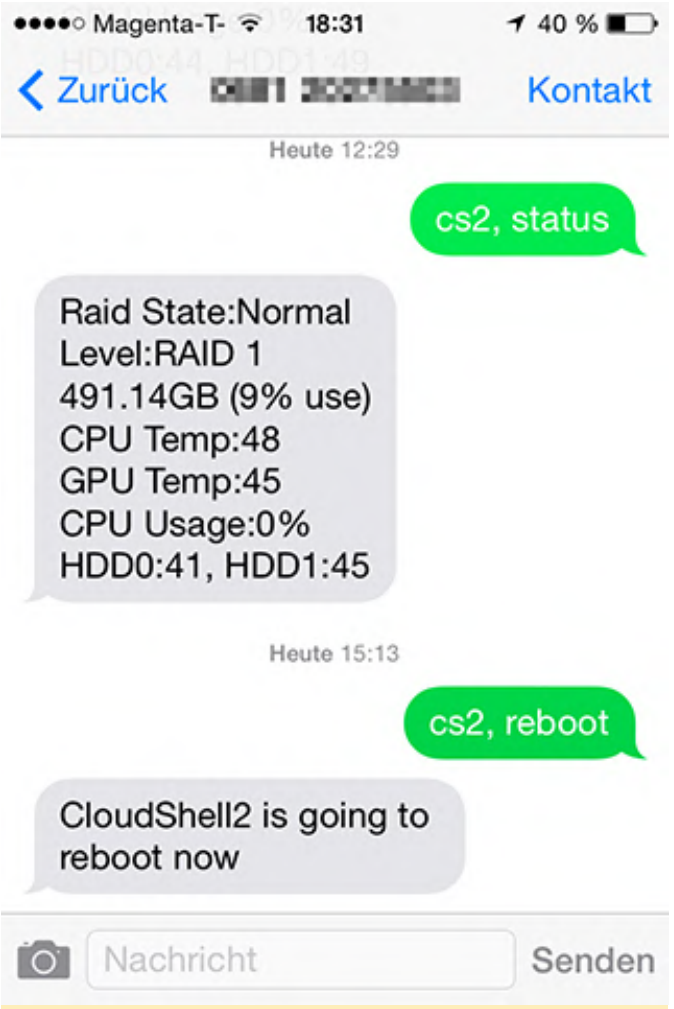


Figure 04 - iPhone SMS messages

Valid commands at the time of publication, include (status, reboot, shutdown) with the nickname of your cloudshell2 as prefix (cs2 as default), like in the picture above.

References **https://bit.ly/2mtQdr0**

Monku R3: Building The Ultimate ODROID-XU4 / XU4Q Gaming Console - Part 1

October 1, 2019 By Brian Ree Gaming, ODROID-XU4

Requirements

Tools:

- A small screwdriver set that contains a few small Phillips head screwdrivers.
- A clean static free work surface.
- Monitor or TV with HDMI support to test the device.
- Mac SD card image writing software. I use balenaEtcher, it is free and works great.
- Window SD card image writing software. I use Win32 Disk Imager, it is free and works well but can be a bit finicky with very large drives.

ODROID items:

- ODROID-XU4 / ODROID-XU4Q x1
- Case x1
- 64GB Micro SD Card x2
- HDMI Cable x1
- Power Supply 5V/4A x1
- GameSir Wired Controller x1
- USB WiFi Module x1
- SD Card Reader x1

The total project cost as described above and not including shipping or tools you do not have is around $94. If you exclude the game controller or already have one you can save around $17. Also you do not need two micro SD cards. I like to have a spare one, in case one goes bad and the dual set listed above has a great price. Also, the card is well rated and from my personal experience I have only had one fail unexpectedly out of 12 purchases so I have been using them regularly during the configuration and development of these devices. If you have an HDMI cable, a micro USB cable, and a 5V/4A 5.5mm barrel AC adapter then you can save even more on the cost of this project. So while the price listed is around $94 you can probably get it done for around $80 if you

have some parts already. Not too bad once you see what this thing can do.

Introduction and Tutorial Goals

This tutorial will cover the setup, and construction of the game console from a hardware and software point of view. Now unlike the Monku Retro 1 and 2, we will not be adding any special hardware buttons. The ODROID-XU4 comes with a hard reset button built in, so that is already done for us. As for the custom control button I have not found a good location for it using the current case. However, this device is much more powerful than the ODROID-C1+ or even the ODROID-C2, so for now we will not add one to the R3. We will also cover all the software setup including installing and configuring Ubuntu, retroarch, and antimicro. Let us take a look at some of the features of the device we are working building. And look at that emulator list!

R3 / ODROID-XU4 Features:

- ODROID Goodness!
- Hardware Reset Button
- Support for Atari 2600, Atari 7800, Atari Lynx, ColecoVision, MSX-1, MSX-2, NES, GameBoy, GameBoy Color, Virtual Boy, SNES, GameBoy Advance, WonderSwan Pocket/Color, NEO GEO Pocket/Color, Sega SG-1000, Sega Mark 3, Sega Master System, Sega Genesis, Sega GameGear, NEC Turbo Graphics 16, NEC Super Graphics, PSP, and PS1 emulators configured and ready to go.
- Retroarch with XBM, custom scripts to monitor the software button, start retroarch, maintain antimicro.
- Configured for low memory usage and for use with included controller.
- Full linux desktop environment gamepad control when not in game kiosk mode via antimicro.
- WiFi network connectivity.

Take a look at the performance specs of this device when compared to some other common devices.

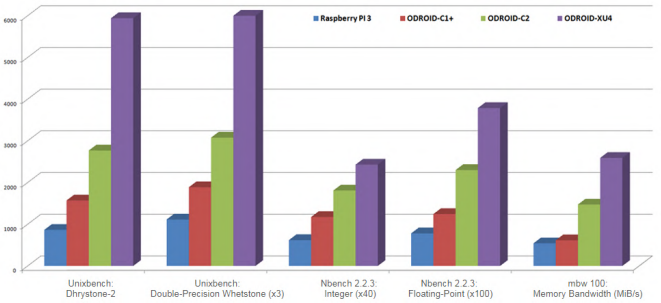


Figure 01 - ODROID-XU4 versus other boards.

The Hardware

First things first - let us go over the tools and parts, lay them out, and get ready to build. We have an electronics screwdriver set. If you have built an ODROID-GO, the same screwdriver set should work fine. Our target device, an ODROID-XU4, is depicted below. This tutorial applies equally to the ODROID-XU4 or the ODROID-XU4Q version of this device. The device runs just about every emulator you can think of and it runs them wonderfully. We have our board, case, SD cards, USB WiFi module and tools all ready to go.

Figure 02 - Monku R3 build 1

Clear your workspace and grab the case, take it out of its plastic bag if need be, place it down in the center of

the work space. There are two main clips on the case, all-in-all it is easier to work with than the ODROID-C1+ / ODROID-C2 cases. The first main clip is on the left hand side of the case bottom, near the top. The second main clip is on the top side of the case bottom, near the right. You can see slight rectangles near these areas in the image below.

Figure 03

To clear the first main clip, give the case a slight skew as shown below. Ever so slightly pushing the bottom to the left while pushing the top to the right should do it.

Figure 04

Once it comes undone flip the case around so that the other main clip is in the position depicted below. Apply a similar set of actions until the clip separates. It should come apart easily once you get it right. Notice we are using a similar technique for the second main clip as we used for the first.

Figure 05

Figure 06

Once you have the case separated you will find a surprise inside. A bag of tiny screws. If you have an ODROID-GO and you have a good amount of leftover screws I would recommend using them instead of the screws provided. Now this could have changed but at one time the default case screws were a bit smaller than the ODROID-GO screws and I found the extra screws in the ODROID-GO kit to be easier to work with. Let us layout the tools and parts we need to assemble the case with the ODROID-XU4 board mounted inside. You will not have to worry about SD card access, if you are familiar with the ODROID-C1+ or ODROID-C2 case, the SD card is easily accessible by default.

Carefully open the antistatic bag that the ODROID-XU4 comes in. Make sure you do not have a static charge: discharge yourself against something large and metal. Layout the case and the board, rest the board on top of the antistatic bag, we are going to place the board on the bottom half of the case and place and tighten the 2 internal screws. The ODROID-XU4 case is similar to the ODROID-C1+ / ODROID-C2 case in that there are two internal screws and 2 external screws.

Tip: Make sure the screws are tight but do not over tighten them, snug would be a good description of how much to tighten them.

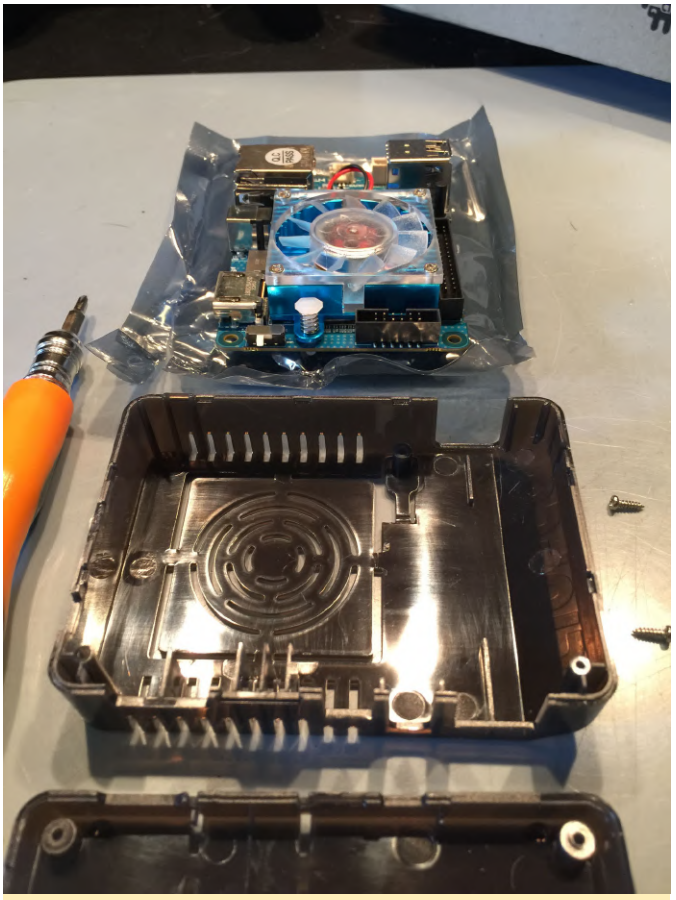


Figure 07

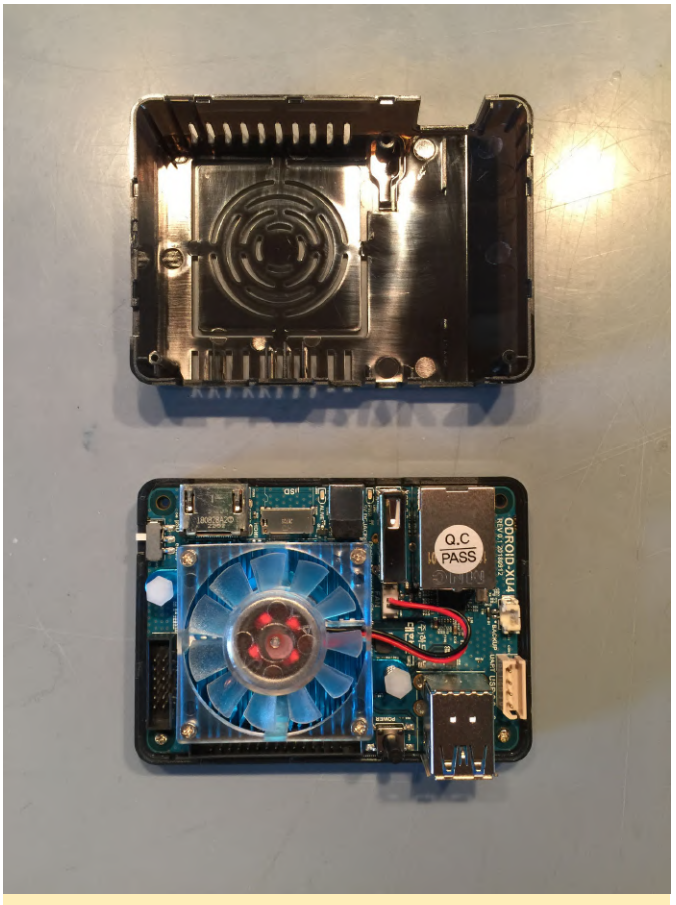


Figure 08

Flip over the case and place and tighten the two external screws. With the back of the case facing towards you, note the small white switch. Flip the switch closer to the edge of the case for SD card use, flip the switch the other way, closer to the center of the case, for eMMC use. You are all done with the hardware construction. Next up we will be working on the base SD card and OS image.

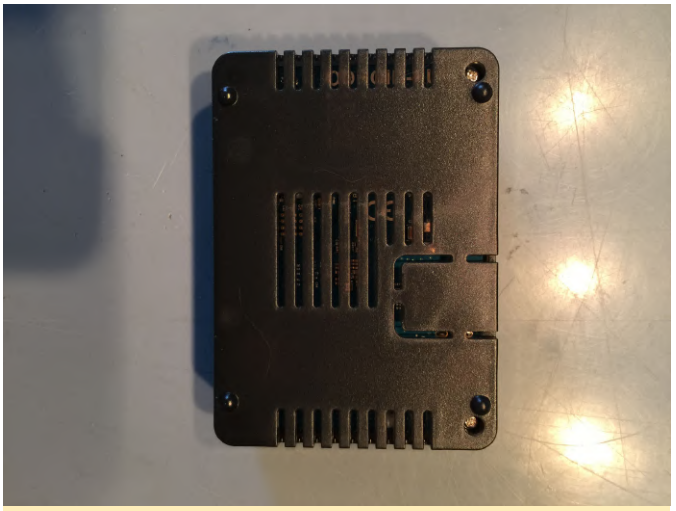


Figure 09

Figure 10

The Software

For the next step we are going to locate a specific Ubuntu OS image. I have tried a few images and ran all sorts of tests and I have come to the conclusion that Ubuntu 16.04 LTS offers the best mix of performance, features, and support. Head over to the Hardkernel wiki. Locate the ODROID-XU4 entry on the right hand side of the site. Select it, and then select os_images from the list of options. Next select linux and ubuntu_4.9. I know it is an older kernel and no longer supported but I wanted to run with a slightly older kernel that has less features and is a little bit more efficient. You can try a newer one. The software configuration steps should be almost identical as long as you are using MATE.

Figure 11

I use the South Korean mirror most of the time. You can use any one that works best for you. Find the entry ubuntu-16.04.2-mate-odroid-xu4-20170510.img.xz or you can use this direct link https://tinyurl.com/y3oj4lr3.

Figure 12

Next decompress the downloaded OS image. You should end up with a file entry named ubuntu-16.04.2-mate-odroid-xu4-20170510.img that is almost 5GB in size. You will need to flash a microSD card with the OS image now. A link to the SD card I currently use is posted above in the parts listing. For Apple MacOS, I use Balena Etcher. On Microsoft Windows, I use Win32 Disk Imager. Linux users can use some CLI tools like dd.

On MacOS, install Balena Etcher from the link. Select the uncompressed OS image we just downloaded. Insert your micro SD card into your Mac using a converter of some kind, link to one listed above. Make sure to select the proper target drive. You do not want to overwrite important data so make sure to double check the destination drive. Once you are sure everything is set correctly, flash the OS image to the SD card. This will only take a few minutes. The SD card will be unmounted and ready to remove at the end of the process.

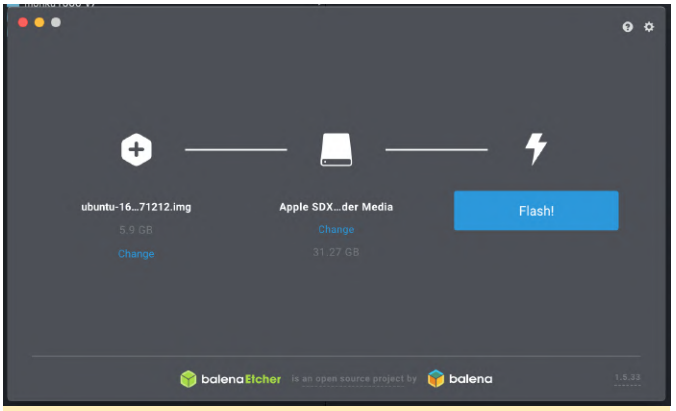


Figure 13

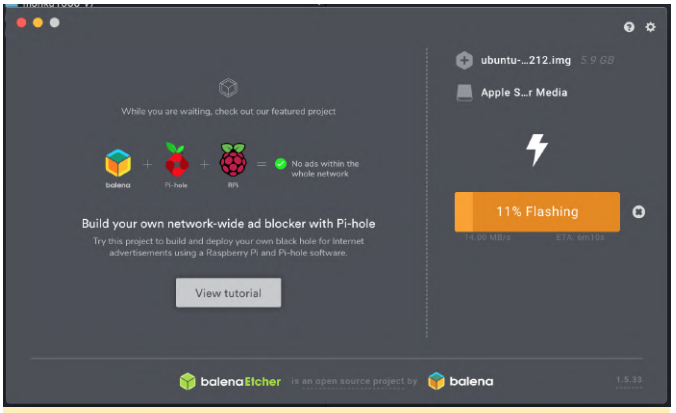


Figure 14

On Windows, install Win32 Disk Imager from my link. Select the uncompressed OS image we just downloaded. If you have trouble uncompressing the OS image, try using 7-Zip. It has come in handy for me in the past.

ALERT: Make sure to triple check your destination drive letter. It is very easy in Windows, using this software, to select the wrong drive and ruin your PC. So triple check and make sure you have the correct drive letter selected. Once that is done, click the Write button to write the OS image to the SD card. Wait a few minutes and let it complete. You will want to unmount the SD card before removing it from the computer.

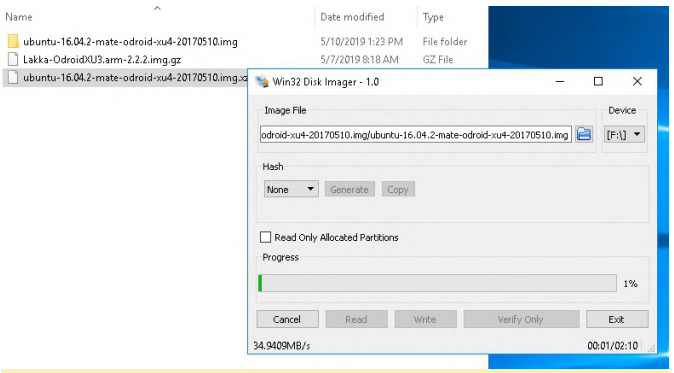


Figure 15

On Linux use the following command to list the drives on the system.

```
$ sudo fdisk -l
```

Look for an entry like the following. Make sure you have identified the correct drive by ejecting the SD card and checking that the entry is not longer available.

ALERT: Be careful and take the time to correctly identify the correct Linux device or you could end up overwriting one of your drives.

```
Device Boot Start End Sectors Size Id Type
/dev/sda1 8192 30253055 30244864 14.4G c W95 FAT32
(LBA)
```

To restore an image to a drive using the terminal on an Ubuntu linux system, use the following command:

```
$ sudo dd of=/dev/sda1 if=~/ubuntu-16.04.2-mate-
odroid-xu4-20170510.img
```

Wait a few minutes for the process to finish and an output report to be printed to the terminal. Now you have an SD card ready to use on your ODROID-XU4!

Software Updates, Partition Sizes, and More

First things first--let's get rid of that pesky login prompt. If your device has USB trouble on boot up just power cycle it. I would say though to use the software control buttons once things are all setup and never use the hardware power button unless you are turning it back on again. For the remainder of the tutorial, you should have a keyboard and mouse hooked up to the device. If you go to the following menu location Applications -> System Tools -> MATE Terminal it will open up a terminal window for you.

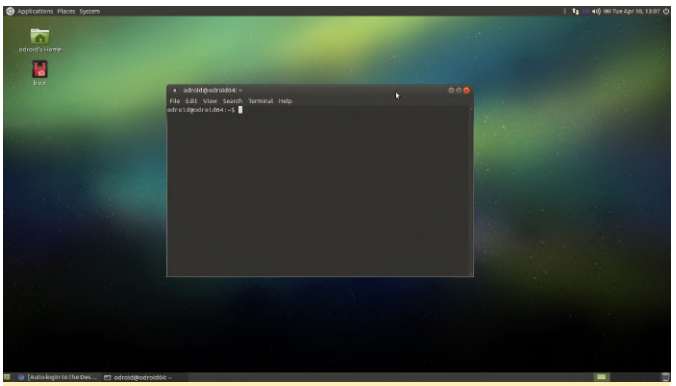


Figure 16

We are going to run a series of commands at the terminal. I am going to list them below. Some take a while to run but you may have to sit near the screen in case you are prompted by an installation question. We are setting up auto login, updating Ubuntu, and installing some packages.

Type this command or copy and paste it into the terminal window.

```
$ sudo nano /usr/share/lightdm/lightdm.conf.d/60-
lightdm-gtk-greeter.conf
```

You will be prompted for the password, use odroid. You will see some text like this in the file.

```
[Seat:*]
greeter-session=lightdm-gtk-greeter
```

You are going to add a line at the bottom, autologin-user=odroid, and then save and close the file. Press Ctrl+O then hit enter to save the file. Press Ctrl+X then press enter to exit the editor. You will not be prompted to login on the next reboot.

Next up we will be running updates on the OS packages and installing a few things. This part takes some time, but, for the most part, it runs by itself. Enter and execute each of these commands in the order shown below from the terminal window.

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

If you get a boot.ini prompt for this command just hit enter.

```
$ sudo apt-get upgrade -y
$ sudo apt-get install git -y
$ sudo apt-get install gparted -y
$ sudo apt-get install make -y
```

```
$ sudo apt-get install cmake -y
$ sudo apt-get autoremove -y
```

Now that all of that is done, the system is starting to shape up a bit. The next thing we want to do is turn off any swap space the OS is using. The conventional wisdom is that swap partitions will degrade the SD card which I think are rated for some number of read/write operations before they begin to fail. So far I have been disabling them without a noticeable performance hit. Run the following command at the terminal.

```
$ swapon -s
```

If you see output similar to the following, then you have zram enabled. Follow the steps below to turn it off and remove it.

```
$ swapon -s
Filename Type Size Used Priority
/dev/zram0 partition 219824 2080 5
/dev/zram1 partition 219824 2076 5
/dev/zram2 partition 219824 2076 5
/dev/zram3 partition 219824 2076 5
```

Ok so we want to disable these swap partitions so our SD card lasts as long as it can. Run the following command. The ODROID-XU4 I believe does not have a swap partition by default.

```
$ sudo apt-get remove --purge zram-config -y
```

If you notice a standard swap partition listed, use these steps to remove and disable the swap partition.

```
$ swapoff -a
```

This will immediately disable swap on the system. Next remove any swap entries from /etc/fstab by editing the file as root and commenting out any swap partition entries. Reboot the system. If, for some reason, the swap partition is still there open gparted, System -> Administration -> GParted. Locate the partition in the list of active partitions and unmount, then delete the partition.

Next up we are going to run a MATE software update by navigating to System -> Administration -> Software Update in the menu system.

ALERT: If you are asked to perform a partial update, then skip this step. We will run with the packages we have installed. I have noticed that in some cases the packages can get out of control. I would only proceed with an update, if no Partial Update dialog pops up.

If there is not a partial update prompt, click the Update button, wait a little while it slowly turns into a progress bar dialog and perform the updates. At the end of the process, you will be prompted to restart the system.

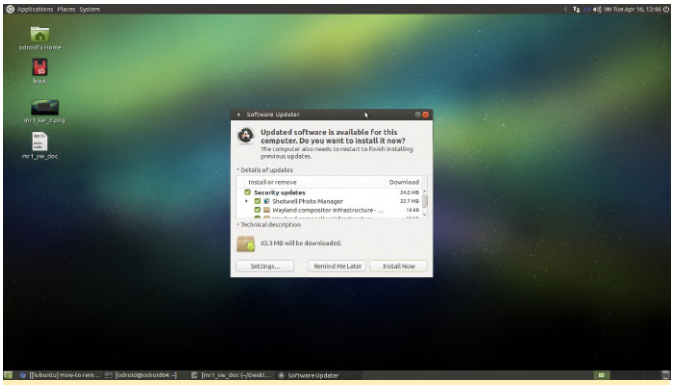


Figure 17

Notice the absence of a login prompt when the system started up. It's looking more and more like a retro gaming console by the minute. Ok last thing we are going to do is resize the main partition to use all the available space. If you go to System -> Administration -> GParted in the menu system you should see something like what's depicted below. Notice that there is around 50GB of unused space. If you have less unused space, but still have room on the SD card, you need to resize your root partition.

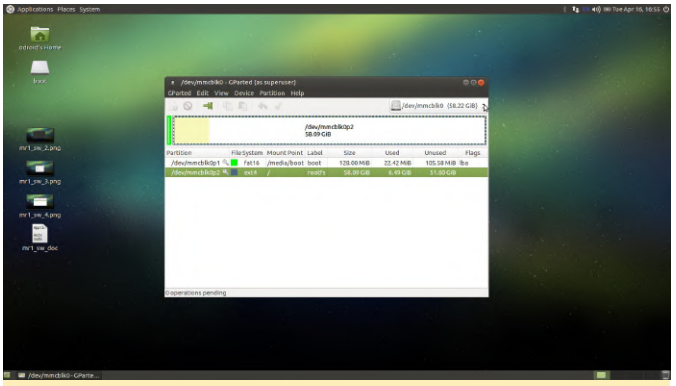


Figure 18

I will give you a quick rundown of the process.

ALERT: You may not need to do this but you should double check anyway. The best way to work with an ext4 file system is on our ODROID-XU4. Use the

second SD card that comes with the recommended purchases listed on the hardware build. Write the base OS image onto the SD like we did above and install gparted, also like we did above. You do not have to run all the updates, just make sure gparted is installed. Use the SD to USB adapter, listed above, to mount the SD card that we want to resize its partitions onto the OS. You should see a little drive icon appear, we will call this SD-USB for SD to USB.

Fire up the bare bones Ubuntu SD card if you haven't already. Start gparted from the menu System -> Administration -> GParted and select the SD-USB card attached--not the root file system running Ubuntu.

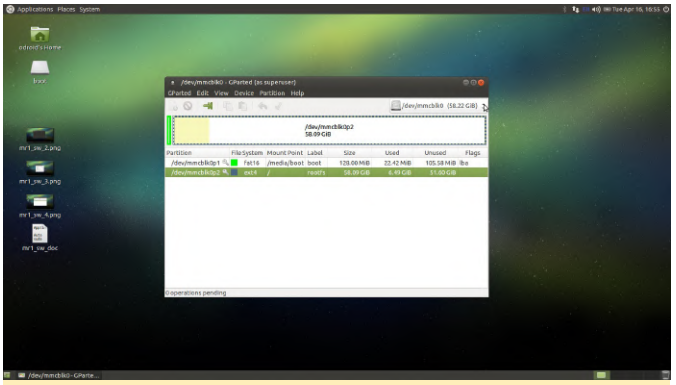


Figure 19

Select the root file system on the SD-USB card. Right click on it and go to the resize option if you get an error you may need to unmount this partition and then try to resize it. You can visually resize the partition now by dragging the arrow all the way to the right or by setting the fields contained in the form to have 0 free space following. We are done with this section.

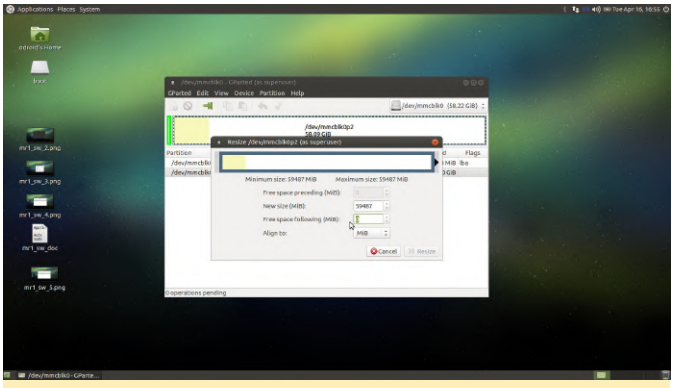


Figure 20

Retroarch and Antimicro Installation

Alright now we are getting somewhere. Let's get retroarch and antimicro installed, so we can begin the configuration process. To install retroarch we need to open a terminal, Applications -> System Tools -> MATE Terminal.

```
$ sudo add-apt-repository ppa:libretro/stable &&
sudo apt-get update -y && sudo apt-get install
retroarch* libretro-* -y
```

Try the command above on the ODROID-XU4, hit enter if prompted, and Y if prompted. If it fails, do not worry. I noticed it fails on the ODROID-C2, but I am not sure about the ODROID-C1+--the *'s in the package lists are the culprit. It will pull down some packages that have dependency issues and that then halts the whole command. If it fails, try running this command instead.

```
$ sudo apt-get install retroarch retroarch-assets
retroarch-dbg libretro-beetle-lynx libretro-
genesisplusgx libretro-handy libretro-4do
libretro-bsnes-mercury-performance libretro-bsnes-
mercury-accuracy libretro-bsnes-performance
libretro-beetle-wswan libretro-dinothawr libretro-
beetle-ngp libretro-bsnes-balanced libretro-
gambatte libretro-fbalpha2012 libretro-fba
libretro-beetle-psx libretro-vba-next libretro-gw
libretro-mupen64plus libretro-beetle-sgx libretro-
2048 libretro-tyrquake libretro-beetle-pcfx
libretro-prosystem libretro-bsnes-accuracy
libretro-parallel-n64 libretro-picodrive libretro-
mame libretro-nestopia libretro-mednafen-psx
libretro-core-info libretro-gpsp libretro-mess
libretro-beetle-pce-fast libretro-mgba libretro-
fbalpha2012-neogeo libretro-fba-neogeo libretro-
beetle-vb libretro-tgbdual libretro-fba-cps1
libretro-fba-cps2 libretro-fmsx libretro-stella
libretro-yabause libretro-mess2014 libretro-
mess2016 libretro-desmume libretro-beetle-bsnes
libretro-glupen64 libretro-catsfc libretro-
quicknes libretro-bsnes-mercury-balanced libretro-
vbam libretro-bluemsx libretro-fceumm libretro-
nxengine libretro-snes9x-next libretro-mame2014
libretro-mame2016 libretro-fbalpha2012-cps1
libretro-fbalpha2012-cps2 libretro-fbalpha2012-
cps3 libretro-fbalpha libretro-snes9x libretro-
prboom libretro-beetle-gba -y
```

Ok, if one of these packages fails, remove it from the list, and try again. The command above is the exact command I use on my ODROID-C2s. For my ODROID-XU4s, I have been using the initial command listed

above. After that is done, do a little updating and cleaning.

```
$ sudo apt-get update -y
$ sudo apt-get upgrade -y
$ sudo apt-get autoremove -y
```

Now you should have this menu option available after the commands are done running, Applications -> Games -> Retroarch. Click on it and you should see something similar to what is depicted below.

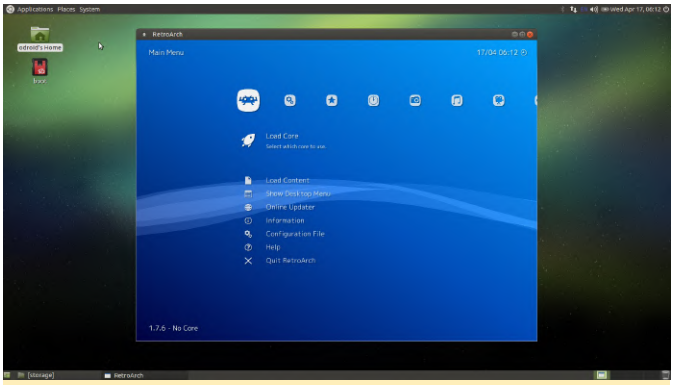


Figure 21

Do not mess with retroarch now, we will come back to it. Next we have to get antimicro installed so we can control everything with a gamepad when retroarch is not running. Go to this URL, https://github.com/AntiMicro/antimicro/releases and download the latest release as a zip file. You should see it below the Windows EXE entries. Once it is done downloading, open the ODROID home folder. There is a link on the desktop for it. Create a new folder called install_zips as depicted below.

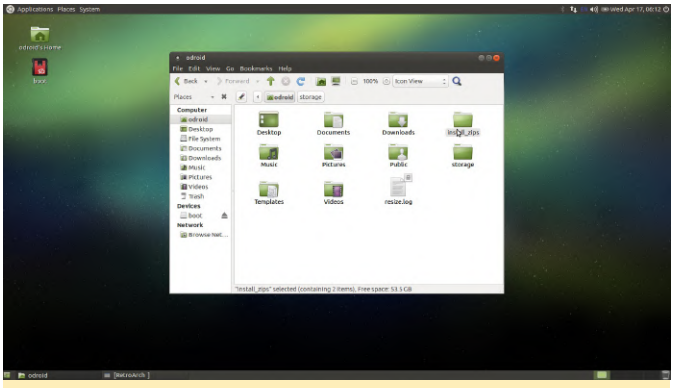


Figure 22

Now go to the downloads folder and copy the antimicro zip file from there and paste it into the install_zips folder. Right click on it and select Extract Here.

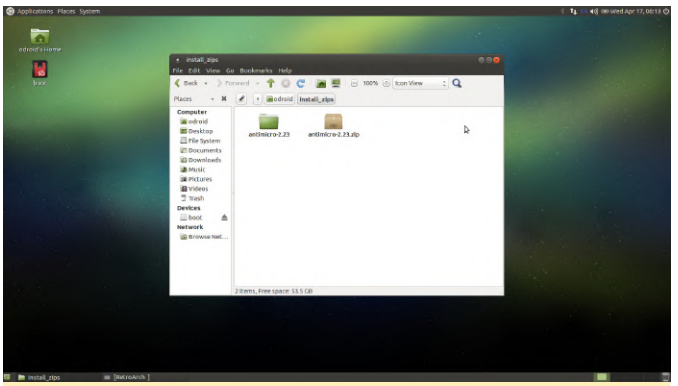


Figure 23

Now open up a terminal at, Applications -> System Tools -> MATE Terminal, and run the following commands. If your antimicro folder, after file extraction, has a different name use that name in the change directory command below.

```
$ cd install_zips/antimicro-2.23/
$ sudo apt-get install libsdl2-dev -y
$ sudo apt-get install qttools5-dev -y
$ sudo apt-get install qttools5-dev-tools -y
$ sudo apt-get install libxtst-dev -y
```

Once all those packages are installed we can compile antimicro without any errors. Run the following commands.

```
$ cmake .
$ sudo make
$ sudo make install
```

You should see something like the following during this process.

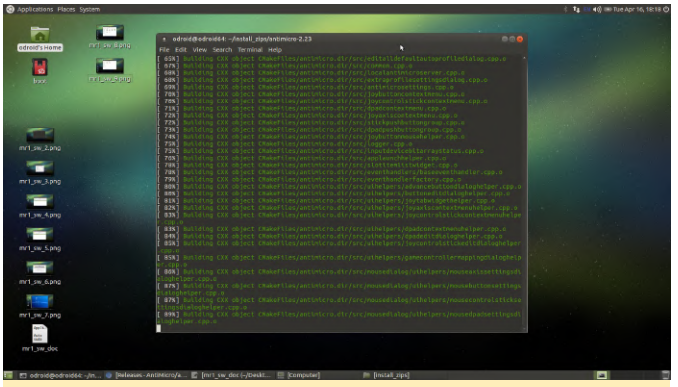


Figure 24

Once that is done, test antimicro, by typing run antimicro at the terminal. If everything is ok you should see something like what is depicted below.

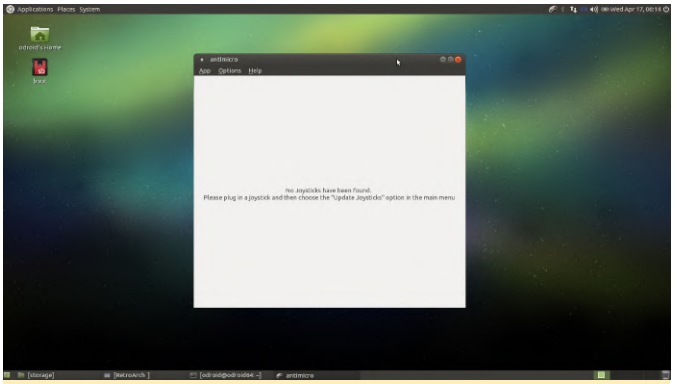


Figure 25

Plug in a game controller that is supported by Linux. Hardkernel has the best price I have seen for GameSir controllers. So, if you pick up some ODROID-GOs or other hardware grab one of these GameSir Wired Controllers. Start retroarch, Applications -> Games -> RetroArch and you should see large yellow text flash across the bottom of the screen, look closely. I have had some versions of the controller act a bit weird, but I have had no problems. Close retroarch. Now type into the terminal window the antimicro command. You should see something similar to what is depicted below with a properly detected controller.

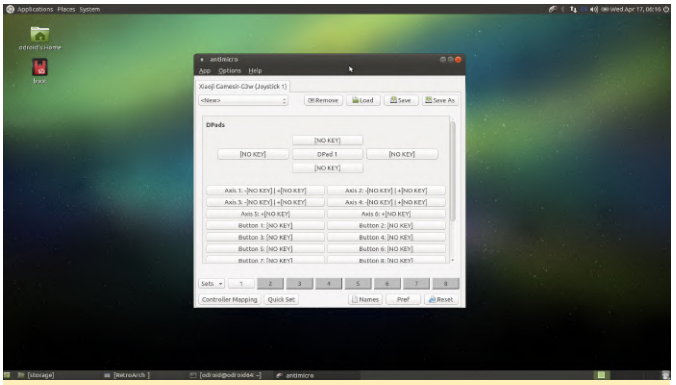


Figure 26

Next up we are going to configure antimicro, and retroarch.

Retroarch and Antimicro Configuration

Get antimicro configured first, so that we can start controlling the desktop environment with the gamepad. Open up a terminal, I will not list the menu path for it from this point forward. Type antimicro in the terminal and wait for the app to launch. Connect your linux supported controller and make sure that antimicro recognizes it. If it does not you will need to try another controller.

Click the Controller Mapping button on the bottom left hand corner of the UI. This is where you tell antimicro about the base functionality of your controller. If you do not have a button for a specific position in the list, for instance Linux seems to ignore the blue central button on the GameSir controllers, use your mouse to click down to the next viable option. Match up the buttons on the gamepad with the controller graphic's green button indicator.

ALERT: Some buttons act like triggers and fire multiple times. Use the mouse to back up the position of the mapping and fix the double entry. Click save when you are done and return to the main antimicro UI.

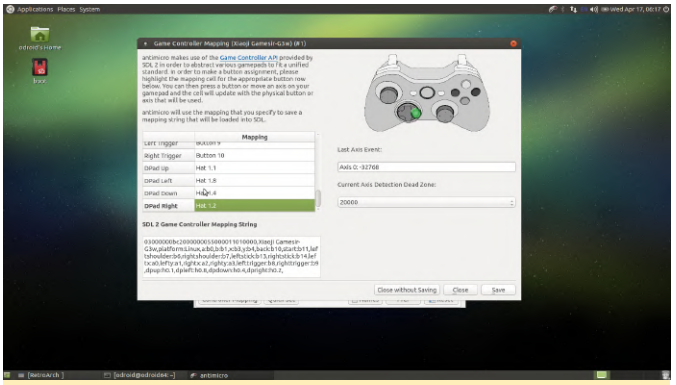


Figure 27

You will see a new mapping on the main antimicro UI that contains buttons for all the new mappings you just made. What we are going to do here is setup mouse support so that you can control the desktop environment from the gamepad when retroarch is not running. We will use the left thumbstick for precise, slower, mouse control and the right thumbstick for faster, coarse mouse control. The A and B buttons will serve as the left and right mouse buttons. Right click on the left thumb-stick area and select mouse normal from the option list.

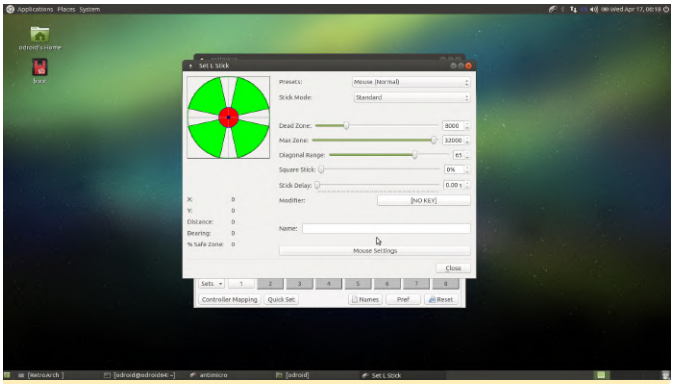


Figure 28

Click on the left thumb, tick buttons again, and find the Mouse Settings button at the bottom of the window. The image above shows the button. In the mouse settings window, set the Horizontal Speed and Vertical Speed to 10 for the left thumbstick as depicted below.

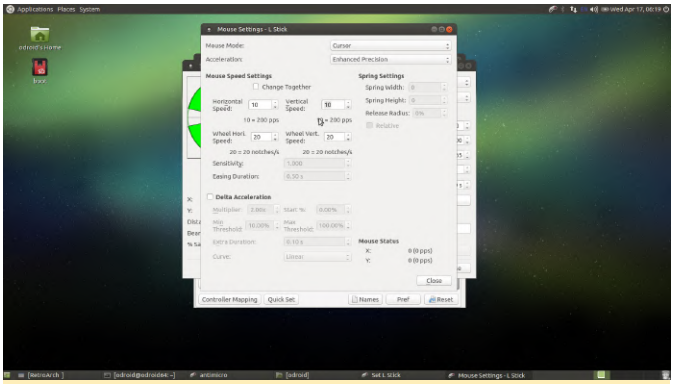


Figure 29

Do the same thing for the right thumbstick except now set the Horizontal Speed and Vertical Speed to 30 as depicted below.

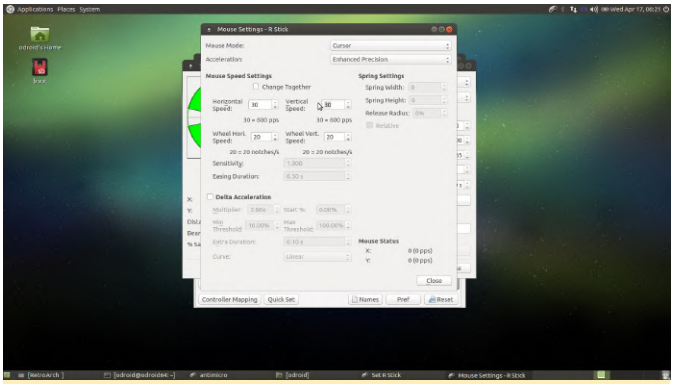


Figure 30

Now let us map the mouse buttons, close all dialogs and get back to the main antimicro UI. Find the A button in the button list below the thumbstick and dpad listing. Click on it then click on the Mouse tab. Select the left mouse button. Do the same thing for the B button except choose the right mouse button for that mapping. Below is a screen shot depicting the left mouse button mapping in action.

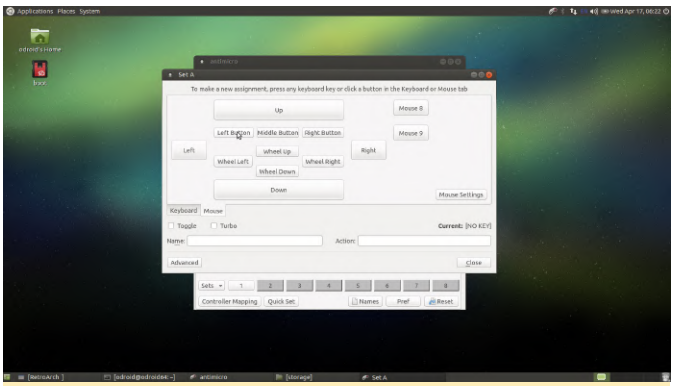


Figure 31

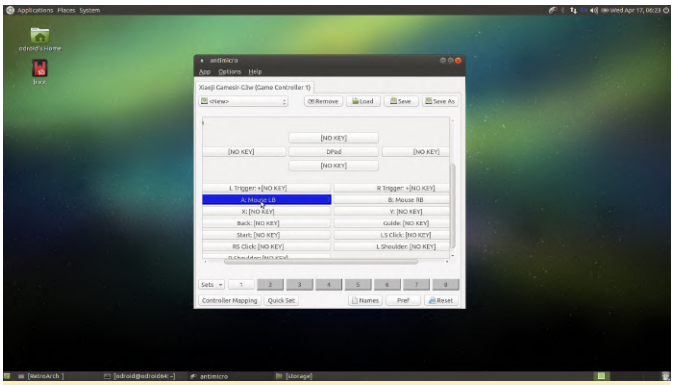


Figure 32

Take it for a spin while the main antimicro UI is open. You should see the mouse move around the screen as the button listings in the antimicro UI turn blue to indicate they are active. Adjust the speeds of the mouse controls according to your personal preferences.

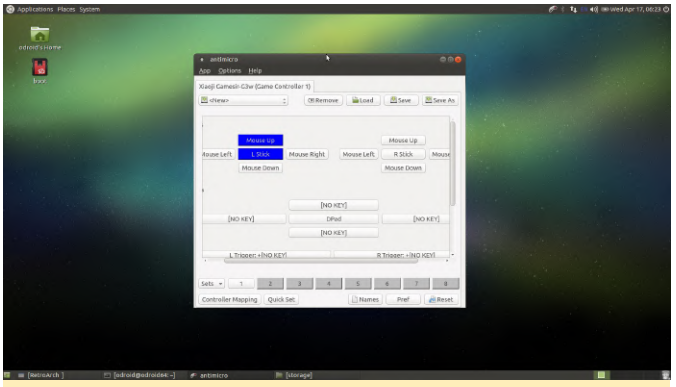


Figure 33

When you are all set, go back to the antimicro main UI and click the Save As button at the top right hand side of the screen. Save the controller configuration as game_sir_wired.xml or whatever you want to name your controller in the odroid home directory as shown below.

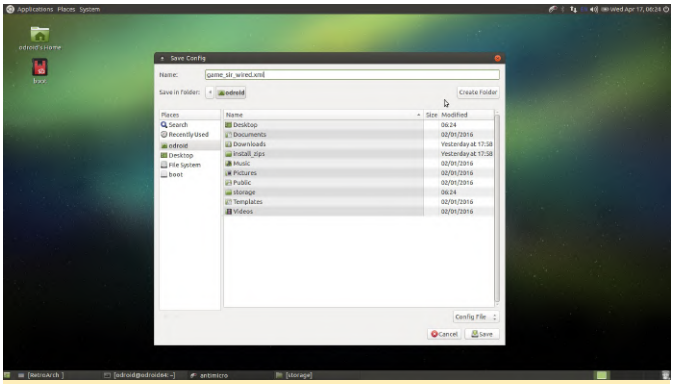


Figure 34

Click on Options -> Settings in the antimicro menu and make sure only Close To Tray and Launch In Tray are checked. This will ensure antimicro lives in the app tray and does not clutter up our screen. We have one more setting to adjust and then we will be done with antimicro and on to retroarch!

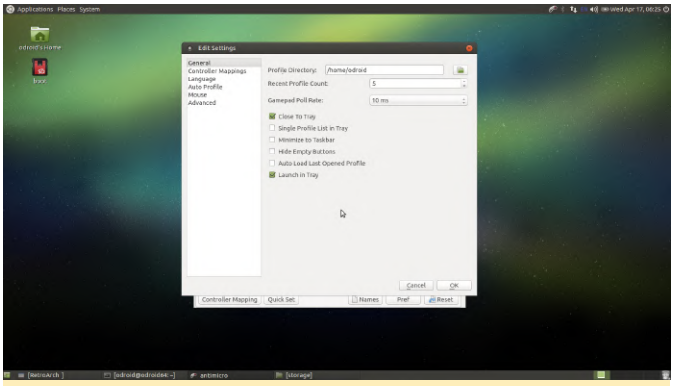


Figure 35

While still on the antimicro setting window click on the Auto Profile option on the left. This will determine what profile will automatically be associated with the attached gamepad. You only get one mapping. Click the Active checkbox at the top of the window. Then select the Default row in the table. Click the Edit button and browse to the controller mapping XML file you saved just a few steps back. Click Ok then quit out of antimicro, if it appears in the system tray click the controller icon in the system tray and quit the app. We are done with the antimicro configuration!

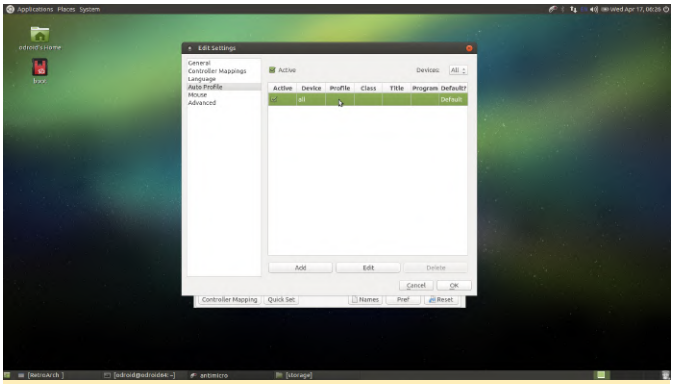


Figure 36

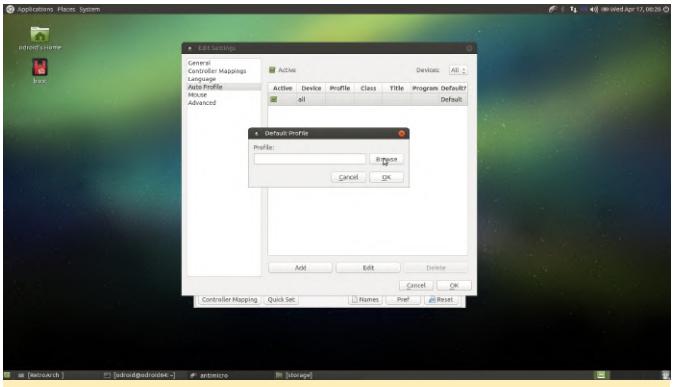


Figure 37

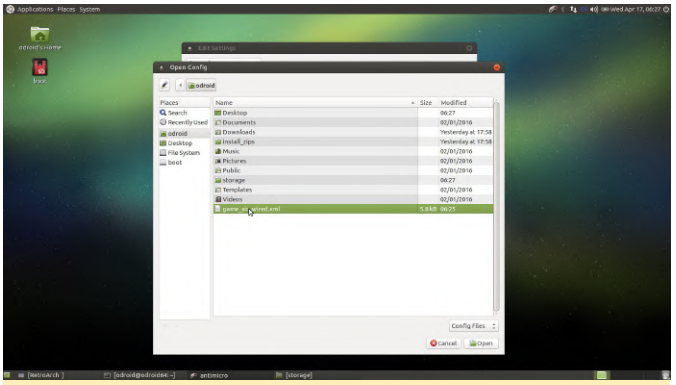


Figure 38

Next up, fire up retroarch from the menu system, I will not list the menu path for it from this point forward. First Let us get the gamepad working in retroarch. In retroarch, you can use the keyboard arrow keys, enter, and backspace to navigate the menu system without the gamepad. Make sure you have a mouse, keyboard, and game controller connected to your ODROID. First thing we will do is get the controller working. Use the arrows on the keyboard to navigate left to the Settings section, the move down to the Input section as shown below.

Figure 39

Adjust the settings on this screen as you see below. I usually set the max number of controllers to 4 since there are 4 USB ports. I like the "L1 + R1 + Start + Select" Menu Toggle Gamepad Combo setting. Leave the remaining settings and scroll down to the User 1 Binds. You will have to setup each user input.

Tip: Map the A and B buttons by name not position. If you are using a GameSir controller, map the colors green and red to positive/select, negative/back button usage. It is just what I like to do, you can map 'em anyway you like!

Figure 40

Figure 41

The next step takes a little while but requires very little work on your part. You just have to click on a few things and wait for them to complete. Navigate back to the Main Menu which is the first section retroarch starts on. Make sure you're connected to the internet. Scroll down to the Content Updater and open that section.

Figure 42

I usually run each option in order. Try the Core Updater option first.

Tip: You may have to start retroarch from the terminal: sudo retroarch; when updating core files. If you run retroarch with a -v for verbose output and you see an error message during the Core Update command's execution, then you should close retroarch and relaunch it with the sudo command.

Next go to the Thumbnails Updater, this part can take a while, and select each system you plan to load ROMs. Resist the urge to click a bunch in a row. I have done that and it can crash the app or slows down each individual process as they are all competing for resources. Best to let one blast through then move on.

Tip: There are some one-off games you can get thumbnails for here like DOOM Demo, Cave Story, and Dinothawr. You will be able to load those games from the Content Downloader.

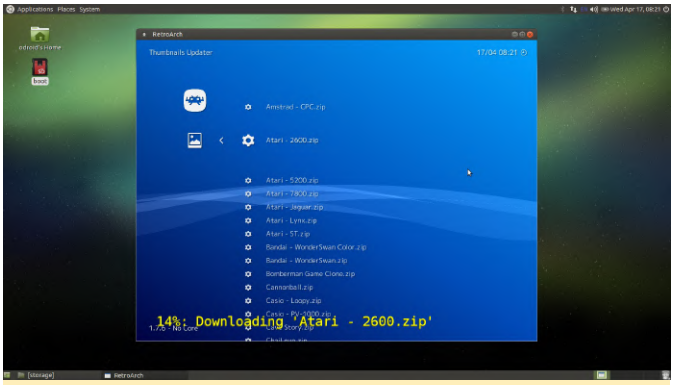


Figure 43

Next you will want to back out of the Thumbnail Updater section with the backspace key. Scroll down to the Content Downloader and download any special content you want from the listing.

Tip: If you see a screen that has no options wait a minute, back out, and go into that section, again. It might not have finished downloading the index of available options. I usually do not install the one-off games, but this is the spot for it. Most of the content I think you can safely ignore, but I will leave that up to you. Run the following list of updates one at a time.

- Update Core Info Files
- Update Assets
- Update Joypad Profiles
- Update Cheats
- Update Databases
- Update Overlays
- Update GLSL Shaders
- Update Slang Shaders

Close retroarch. Next up, load some ROMs. I own a copy of Contra for the NES and I have a version of it in ROM form. Open the odroid home folder using the link on the desktop. Right click on the folder and create a new directory, name it ROMS. Open the ROMS folder and create a new folder and name it NES. Obviously you would substitute the NES folder for whatever system you intend to load ROMs. Not every single system is supported and some emulators have caveats for BIOS files that need to be copied, etc. Many of the older systems are well supported and easily added. The ODROID-C1+ can probably handle SNES and Genesis, but I just like to use it for 8-bit games. The ODROID-C2 certainly has enough power for 16-bit systems and I like to use it for 8-bit, 16-bit, and all of the handheld systems.

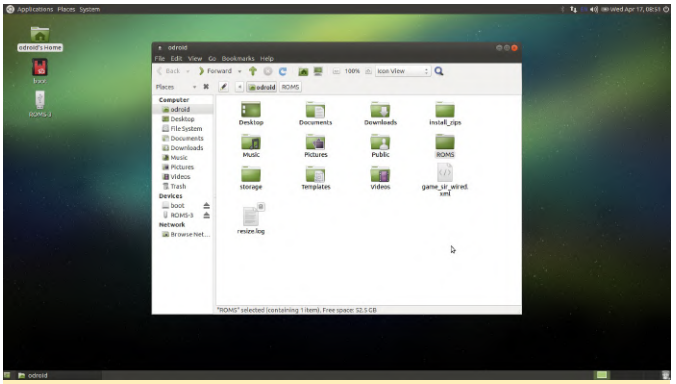


Figure 44

Once you have all your ROMs installed, we can start telling retroarch about the files. Fire up retroarch and scroll to the right until you see the Scan Directory option.

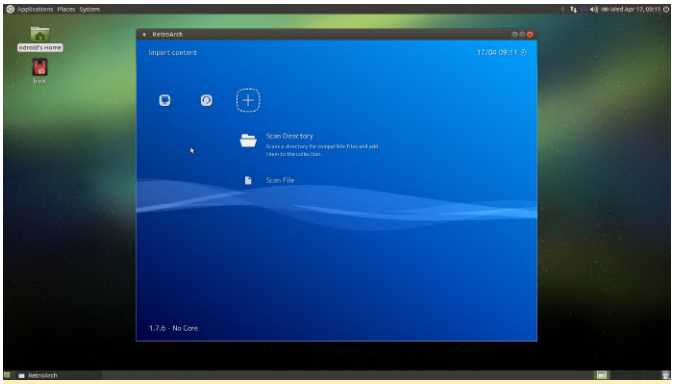


Figure 45

Navigate to the ROMS folder and you should see a list. Then select one system at a time for scanning. Scanning large ROM sets can take a while. Again resist the urge to start multiple scans, things will go smoother if you let an operation complete before starting a new one. In this case we only have one ROM, Contra, so it will complete the scan instantly.

Figure 46

Back out of the Scan Directory section and you should now see a little NES controller on the section listing. Scroll to it and now you should see Contra, or

whatever ROMS you were trying to load. Load up a game by selecting it and then selecting a target emulator. Your game should load.

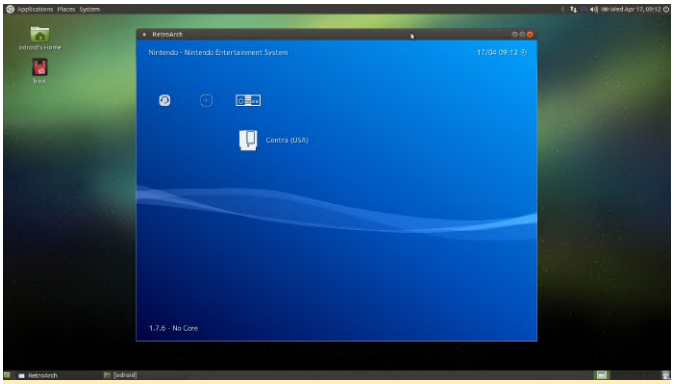


Figure 47

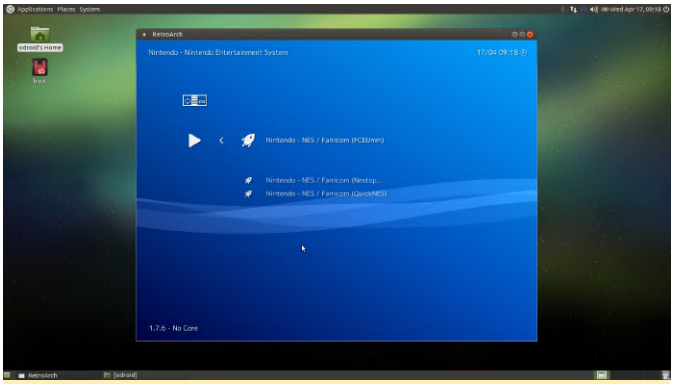


Figure 48

Figure 49

Retroarch will not recognize every single ROM for every single system. Also you may want to try different emulators on troublesome ROMs. That is more advanced stuff and I will cover that in the next tutorial where I will wrap everything up.

Reference

http://middlemind.com/tutorials/odroid_go/mr3_build.html