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

ODROID Magazine

Year Six
Issue #70
Oct 2019

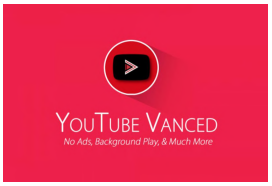
SPEED BEYOND LIMITS

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 ▶

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 ▶

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!

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.

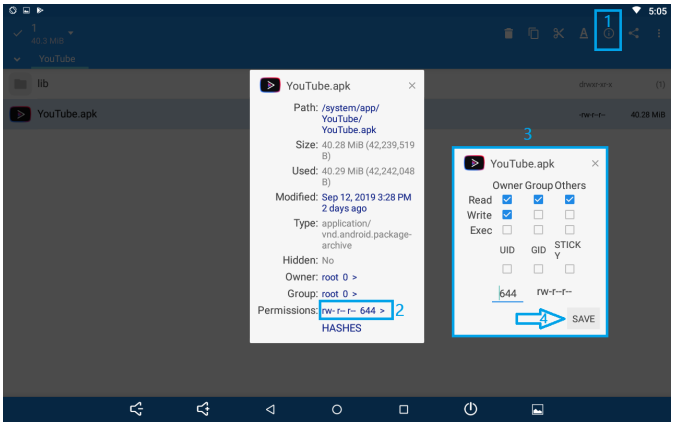
Step 14: Open MiXplorer, Click the 3 lines in the top left corner and click Root. Supersu will ask if you want to allow Root access. Click grant access forever, then go ahead and grant it access.

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.

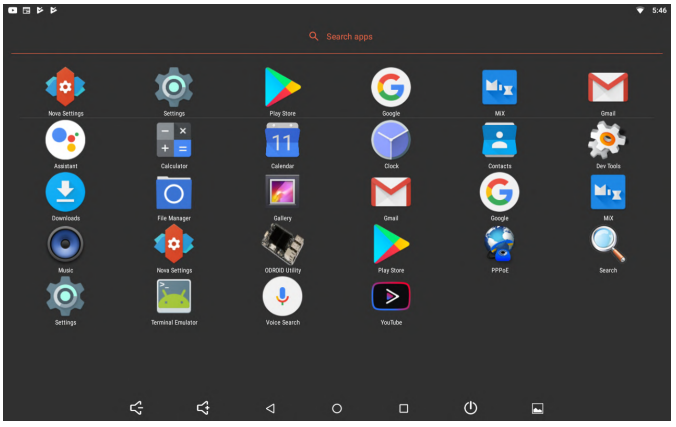
Step 17: Go back to /System/app/Youtube/ and Click the first icon in the top right, the Clipboard, and click Copy. And it will copy the Youtube.apk and the libfolder into /system/app/Youtube/. So it should look like /system/app/Youtube/Youtube.apk /System/app/Youtube/lib/armeabi-v7a/ bunch of .so files.

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.

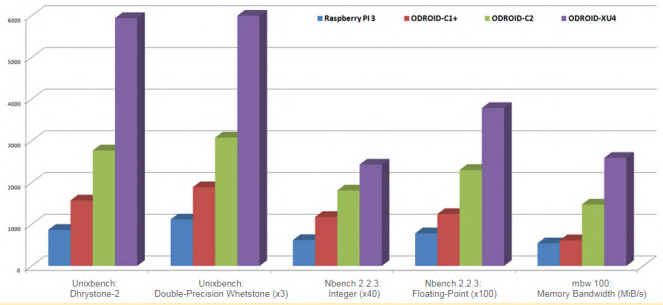


Figure 1 - ODROID performance comparison

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

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 4 - Case clip close up

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.

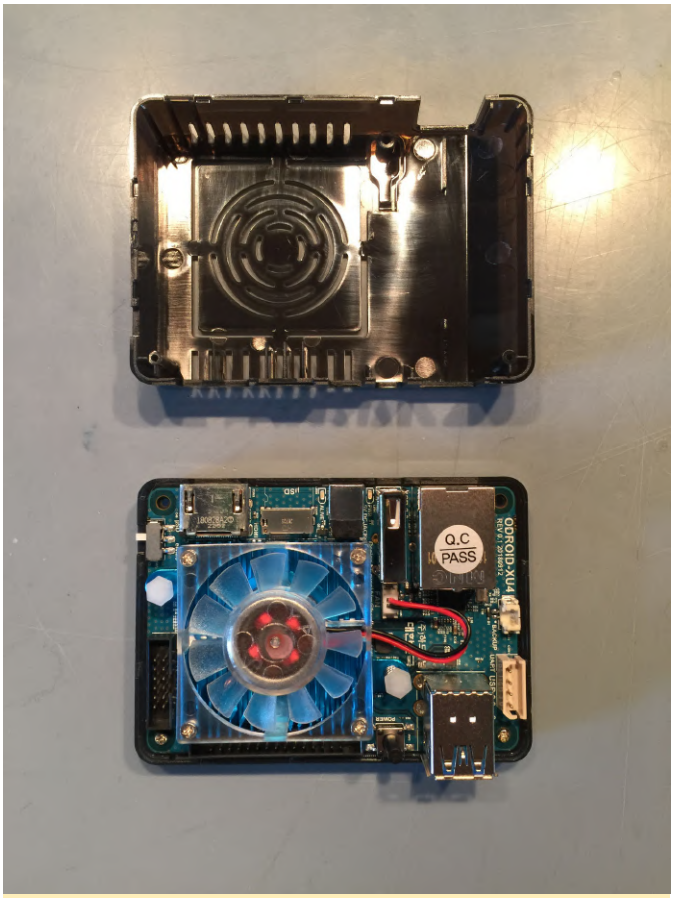


Figure 7 - ODROID-XU4 resting in the bottom of the case

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.

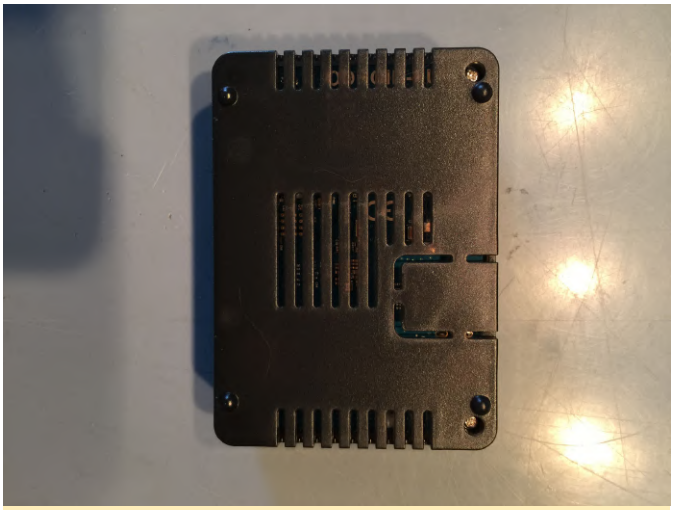


Figure 8 - Screw together the case

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.

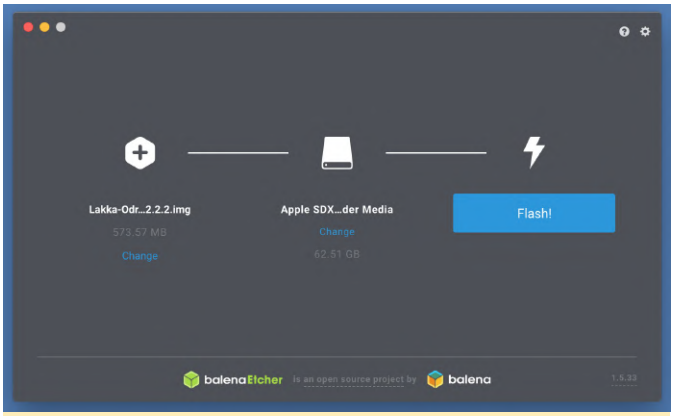


Figure 10 - Screenshot of etcher setup to flash an SD card

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.

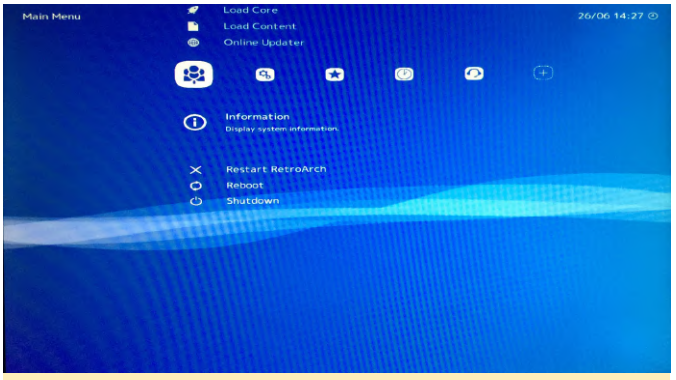


Figure 11 - Lakka Information entry

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

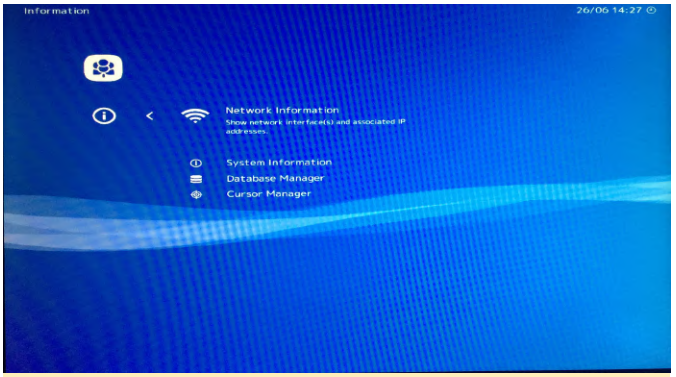


Figure 12 - Lakka network information

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.

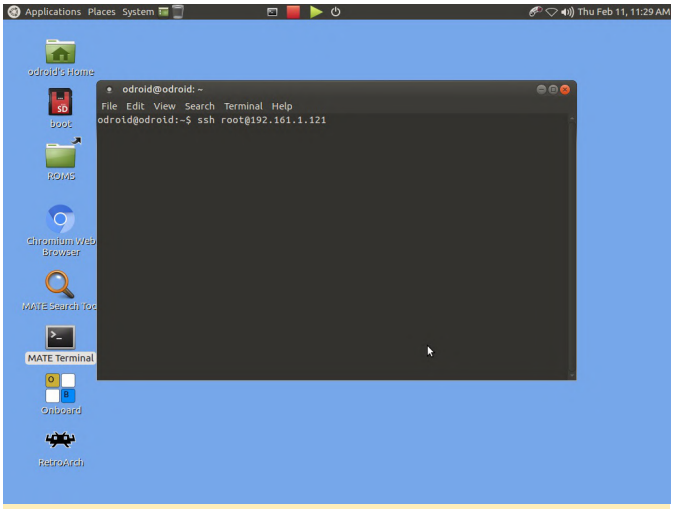


Figure 13 - SSH command from macOS or Linux

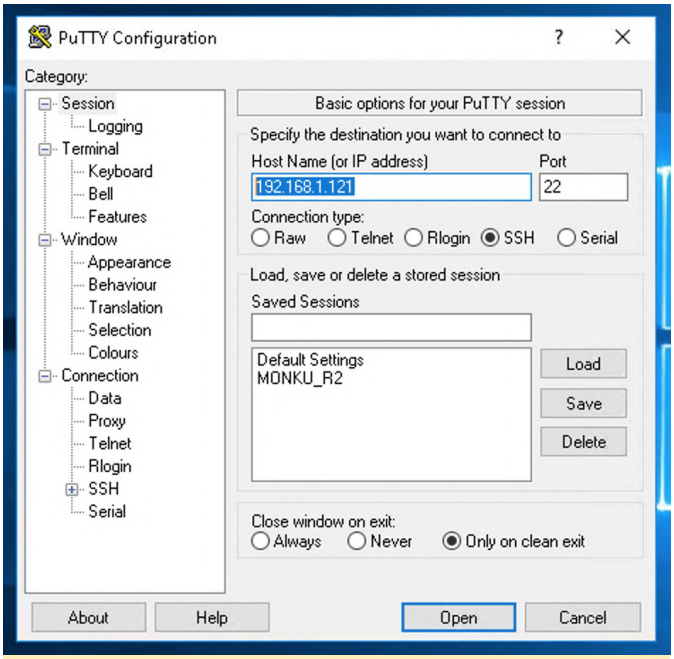


Figure 14 - Putty on Windows

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.

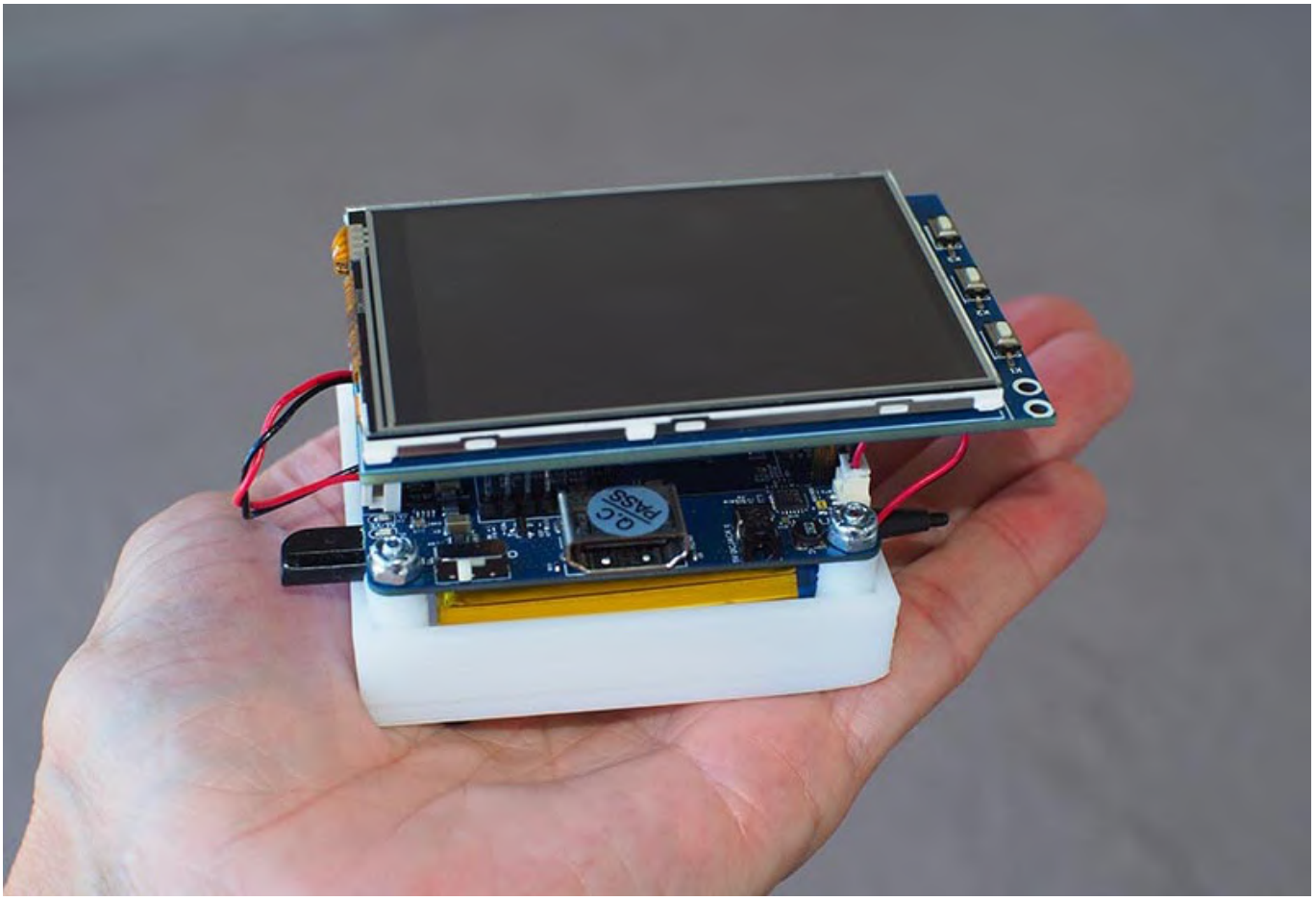
This article was taken from middlemind.com, for more information please visit the original website or Lakka at the following links:

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

www.lakka.tv

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.

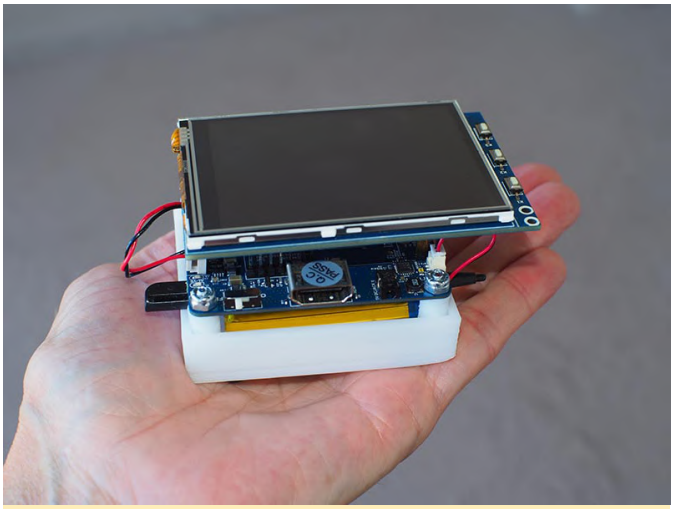


Figure 1 – A "pocketable" palm-sized Linux computer.

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.

(

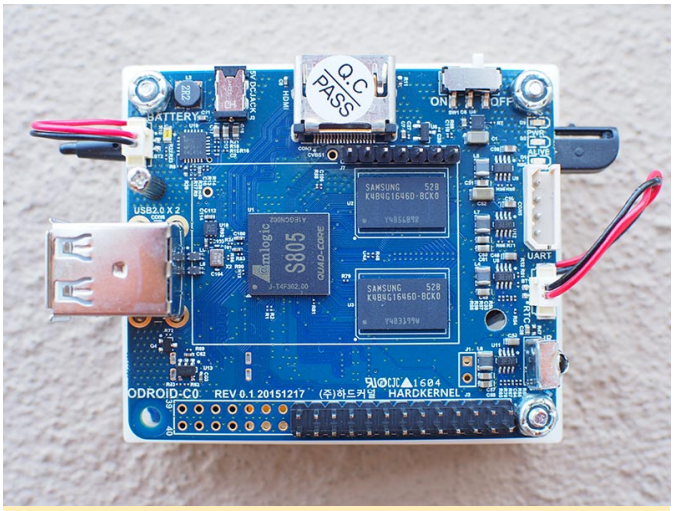


Figure 2 - All of the soldering is now complete. Notice that the dual row header has only been soldered to 26 GPIO pins.

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

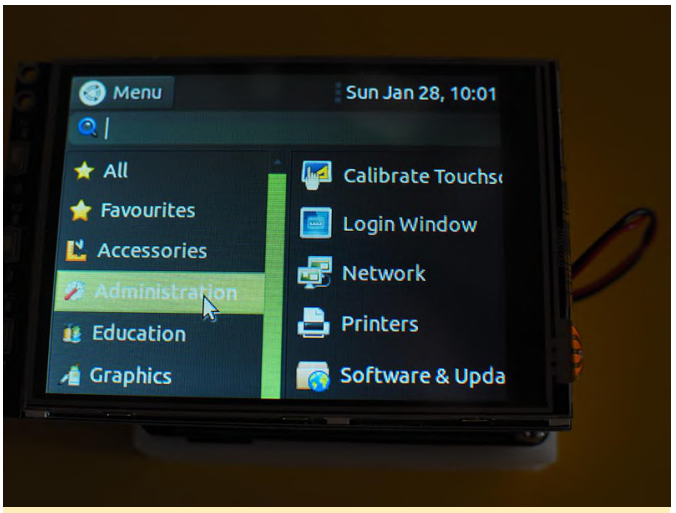


Figure 3 - After you've completed the Hardkernel C1 shield wiki, look for the touchscreen calibration app inside the Ubuntu menu. Run this app for finalizing the touchscreen shield setup.

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.

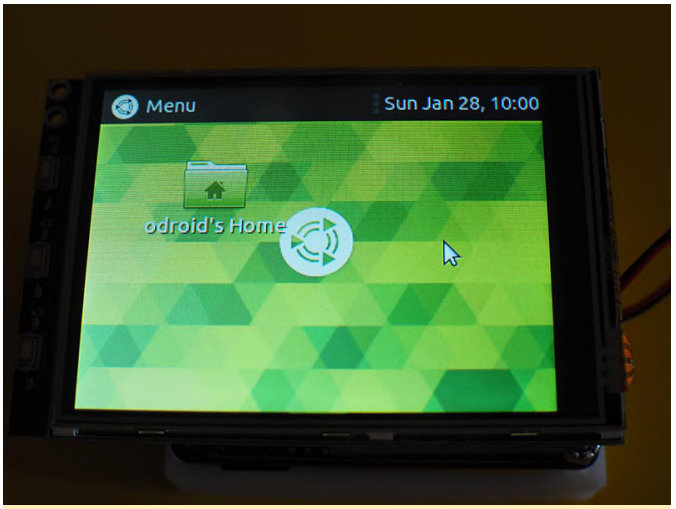


Figure 4b - There isn't a lot of screen real estate on the touchscreen shield. Use the Notes (i.e., numbers 3 and 4) below for maximizing your little screen's potential.

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.

gamescom
The Heart of Gaming

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).

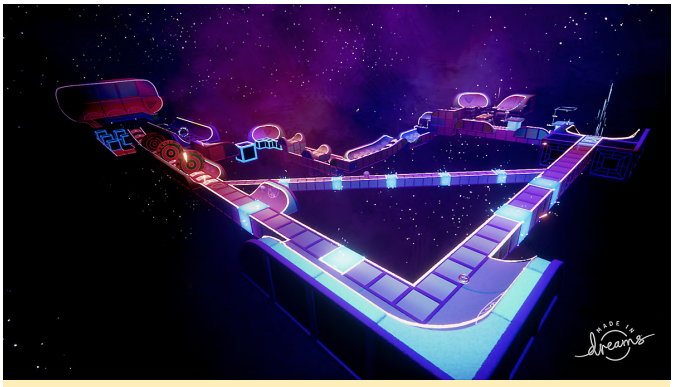


Figure 2 - No, you're not dreaming, Sony Interactive Entertainment Dreams title is a worthy winner of the "Best of gamescom" award. Image courtesy of Sony Interactive Entertainment.

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.

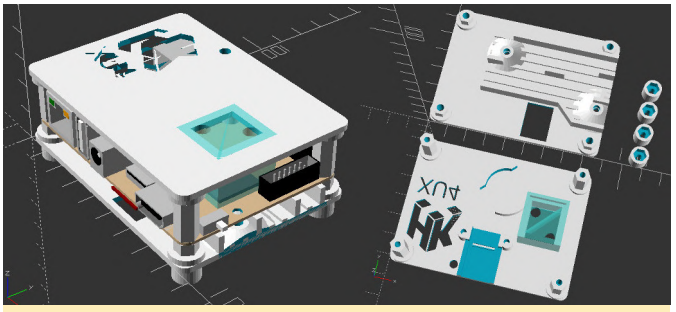


Figure 1 - ODROID-XU4 SBC Water Cooled Case Design with RTC and UART holder

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.

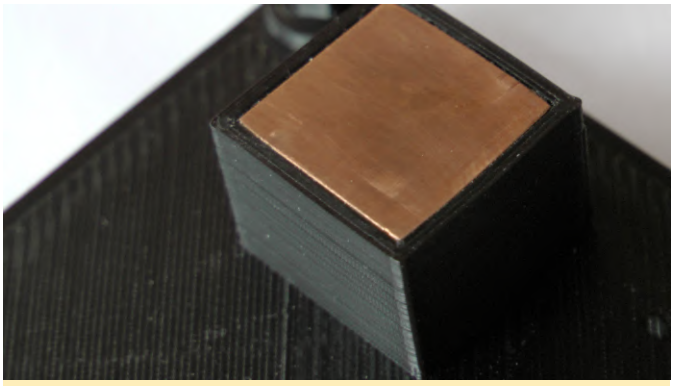


Figure 2 - ODROID-XU4 Water Block

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 is 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
```

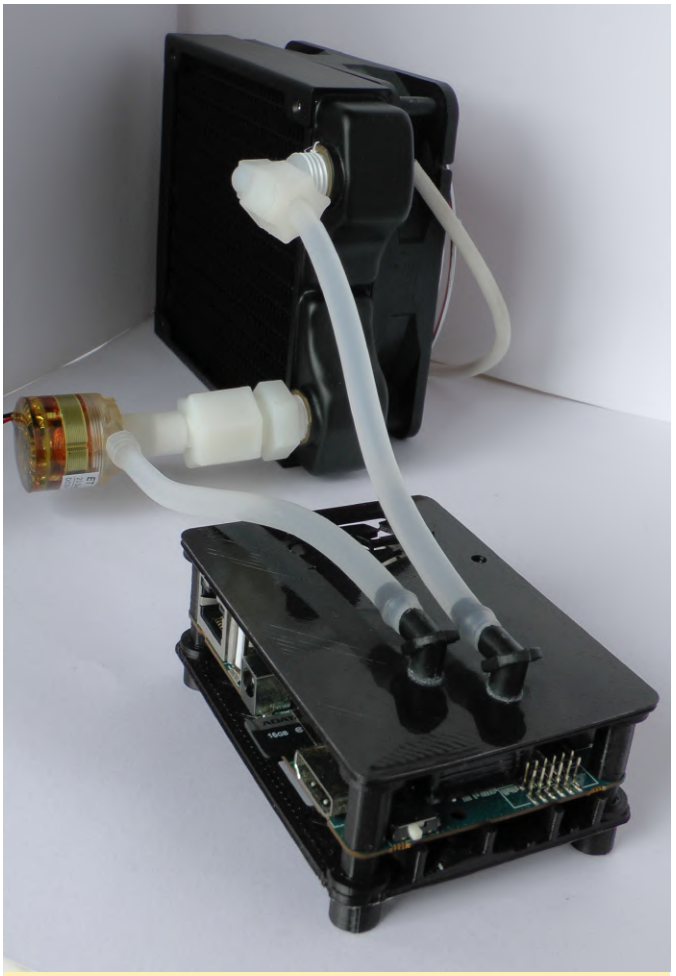


Figure 5 - Waterblock Test Bed

Kernel compile at 2ghz @ 71F(21.66c) Ambient Temperature real 25m12.282s user 172m3.503s sys 16m48.678s

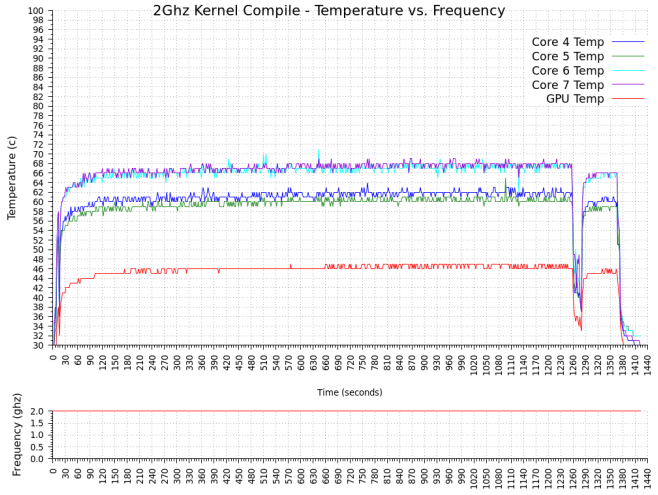


Figure 6 -Kernel Compile Test 2ghz

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.

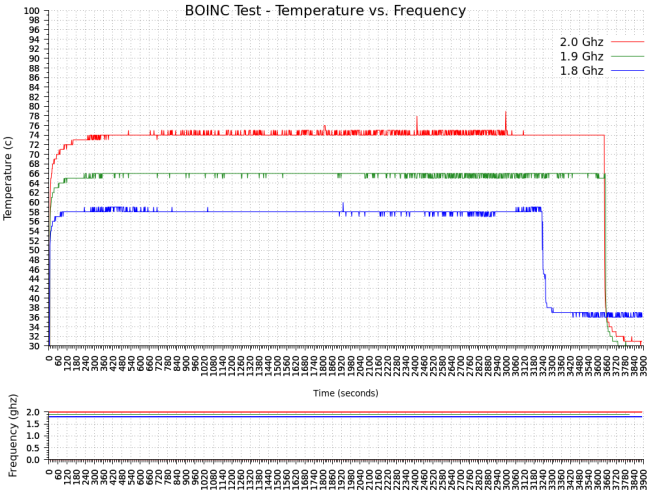


Figure 7 - BOINC Test at 2ghz, 1.9ghz and 1.8ghz

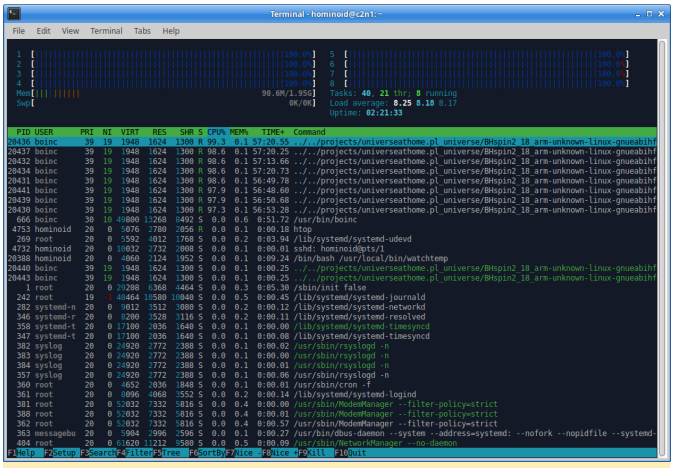


Figure 8 - BOINC htop

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

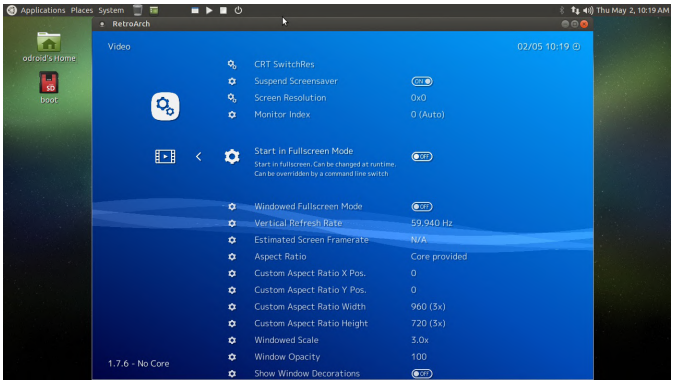
Introduction and Tutorial Goals

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

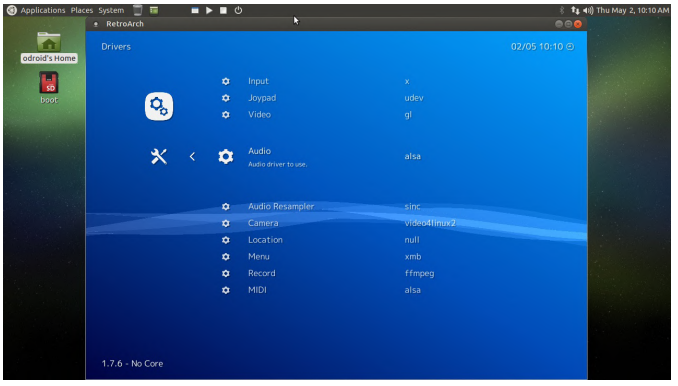
with no sound? The WiFi adapter is also optional if you do not plan to do any web surfing on the nearby cafe's customer WiFi network, otherwise, it is a perfect addition to our stealthy retro gaming console. Alright, let us get started.

Setting Up the Audio

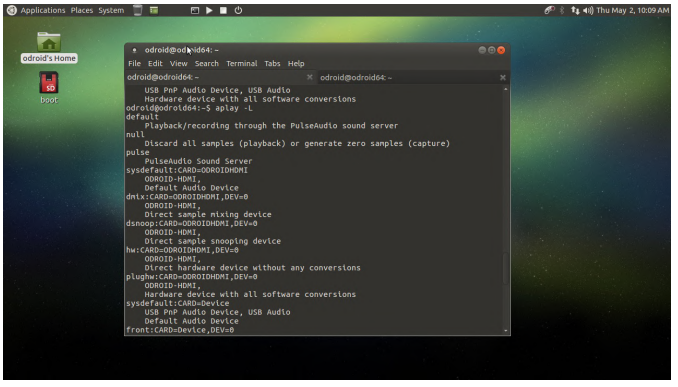
First things first, let's turn off full screen mode in retroarch. Launch retroarch, if needed, navigate to the Drivers section and scroll down to the Video option, scroll down to the Start in Fullscreen Mode option, and set it to off. Use this same process and turn it back on when we are done adjusting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

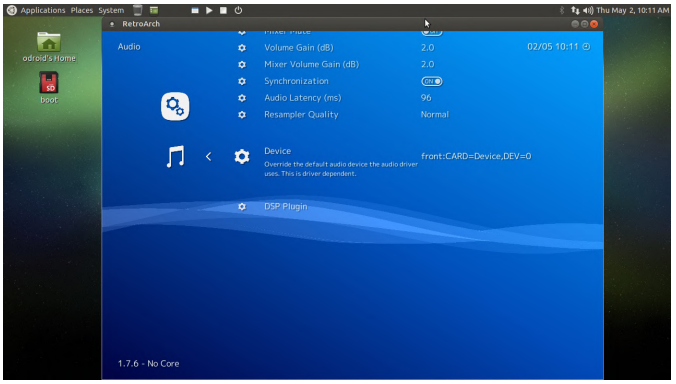
Once these changes have been made, write some scripts that utilize the files we have just created. Use your favorite editor to create the following files.

```
#!/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
```

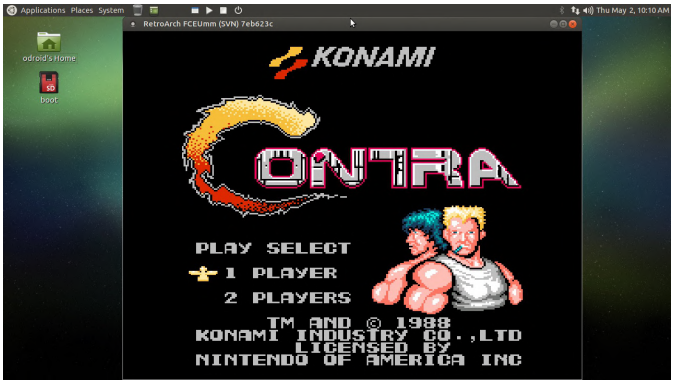
Let's set the proper permissions for these files. Run the following commands at the terminal.

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

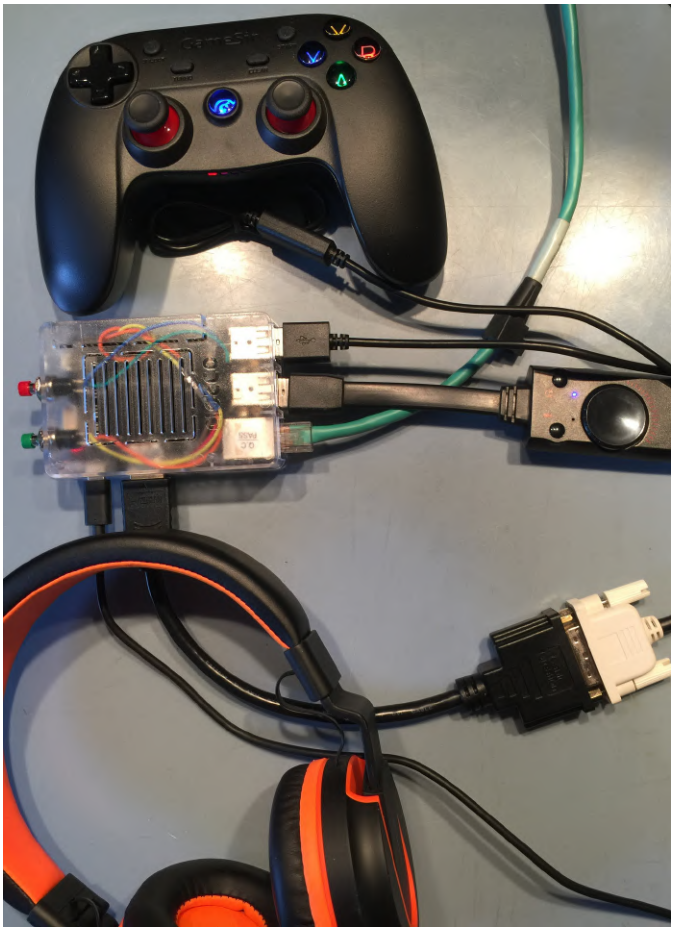
Run the set_vga_audio script you just created and get ready to test out your retro gaming console on a VGA screen with USB audio adapter. Fire up retroarch and verify the Device setting.



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.

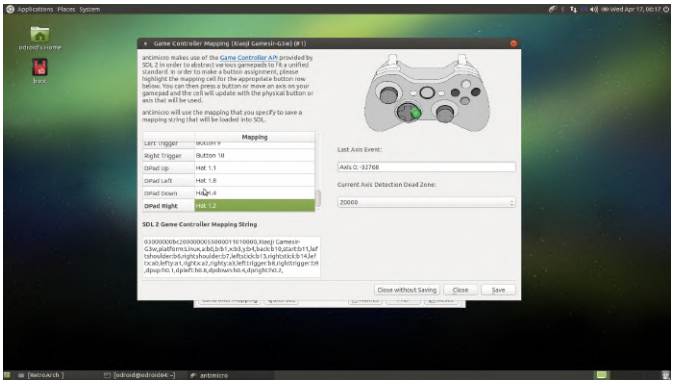


A slightly less stealthy Cubicle Commando setup but it gets the job done.

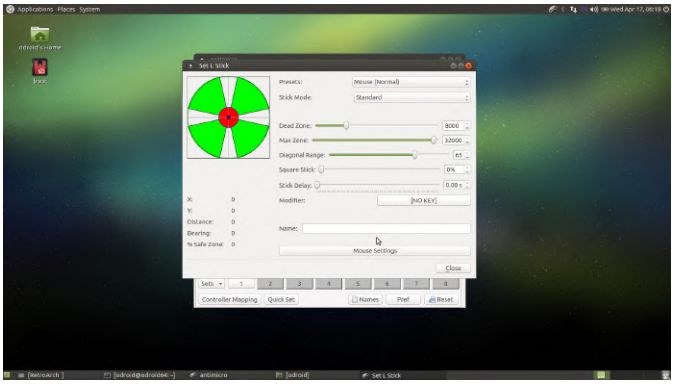


Setting Up the Controller

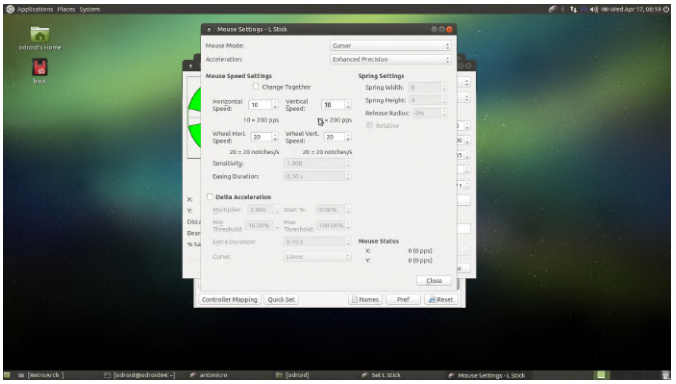
I will assume that the GameSir wireless controller has not been previously configured for your retro gaming console. This next part will walk you through configuring a new controller for a Monku Retro 1, 2, 3 (ODROID-C1+, -C2, -XU4) device. Let us get antimicro configured so we can start controlling 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.



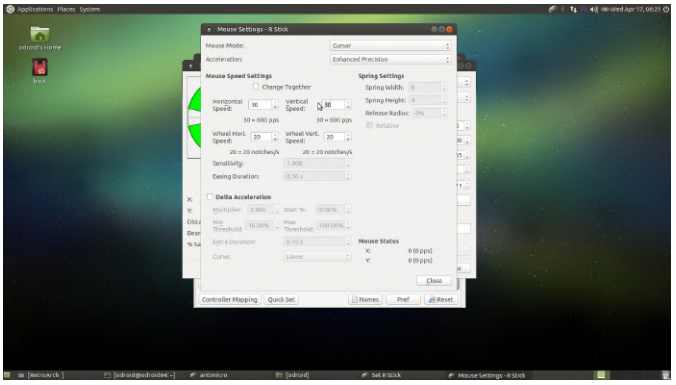
You will see a new mapping on the main antimicro UI that contains buttons for all the new mappings you just made. What we are going to do here is setup mouse support so that you can control the desktop environment from the gamepad when retroarch is not running. We will use the left thumbstick for fine-grained, 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.



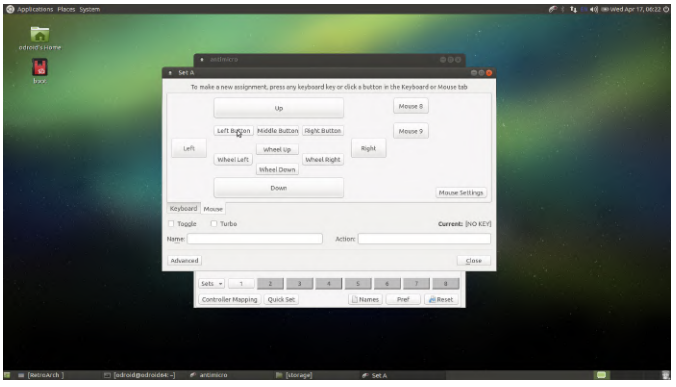
Click on the left thumbstick buttons again and find the Mouse Settings button at the bottom of the window. The image above shows the button we are looking for. In the mouse settings window, set the Horizontal Speed and Vertical Speed to 10 for the left thumbstick as depicted below.

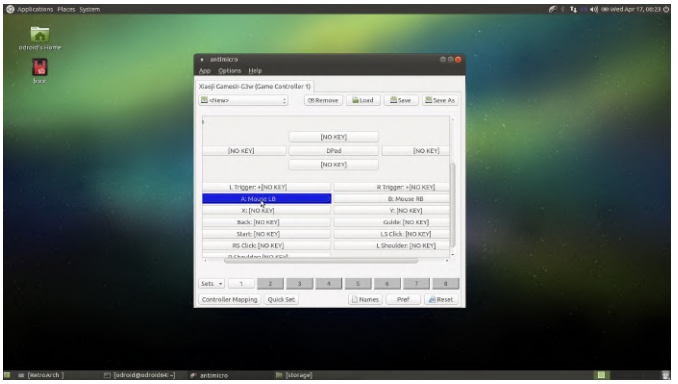


Do the same thing for the right thumbstick except set the Horizontal Speed and Vertical Speed to 30 as depicted below.

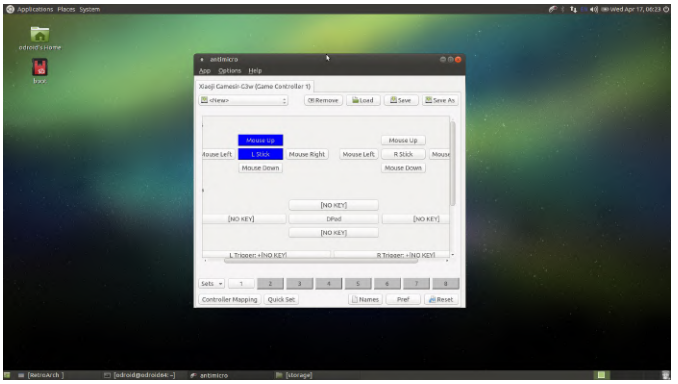


Now let us map the mouse buttons, close all dialogs and get back to the main antimicro UI. Find the A button in the button list below the thumbstick and dpad listing. Click on it then click on the Mouse tab. Select the left mouse button. Do the same thing for the B button except choose the right mouse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

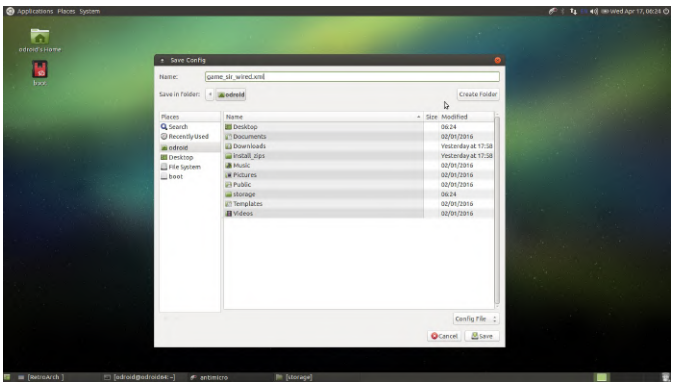




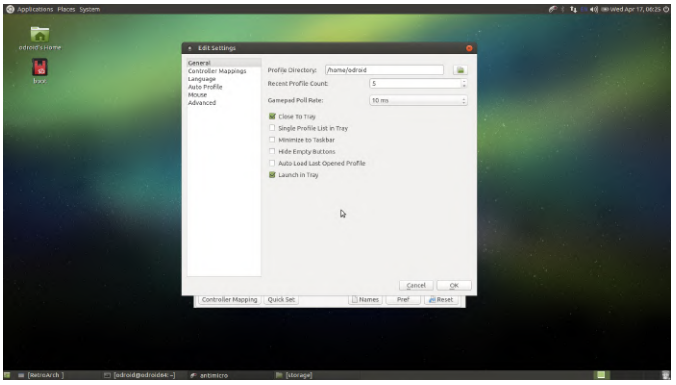
Take it for a spin while the main antimicro UI is open. You should see the mouse move around the screen as the button listings in the antimicro UI turn blue to indicate they are active. Test how it feels, adjust the speeds of the mouse controls as you see fit.



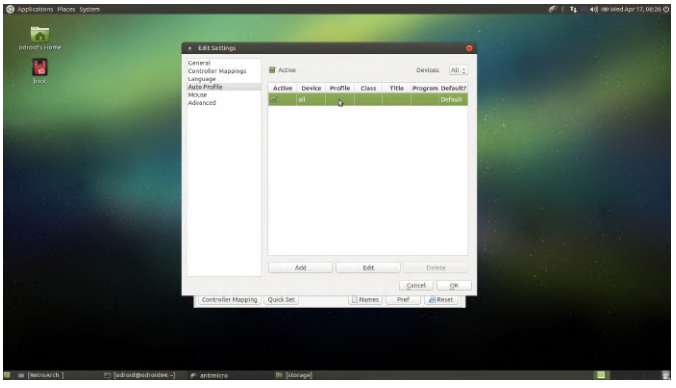
When you are all set to go back to the antimicro main UI, click the Save As button at the top right hand side of the screen. Save the controller configuration as game_sir_wired.xml or whatever you want to name your controller in the ODROID home directory as shown below. I will provide a copy of my XML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

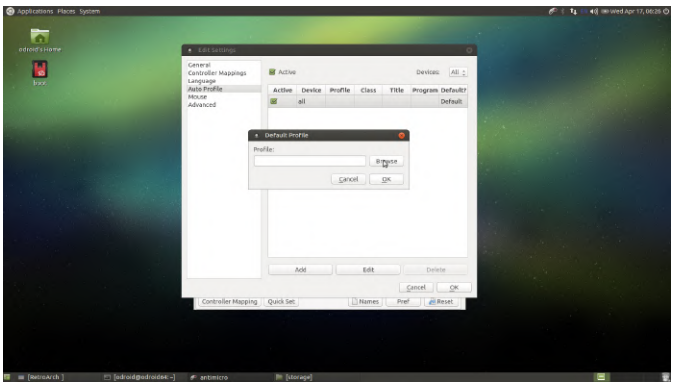


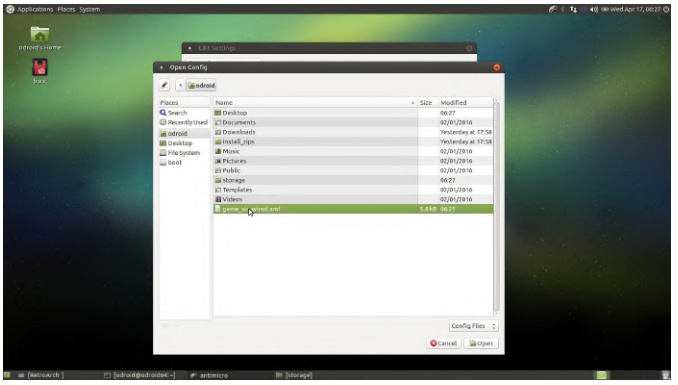
Click on Options -> Settings in the antimicro menu and make sure only Close To Tray and Launch In Tray are checked. This will ensure antimicro lives in the app tray and does not clutter up our screen. We have one more setting to adjust and then we will be done with antimicro and on to retroarch!



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 +

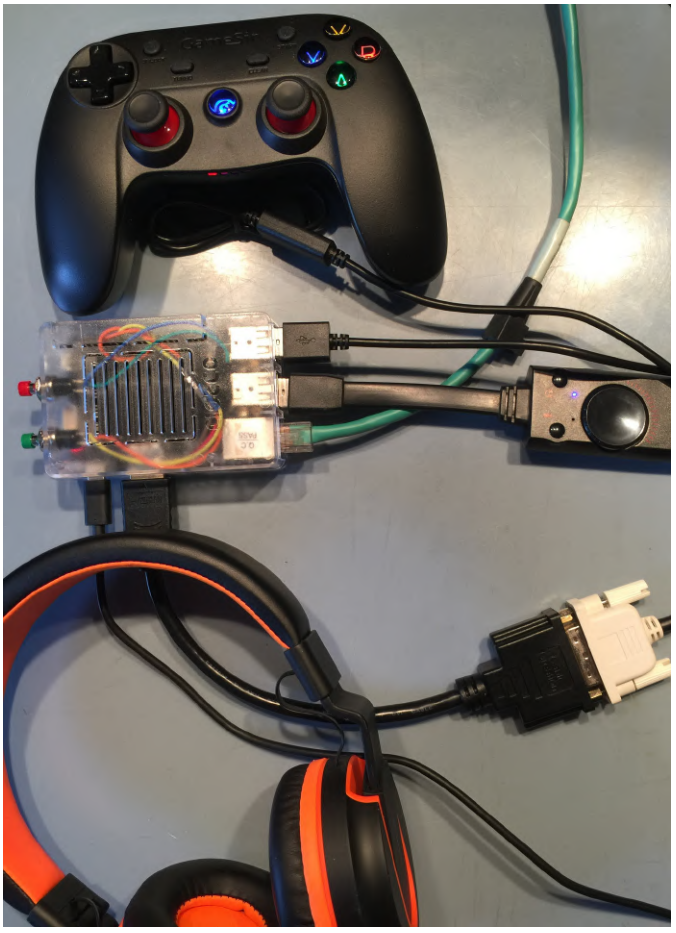
Select" Menu Toggle Gamepad Combo setting. Let's face it, if you're accidentally hitting this combination during game play something isn't right. Leave the remaining settings and scroll down to the User 1 Binds. You will have to setup each user input in this way it is not too bad and only takes a minute. 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