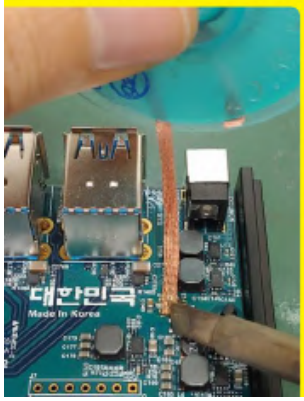
Power Outage • Basic NAS • ODROID KVM • Privacy • Verium • G Spot

ODROID Magazine

Year Six
Issue #72
Dec 2019

A FULL-FEATURED COMPUTING EXPERIENCE: *Mobile* WORKSTATION



ODROID-N2 REPAIRING:
THE BASICS OF BRINGING YOUR HARDWARE BACK

ODROID BLADECENTER:
HIGH PERFORMANCE COMPUTING STATION

DEEP LEARNING AI:
USING OPENVINO AND OPENCV WITH THE ODROID-C2

Cryptocurrency Mining: Earning Verium Coins With Your ODROID

December 1, 2019

Odroid XU4's, developed by Hardkernel Co. Ltd, are quite popular. SBC systems in general are designed to be very power efficient, because of this they are pretty good at Verium mining.

Protect Your Privacy: Why You Should Probably Make Your Own Home Automation Devices

December 1, 2019

With all of the hacks, vulnerabilities, data leaks and other disheartening news surrounding cloud services, cloud products, and big data companies, you may want to consider putting in the effort to create your own version of these solutions. "Smart" doorbell Have you read the recent news that the "smart doorbell" ▶

Mobile Workstation: Using An ODROID-N2 To Create A Full-Featured Computing Experience

December 1, 2019

Because I somehow managed to lose my BIOS password and lock myself out, I had to part with my old and beloved laptop, the one running Debian, but instead of just jumping to the latest Thinkpad, I wanted to see if I could somehow mix those cravings mentioned above: a ▶

KVM: Fun with virtualization on the ODROID-H2

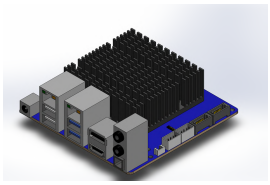
December 1, 2019

When it comes to the ODROID-H2, my use case for it is to work as a virtualization host, to run a few test VMs to run software on, or test upgrades of systems for my company (e.g. upgrading from older OS versions to newer versions). Looking at the forum, I ▶

The G Spot: Your go-to destination for all things that are Android Gaming

December 1, 2019

Our long wait is finally over: Google Stadia, this universal game-changing streaming service, has gone live now! It debuted late November, 2019. As discussed in earlier articles from this column, on this launch date, Google Stadia will ONLY be available to subscribers of the Stadia Founder's Edition. As you may ▶



Building An ODROID-H2 BladeCenter: Create A Micro-footprint High Performance Computing Station

December 1, 2019

After receiving inspiration from the excellent OpenSCAD H2 Model posted at https://forum.odroid.com/viewtopic.php?f=172&t=33824, I created a remix of the fantastic Raspberry Blade Center to house 3 ODROID-H2 units. I made the following changes to the original project: Migrated the files to SolidWorks Built an assembly Aligned the fasteners Widened the cart ▶

Surviving A Power Outage: Operating e-Commerce Business During Regional Power Outage

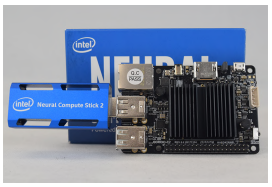
December 1, 2019

Imagine if the electricity to your business was cut off. But not just your business -- your city, your county and your entire region! That is what ameriDroid was faced with due to PG&E's Public Safety Power Shutoff that started on Saturday, October 26, 2019, and lasted until the afternoon ▶️

Repairing Your ODROID-N2: How To Recover From An Accidental Short Circuit

December 1, 2019

In this guide we will show you how to repair Q1 on your ODROID-N2. The original article was taken from the Hardkernel wiki page.



Creating a Vision Application In Low Power Situations: Using OpenVino and OpenCV With The ODROID-C2

December 1, 2019

This article will guide you on your journey of setting up an ODROID-C2 with Ubuntu* 16.04 (LTS), building CMake*, OpenCV, and Intel® OpenVINO™ toolkit, setting up your Intel® NCS 2, and running a few samples to make sure everything is ready for you to build and deploy your Intel® OpenVINO™ ▶️

How-To Set Up A Basic NAS: Using Samba To Share Files

December 1, 2019

Using Samba, anyone can turn an old tower or SBC into a file server!

Cryptocurrency Mining: Earning Verium Coins With Your ODROID

December 1, 2019 By Joe Rondx, wiki.vericoin.info ODROID-XU4, Tutorial

Odroid XU4's, developed by Hardkernel Co. Ltd, are quite popular. SBC systems in general are designed to be very power efficient, because of this they are pretty good at Verium mining. The downside is you need a lot of them to amass a sizeable amount of hashes. The up-front cost of SBC's can be very high, but their power usage is very low. Remember to take into account all the required extras to make SBC's function (SD cards, power cables/supplies, network cables, switches, cooling, mounting mechanism, etc...).

For SBCs the mining software usually needs to be compiled for 1way using 128 MB per thread. 2Many devices have been tested in terms of their hashrate and it can fairly be stated that the Odroid platform with its Octa core CPUs (Exynos5422 big.LITTLE) and 2 GB LPDDR3 RAM outperforms any other device. By now Hardkernel has even released a special version of the original XU4 which is called Odroid HC1 and is designed for clustering. It should be noted that despite their relative low total hashrate, the ration hashrate per energy is still good. Also, there is software out in the community that helps on the maintenance of large clusters. For more information visit the official ODROID website http://www.hardkernel.com.

Optimized OS Image for Verium Mining

The ODROID Verium Mining Image (by joe_rondx) includes an optimized OS with preinstalled miner and several other handy features. It is made for the XU4 line, that is Odroid XU4, XU4Q, HC1, HC2, MC1.

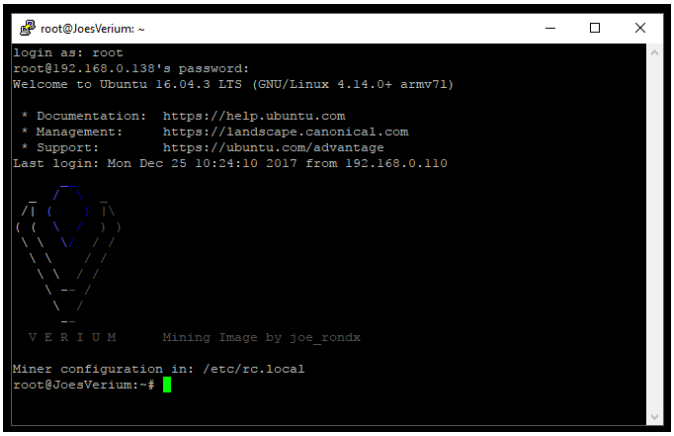


Figure 1 - Logging in to the optimized Verium mining image

Performance features:

- hugepages are enabled (thanks to birty & fireworm)
- Maximal RAM clocking: 933 MHz
- CPU downclocking of big cores: 1.9 GHz (read here why 2 GHz is not worth it)
- Optimal two miners configuration for big.LITTLE cores.

Helper scripts:

- Temperature logging in verium/cpu_temp.log
- Status overview script vrmcheck.sh
- Filesystem expansion by resize.sh

Setup the Image

1. Download the Image from: https://drive.google.com/open?id=1RbXnGUh5fwmfhMQNzefCK75PajyrXjQi
2. Burn it without extracting it (The real issue is the following: the image consists of two partitions, if you extract the gz file you have to make sure you burn both partitions and not only the fat partition).
3. On first bootup give it 5-10 mins time until you should be able to find the device in the network.
4. Log in with standard root/odroid credentials.

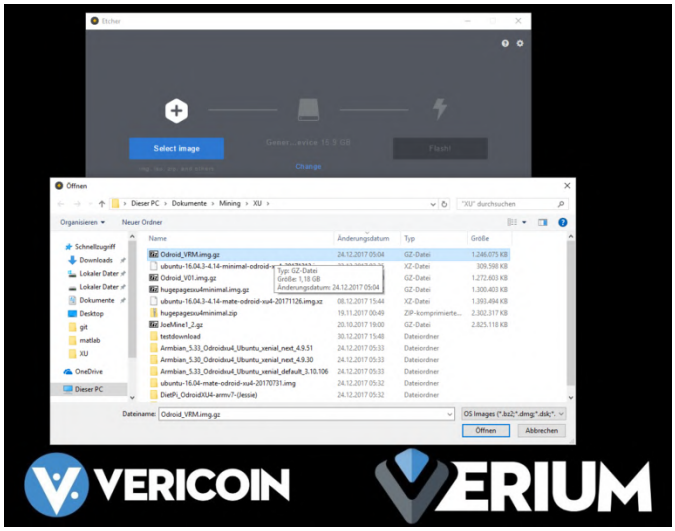


Figure 2 - Flashing the Verium image using Etcher

Configuration of the Image

The first thing you want to do is the Image Update. The script to do so is at: https://raw.githubusercontent.com/DJoeDt/verium/master/odroid_image_update.sh. From the command line use:

```
$ wget
https://raw.githubusercontent.com/DJoeDt/veriu
m/master/odroid_image_update.sh
$ chmod +x odroid_image_update.sh
```

Now before you run the update make sure to kill the miner, I use

```
$ top
$ kill
```

Where, nnn would correspond to the pid of the instance of the miner on your system. Now execute:

```
./odroid_image_update.sh
```

Then edit

```
$ nano /etc/rc.local
```

What it does is:

- Get and compile the latest fireworm miner
- Add the new one miner command to /etc/rc.local
- Update vrmcheck.sh (not nice but well...)

Why it does this: It turned out that the configuration with two miners is bad for a couple of reasons. I must

admit that I was just too greedy and only looking at H/m output. The reasons are the following:

- The low hashrate of the 1-way miner reduces the chance to actually submit a share when pool mining.
- The additional miner connection causes a lot of work on the pool server side.
- It is similarly true for solo mining also.

Oh, and one miner is just easier to maintain, monitor, etc. 🙂 And of course the update of the fireworm miner gives some nice features. I am getting about 530 H/m .

Personal configuration

Let us start with the configuration of the miner. The autostart of the miner is currently done in the rc.local file.

```
$ nano /etc/rc.local
```

At the end you will find two mining commands which you need to adjust for your pool/solo setup. Actually no, after the update you should just delete those commands and use the newly added One miner command.

In addition to the miner configuration you might want to change the hostname and password:

```
$ nano /etc/hostname
$ nano /etc/hosts
$ passwd
```

You can also expand the filesystem by using the script

```
$ ./resize.sh
```

Reboot before going on

```
$ reboot
```

and run the resize script again

```
$ ./resize.sh
```

Network

Usually the Ethernet Port should just connect via DHCP. If you have a Wifi Stick you should use

```
$ nmtui
```

XU4 & XU4Q Tipps

Since the image was created on an HC1 you might want to check the GPU setting: **https://wiki.vericoin.info/index.php?title=Odroid#Downclocking_the_GPU**.

Usage and Monitoring

There is a simple status script in the home directory, call

```
$ ./vrmcheck.sh
```

which prints out the configuration in rc.local, the last 10 lines of each miner log, CPU frequency and the current temperature. Monitoring both miners is a bit of a challenge, but DerrickEs miner Monitor scripts (**https://github.com/derricke/MinerMonitor**) or casanova's CLI monitor (**https://github.com/bezeredi/verium-cli-monitor**) support the configuration of ports (4048 & 4049 in this case). 1I also recommend mining on two pool to decentralize the hasrate, keep some hashes going if one pool is down and also for monitoring.

Donation suggestion

With the new update it's more difficult to spread small H/m portions. I won't take the time to rewrite this, basically: if you run a lot of Odroids with this image consider to have one running for me, thanks a lot!

1. The image starts mining right away - for joe_rondx (next version will include fireworm). This is not meant to be a rip off. Consider to have a freshly burned card mine for us perhaps one hour as initial donation.
2. If you are running less than 10 units (XU4, XU4Q, HC1, MC1/4) the initial donation is all we are asking for.
3. If you are running 10-19 units you may consider to have one small miner running on joe_rondx address.
4. If you are running 20-29 units you may consider to have one more small miner running on fireworms address.
5. If you are running 30-39 units you may consider to switch joes small miner to a big one.
6. If you are running 40-49 units you may consider to switch fireworms small miner to a big one.
7. If you are running 50+ units please also consider donating to the project via its donation page.

Justification: Individual hasrates may vary, but let's assume you were getting 450 H/m per unit without this image. The image should give you 537 H/m which is an increase of nearly 20% . 10 units should produce 5370 H/m while one small miner does about 137 H/m - which is about 2.5% of the total hashrate. Thank you very much for your support! - joe_rondx

Other OS Images

Let us start with special images by Odroid God birty: Odroid Miner Images (ttps://drive.google.com/drive/folders/0B26cQdlGFXo2S3ViQ3IxaVhfUkk). In particular the newest image with enabled hugepages in combination with fireworms miner gives a significant boost of hashrate. It was used for the optimized image. Odroids Official Images (**https://wiki.odroid.com/odroid-xu4/os_images/linux/start**) are of course ve1ry well made. The newest Ubuntu Mate 16.04.3 (20171212) which was release after birtys image even has hugepages enabled. But unfortunately it uses too 1much RAM. Even more unfortunate is that the Ubuntu 16.04.3 (20171213) (MINIMAL, BARE OS) image does not have hugepages enabled. I have once tested like 6 different images that are available for the XU4 platform. For the beginning I recommend the DietPi image (**https://dietpi.com/#download**) 2because its included diet-config tool already supports lots of the configurations you want to set.

Tweaking the OS for Mining

The whole thing as a script (ARM Miner + XU4 Setup): Download the shell script at: **https://raw.githubusercontent.com/DJoeDt/verium/master/1wayARM_XU4_VeriumMiner_install.sh**.

```
$ wget
https://github.com/DJoeDt/verium/raw/master/1w
ayARM_XU4_VeriumMiner_install.sh
$ chmod +x 1wayARM_XU4_VeriumMiner_install.sh
$ ./1wayARM_XU4_VeriumMiner_install.sh
```

Downclocking (yes, down!) the CPU

To prevent throttling due to heat it may actually increase your hashrate if the CPU does not run at 2 GHz (max). Even if you can prevent throttling at 2GHz it is anyhow not worth it: you might get 10-15 H/m more but it costs about 2 Watts (out of 12) to get this last increase - so it will not pay back, for Details checkout the benchmark at: **https://www.planet3dnow.de/vbulletin/threads/428622-Odroid-HC1**. Install the utility (or use DietPi config)

```
$ sudo apt-get install cpufrequtils
```

use it directly

```
$ sudo cpufreq-set -c 7 -u 1.9GHz -r
```

and make the change permanent by creating a config file

```
$ sudo nano /etc/default/cpufrequtils
```

with the following settings

```
ENABLE="true"
GOVERNOR="performance"
MAX_SPEED=1900000
MIN_SPEED=1900000
```

Now the CPU should always run at constant speed. This also saves a reasonable amount of power.

Overclocking the RAM

On the boot-FAT-Partition edit the boot.ini

```
$ sudo nano boot.ini
```

Find

```
ddr_freq
```

and change the value to 933

```
# DRAM Frequency
# Sets the LPDDR3 memory frequency
# Supported values: 933 825 728 633 (MHZ)
setenv ddr_freq 933
```

Make sure before bootz to

```
# set DDR frequency
6dmc ${ddr_freq}
```

Downclocking the GPU

Install this utility

$ sudo apt-get install sysfsutils

Then edit

```
$ sudo nano /etc/sysfs.conf
```

and insert the following line

```
# Put GPU into powersave mode (= Downclocking
it)
devices/platform/11800000.mali\:/devfreq/11800
000.mali\:/governor = powersave
```

then start the service

```
$ sudo service sysfsutils start
```

Effect: reduced the power consumption by 0.7 – 0.8W, SOC will be 1-3°C cooler. Get information on reducing power consumption in a headless scenario at https://obihoernchen.net/1340/lower-gpu-clock-of-odroid-xu4-for-headless-servers/ Another way might be

```
$ sudo nano /etc/rc.local
```

and add this line before exit 0

```
$ echo powersave >
/sys/devices/platform/11800000.mali\:/devfreq/
11800000.mali\:/governor
```

Setting up a Swapfile

Verium is memory intensive, so we increase the swap file (or use DietPi config).

```
$ sudo fallocate -l 1G /var/swapfile
$ sudo chmod 600 /var/swapfile
$ sudo mkswap /var/swapfile
$ sudo swapon /var/swapfile
```

check it with

```
$ free -h
```

and configure that permanently

```
$ sudo echo "/var/swapfile none swap sw 0 0"
>> /etc/fstab
```

Processes

Further optimization can be done by checking the process tree

```
$ pstree -p
```

and disable/uninstall stuff that is not needed. KILL 'EM ALL! I did not find it yet but if you come across ads7846 remove it.

```
$ modprobe -r ads7846
$ tee /etc/modprobe.d/blacklist-ads7846.conf
<<< "ads7846"
```

XU4 hardware

The key is to exchange the thermal tape of the heatsink with some good thermal paste, decreases the temperature by 10 degrees (C) using the same testing conditions. Also get the under-side cooled as well. To save electricity you may turn down the power suppiies voltage with a screwdriver. Check sd card slot heat.

Optimal big.LITTLE and Maximal RAM Usage

aka "Getting the last Hash out of your Odroid". The first step is to use an OS image that uses a minimum amount of RAM for the system. To illustrate how to use the big.LITTLE cores and most of the memory we first have a look at the configuration with effectstocause miner.

The Goal

Verium mining is a lot about RAM, so you want to maximize the memory usage. How does that work? The verium Miner (https://github.com/effectsToCause/veriumMiner) can be configured to use a different amount of RAM per thread. So the idea is to use 2 different miner compilations and make use of the 2GB LPDDR3 RAM @ 933Mhz the Odroid has. Plus: do that wisely to also benefit from the big.LITTLE cores of the Samsung Exynos5422 Cortex™ ARM Cortex-A15 (2.0Ghz) / Cortex-A7 (1.4 Ghz) Octa core CPUs

Technical details

The miner settings are called 1 way or 3 way where

```
1 way => 128 MB per mining thread
3 way => 384 MB per mining thread
```

The #way of the miner is configured in

```
$ nano veriumMiner/algo/scrypt.c
```

Now you can calculate around for yourself or just trust me that you want those two miner configurations and then run

```
5 threads @ 1 way = 640 MB
3 threads @ 3 way = 1152 MB
1792 MB total RAM
```

The system needs some memory, too and with this setting there is only around 60 MB left free - but only if you use the image linked above (I have tested 6 different ones, only this works with -t 5 & -t 3). A 2 way compilation might be interesting as well, but my compilation try did not work.

Easy Installation

I have prepared two scripts on my git repository (**https://github.com/DJoeDt/verium** - no warranty whatsoever):

```
1wayARM_XU4_VeriumM6iner_install.sh
3wayARM_XU4_VeriumMiner_install.sh
```

which install the miners into

```
~/verium/1wayminer/
~/verium/3wayminer/
```

How to run them

Having both compilations at hand we need to manage them properly. Besides the threads there are two things to configure

```
# the big.little core
# the API Port
```

big cores -t 3

The big cores shall run the 3 way miner on the standard port 4048 with high priority

```
--cpu-priority 4
--cpu-affinity 0x00F0
-b 4048
--api-bind 0.0.0.0:4048
```

those are the options to be set. About using cpu-affinity (**https://wiki.vericoin.info/index.php?title=Cpu-affinity**). This is the complete line from my /etc/rc.local

```
# 3way -t 3 big
/root/verium/3wayminer/cpuminer -o
stratum+tcp://pool-eu.bloxstor.com:3003 -u
VEXMki29ycW5vSt3MmdM5iwHqsHux91EMr.Guide -p
GuidePwd --cpu-priority
4 -t 3 --cpu-affinity 0x00FF --api-bind
0.0.0.0:4048 &
```

Just copy it and give it a try as donation 😉 .

little cores -t 5

The little core shall run the 1 way miner on API Port 4049 with lower priority

```
--cpu-priority 1
-b 4049
--api-bind 0.0.0.0:4049
```

where I just don't touch --cpu-affinity and thus the remaining 4 little + 1 big cores are used. This is the complete line from my /etc/rc.local

```
# 1way -t 5 little
/root/verium/1wayminer/cpuminer -o
stratum+tcp://pool-eu.bloxstor.com:3003 -u
VEXMki29ycW5vSt3MmdM5iwHqsHux91EMr.Guide -p
GuidePwd --cpu-priority 1 -t 5 -b 4049 --api-
bind 0.0.0.0:4049 &
```

Monitoring

If you use the API you need to configure both ports. My workaround with birtys <3 webscripts looks like this: So far I have copied index_monitor.php to index_monitor4049.php, reconfigured

```
defined('API_PORT') || define('API_PORT',
4049);
```

in it, and just included it by adding to the index.php. More information on monitoring can be found here.

The Result

Some remarks before we look at H/m:

1. again: only the image linked above worked for me, but not even by default
2. you still have to create a swap file (included in my scripts)
3. I lied about the lines in my rc.local, I actually mine on two different pools. Decentralize it!

Hashrate Numbers!!11

Originally I ran birtys miner configuration which actually is the 1 way configuration. Without any -t option it just starts 8 threads and with pool mining i got an average of

```
395 H/m = 1 way -t 8
```

The two miners put out like

```
195 H/m = 1 way -t 5
250 H/m = 3 way -t 3
```

Happy adding! 🙂

I wonder how this performs when solo mining?

Update to fireworm miner & hugepages

```
1way : 128MB -> nr_hugepages = 65. 3way :
384MB -> nr_hugepages = 193. 6way : 768MB ->
nr_hugepages = 386.
```

Configuration of birtys image

Setup: birtys hugepages minimal image: [drive] Configure hugepages

```
$ sudo nano /etc/sysctl.conf
```

to

```
$ vm.nr_hugepages=839
```

Change host

```
$ nano /etc/hostname
$ nano /etc/hosts
```

and dram_freq=933 in

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

as well as the password by

```
$ passwd
```

Reboot before going on

```
$ reboot
```

Fireworm Miner Installation

Remove old miner from birtys image:

```
$ rm -rf veriumMiner
```

Install newest miner by script:

```
$ wget \
```

https://github.com/DJoeDt/verium/raw/master/nwayARM_HC1_fireworm_install.sh

```
$ chmod +x nwayARM_HC1_fireworm_install.sh
$ ./nwayARM_HC1_fireworm_install.sh
```

Miner Autostart Configuration

Autostart config

```
$ nano /etc/rc.local

# Verium Miner Configuration
# big cores
$ nice --10 /root/verium/nwayminer/cpuminer
-o stratum+tcp://vrm.n3rd3d.com:3332 -u
joe_rondx.1 -p joe
-t 3 -1 1 --cpu-affinity 0x00F0 --cpu-priority
2
--api-bind 0.0.0.0:4048 --no-color >>
/root/verium/nwayminer/3waymine.log &

$ sleep 5 # delay for hugepages allocation

# little cores
$ /root/verium/nwayminer/cpuminer
-o stratum+tcp://eu.vrm.mining-pool.ovh:3032 -
u joe_rondx.HC1_1 -p joe
-1 4 --cpu-affinity-stride 1 --cpu-affinity-
oneway-index 0 --cpu-priority 0
--api-bind 0.0.0.0:4049 -b 4049 --no-color >>
/root/verium/nwayminer/1waymine.log
```

Run the 3way miner first !!

```
-t 3 -1
```

First because it should make maximal use of hugepages. Secondly run only 1way

```
-1 4
```

where 1 thread runs without hugepages.

Hashrate

```
=> 400 + 137 = 537 H/m @ 1.9 GHz
```

Power to the Rig

The specs say 4 Amps @ 5V for an Odroid. But there is way more to consider if you want to power your rig.

Reference

https://wiki.vericoin.info/index.php?title=Odroid
https://ameridroid.com/products?keywords=aluminum
https://forum.odroid.com/viewtopic.php?f=93&t=27239
http://www.thinkwiki.org/wiki/How_to_use_cpufrequtils
https://wiki.odroid.com/odroid-xu4/os_images/linux/ubuntu_4.14/ubuntu_4.14
https://forum.odroid.com/viewtopic.php?f=146&t=28895&sid=873dc51d2cf97257c807b99826f91525

Protect Your Privacy: Why You Should Probably Make Your Own Home Automation Devices

December 1, 2019 By Miguel Alatorre, ameriDroid Technician Development, ODROID-C2, ODROID-N2, ODROID-XU4

With all of the hacks, vulnerabilities, data leaks and other disheartening news surrounding cloud services, cloud products, and big data companies, you may want to consider putting in the effort to create your own version of these solutions.

"Smart" doorbell Have you read the recent news that the "smart doorbell" sold by a major company sends your home's WiFi password in plain text through the Internet, allowing hackers to potentially gain access to your home network? And how the same company bragging about tracking "Trick-or-treaters" as they went from house-to-house on Halloween?

Have you read the news from this summer of a popular "smart lock" that allowed hackers to unlock your doors without knowing your passcode or having a key?

Figure 1 - Image: Chase Dardaman, Jason Wheeler

From https://techcrunch.com/2019/07/02/smart-home-hub-flaws-unlock-doors:

“Dardaman said any hub connected directly to the internet would be remotely exploitable. The researchers found five such vulnerable devices using Shodan, a search engine for publicly available devices and databases.” Have you heard that Alexa, Siri and Google Assistant can be hacked by a laser from up to 350 feet away? News of this exploitation can be found

here: https://www.vice.com/en_in/article/3kxwvy/alexa-siri-and-google-assistant-can-be-hacked-remotely-with-lasers

These are only three examples of big data companies mishandling the privacy and security of their customers. One of the reasons these "big data products" are so often hacked is because big data companies are often less concerned about your security than they are about profits and getting a product quickly to market. In addition, they generally have sold a lot of the products in question, which makes them a juicy target for hackers.

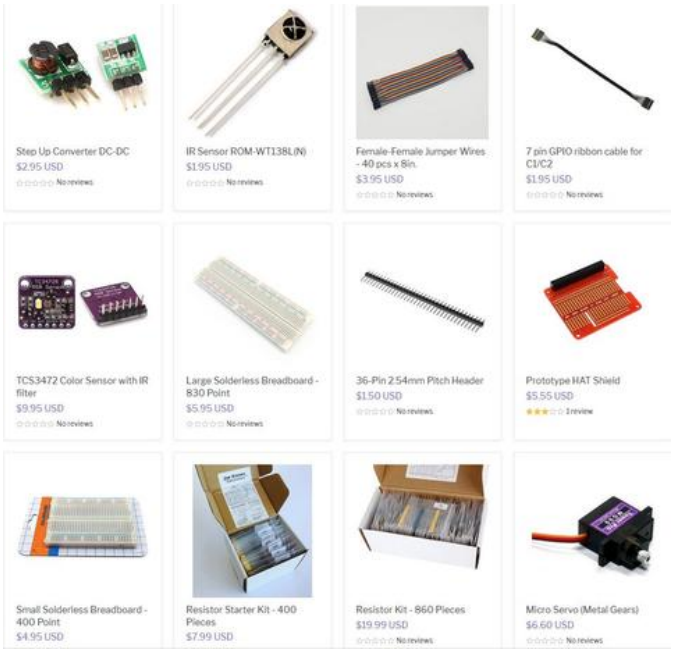


Figure 2 - Resource List

Here are just a few benefits to building your own versions of these products:

- You are free to exercise as much creativity and passion when developing your products as you'd like
- You'll learn new skills and sharpen existing skills
- You can add features that haven't been offered by others and suit your particular needs
- You can take pride in creating your own solutions, and you can fix your own stuff
- You can go "overboard" with your security steps, if you so desire, or you can rely on "security through obscurity," (**https://en.wikipedia.org/wiki/Security_through_obscurity**) or a combination of the two, something which may work much better for a one-off product than a product sold to thousands or millions of customers by a big data company
- You can take an existing how-to project found online and modify it to your personal needs and preferences, often with less effort than starting from scratch There are many open source projects that can be used as a starting point for your solution. As opposed to a "canned solution" provided by a big data company, open source software can be audited by anyone to see if any security or privacy concerns exist.

The original article can be viewed at the following link: **https://ameridroid.com/blogs/ameriblogs/privacy-why-you-should-probably-make-your-own-home-automation-devices**

Mobile Workstation: Using An ODROID-N2 To Create A Full-Featured Computing Experience

December 1, 2019 By Pieterjan Montens ODROID-N2, Tutorial

For some years I have been longing for two things:

- A “Laptop” with a high-quality keyboard and a trackball
- An ARM-architecture based “workstation” to be productive with

Unexpectedly, and because I somehow managed to lose my BIOS password and lock myself out, I had to part with my old and beloved laptop, the one running Debian (I travel with both a Linux and Windows laptop. No I can’t think of a single reason to bring a Mac along). But instead of just jumping to the latest Thinkpad, I wanted to see if I could somehow mix those cravings mentioned above: a deconstructed “mobile” ARM-based workstation where I could choose all the parts, without solder or duct-tape, just off-the-shelf components. It would be power-efficient. It would adapt to the places I’ll be working at, using screens if available, power, etc., so I’d just need to take the components I’d need for my journey. It would be low-budget. So, off I went with a small shopping list.

I already knew I’d use my ergonomic keyboard and trackball, because those are my most proficient input tools. I just needed a computing unit, a screen, and a battery. Those last two components were easy to come by, Amazon has plenty of them:

- A standard 13.3" 1080p panel that somehow works, but can’t change brightness and contrast is wrong: should’ve picked something better.
- A 25000mAh power pack with USB and 12V outputs, that’s important. Mine is from Krisdonia, and unlike the screen, I’m very happy with it.

Choosing an ARM-based computing unit, powerful enough to “work” with (decent computing and graphics capabilities) and with a robust storage medium (i.e.; not microSD) was a bit more difficult: I already had plenty of experience on Raspberry Pi’s

(this was before the 4 came out, more on that below) for home projects and monitoring solutions, but knew it would be lacking as a day-to-day work platform. After a small comparison of available alternatives, I went with the ODROID-N2 4GB, mostly because of the robust eMMC storage. As for a more complete list of what it offers:

- BIG.little architecture: "BIG" quad Cortex-A73 and "little" dual Cortex-A53, making it a heterogeneous hexacore, associated with 4GB DDR4 RAM
- a decent Mali-G52 GPU
- eMMC memory (up to 128GB)
- Gigabit ethernet, HDMI (4k@60hz), 4 USB3 ports and
- Comes in a nice package with a massive passive cooler, which acts as support for the whole unit

Installation went like a breeze (Debian Buster with 4.9.190-odroidn2-arm64 kernel), and before long my usual working environment was installed and fully functional. It worked! I used it to actually make work happen. I'm writing this article on it. It's hardware you'd find in a smartphone, it's small, and now I've plugged my peripherals in and everything is great. It also happens to minimize my environmental impact, the power consumption being very limited compared to a standard laptop, let alone desktop (the screen is the most power hungry element here, just like in Smartphones).

So, what does it feel like to work on ARM?

Well, for starters: Thanks to Debian compiling the whole distribution for multiple architectures, I have access to almost all of my most wished for tools. Some (proprietary!) software, like Synology's cloud station, are not available for ARM architectures. Slack isn't available either, but they offer a workable web-based interface.

I didn't experience any problems with day-to-day tools such as vim, node, npm, though Docker is a tad slow to build. However, even GIMP and FreeCAD work! Working on the battery is nice too, I didn't really stress it but until now I never managed to empty it.

Occasionally a tab crashes in Firefox. Otherwise browsing is a mundane affair, though nothing comparable to a mean-16GB-i7-machine. Client-heavy web applications are another matter and sometimes a bit slow. Switching to Chrome for Google docs helps, however. It didn't become my sole and unique "computer" I take with me though… I still carry a Thinkpad along most of the time, and they turn out to be quite complementary. I just need some more time to set it all up when I arrive and to strip it all down when I leave.

This, however, won't be the end of my ARM adventures. There are still some weaknesses that I would like to be addressed, for a device like this to fulfill all my needs. Like, for example:

- Tens of cores. BIG.little is great, just add more !
- More RAM !
- Even better storage (NVMe ?)
- On-board quality networking (wifi, ethernet, Bluetooth)
- 2 video outputs
- hardware-based full-disk encryption: these things are easy to lose, and may contain sensitive data

Let's not forget that Microsoft is expected to release an ARM-powered Surface hybrid soon: I may end up working exclusively on ARM sooner than I thought was possible.

But how does it compare to the Raspberry Pi 4 ?

Shortly after I bought the ODROID, the Raspberry Pi 4 came out, completely unannounced, much to the surprise of virtually everybody. Most readers will probably want to know how both ARM platforms compare to each other. This short and subjective comparison won't include the advantage of raspberry's formidable community, which is mostly why I went with raspberry when beginning my ARM adventure in the first place. I will only focus on the specific use-case I have, but one should not forget that both platforms don't really address the same market segment. All-in-all, the technical differences with the A72-powered raspberry 4 are small:

- While the ODROID A73 CPU is a completely different architecture, it only offers some improvements to the A72, as detailed on anandtech: https://www.anandtech.com/show/10347/arm-cortex-a73-artemis-unveiled

- HardKernel's ODROID has 2 extra "little" A53 cores, as used by the Raspberry Pi 3 (who has 4 of them). So you get like the computing equivalent of a Raspberry Pi 4 and half a Rasbperry Pi 3.
- Both have Gigabit ethernet and USB 3 ports
- ODROID comes with it's own heatsink, while the raspberry throttles quickly without one
- the ODROID Mali G52 GPU should be at least twice as powerful as the Raspberry Pi's videocore 6 (850Mhz, 6.8 Gpix/s vs. 500Mhz, 2.5 GPix/s)
- The ODROID onboard chipset does not include wifi or bluetooth, it needs a little dongle... that's a shame, really, because the RaspBerry's include both

While I would have loved the dual micro-HDMI outputs of the Raspberry Pi 4, the micro-SD storage is just too unreliable to be considered for day-to-day work. eMMC is, in my opinion, the decisive argument in favor of the ODROID, compared to the Raspberry Pi 4.

The original article can be found at **https://medium.com/@pieterjan_m/reinterpretation-of-the-mobile-workstation-e8dc95d279f9**.

KVM: Fun with virtualization on the ODROID-H2

December 1, 2019 By Tobias Schaaf ODROID-H2, Tutorial

When it comes to the ODROID-H2, my use case for it is to work as a virtualization host, to run a few test VMs to run software on, or test upgrades of systems for my company (e.g. upgrading from older OS versions to newer versions). Looking at the forum, I see people struggling to get VMware or Citrix Hypervisor running, mainly due to unsupported NICs an other components of the board. For me, that was always a little puzzling, as I know about KVM, which is part of the Linux Kernel and allows for easy virtualization, and since under regular Linux (Ubuntu, Debian, etc.) NICs and stuff work fine, as does virtualization without any issue. When I first mentioned this in the forum, I was asked to make a guide, since rarely anyone seems to know about this. So let's see what there is to say: first, let's explain what I'm actually talking about, and what tools and components I use for this.

KVM

KVM is built into the Kernel, and actually means "Kernel-based Virtual Machine" and uses the Linux Kernel to run Virtual Machines (VMs) on it. This allows you to run VMs directly under any Linux you run on your ODROID-H2 (or any other PC that runs Linux).

QEMU

Qemu – Quick Emulator is known as a rather quick and feature rich emulator that allows you to emulate different systems and hardware components. It can be used in combination with KVM to emulate hardware such as hard drives, CD-ROM drives, NICs, etc. but sends the commands from the devices to the KVM virtualization layer. It allows for advanced features such as snapshots for VMs.

libvirt

Libvirt is a open-source API to configure VM platforms such as KVM, Xen, VMWare, or QEMU. The list of supported Hypervisors is rather long and even includes containers such as LXC. We will use it as a

frontend, both graphically as well as on the command line to do things with our VMs in KVM.

First Scenario

Now that we have decided to use KVM, let's see what we need to run VMs and how to install both Linux and Windows VMs, as well as how to control them. Since we're ODROID lovers, we do this of course on our ODROID-H2, but as said before, it will also work on any other x86 based PC/Server that runs Linux. Keep that in mind when I refer to the ODROID-H2 from now on, as it applies to any other system as well.

Requirements

- ODROID-H2
- Installed OS (Debian or Ubuntu for easy start)
- Internet Connection

In this first scenario, we don't need much: only an ODROID-H2 with an installed Linux of your choice. As I favor Debian for server tasks, I will use Debian Buster as my reference OS, but the commands apply to Ubuntu as well and should work the same.

Installation and Configuration

Let's just assume you have already installed a Linux distribution fully installed with a Desktop Environment (DE) and Network Manager and your default user is called "odroid".

```
$ sudo apt install virt-manager
$ sudo adduser odroid libvirt
$ sudo reboot
```

Congratulations, you're done and can start creating VMs! That wasn't so hard, was it?

Virtual Machine Manager (virt-manager)

The Virtual Machine Manager (virt-manager) is a graphical interface that you can use to create and configure your VMs, create snapshots and all kind of other things.

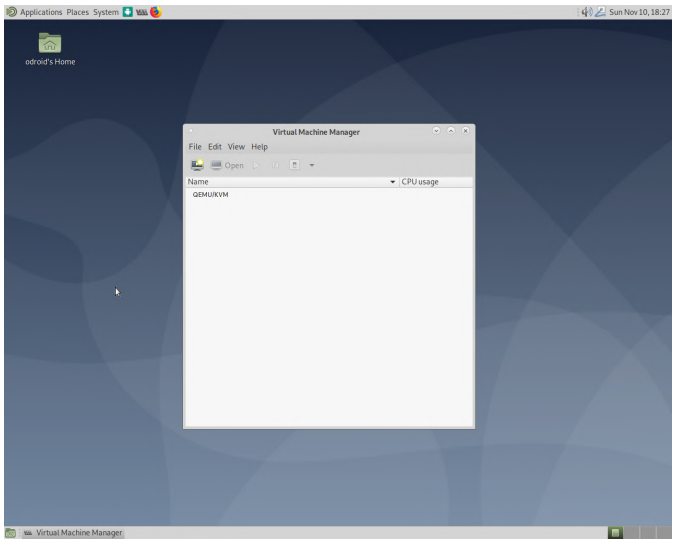


Figure 1 - Virtual Machine Manager on Debian Buster MATE Desktop

The Virtual Machine Manager is what you use to create, configure and interact with your VMs. It can be used to observe resources such as CPU usage, and other things. I don't want to go too much into the details, let's just say there's a lot you can do with it, and you might want to read up on some tutorials for the advanced stuff. For now, let's keep it simple and see what we need to create a VM. Creating our first VM I still have a Debian Stretch netinstall image laying around, which I'm going to use in this scenario. You can mount .iso files directly in VMs similar to VMware and other hypervisors. There is only one detail which is good to know: by default, virt-manager searches for all images under /var/lib/libvirt/images. It uses this as the default pool for all images, both your hard drives, as well as your ISO files that you want to mount. You can add more storage if you want, but for now, let's keep this one and copy the Debian Stretch image that I have into this folder so that it's found directly. After this, let's just click on the button in the upper left corner of our Virtual Machine Manager to create a new VM. It will open a wizard that will guide you through the entire process.

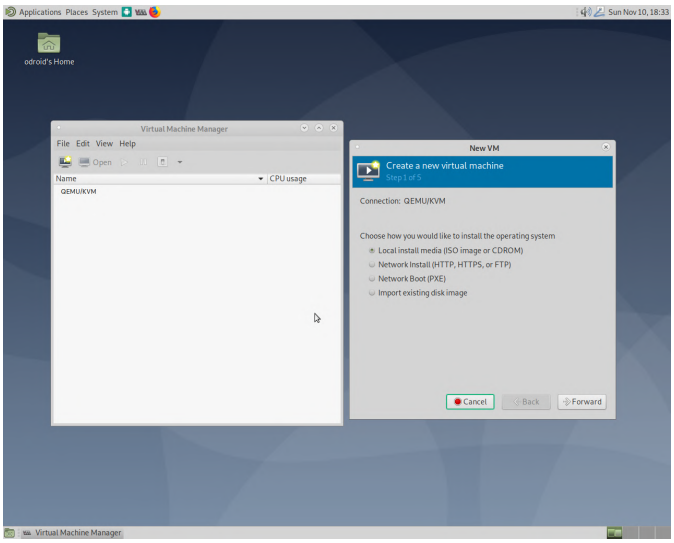


Figure 2 - Setting up a new VM with virt-manager is very easy

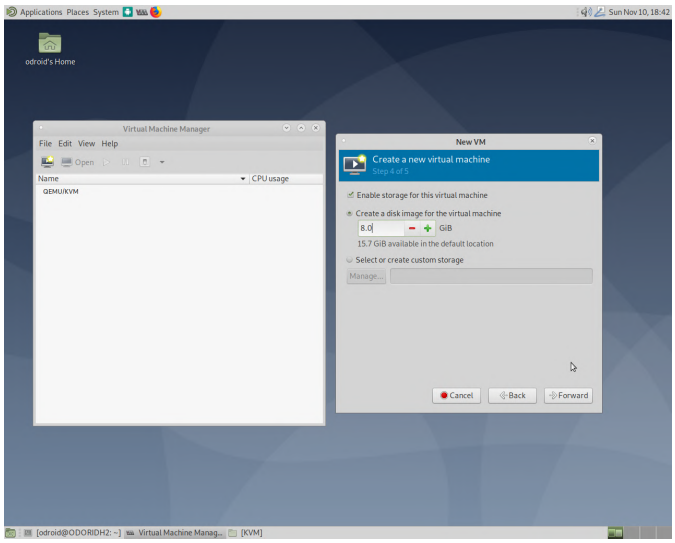


Figure 5 - I only want a small test VM so 8GB is enough disk space

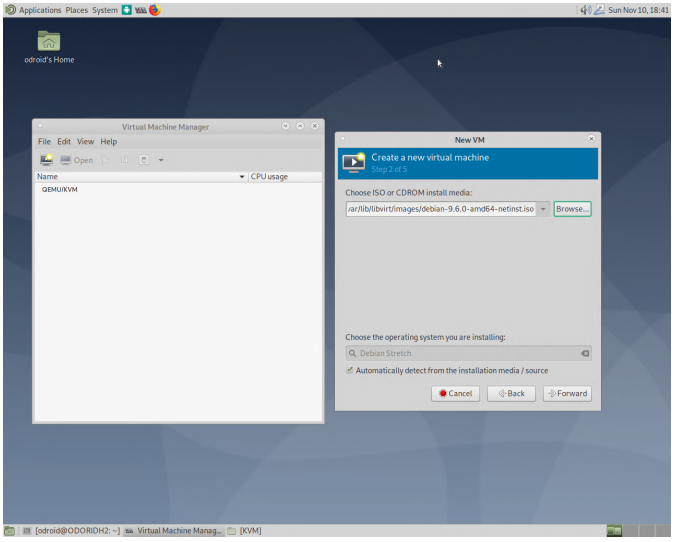


Figure 3 - Selecting Debian 9 netinstall image as CDROM

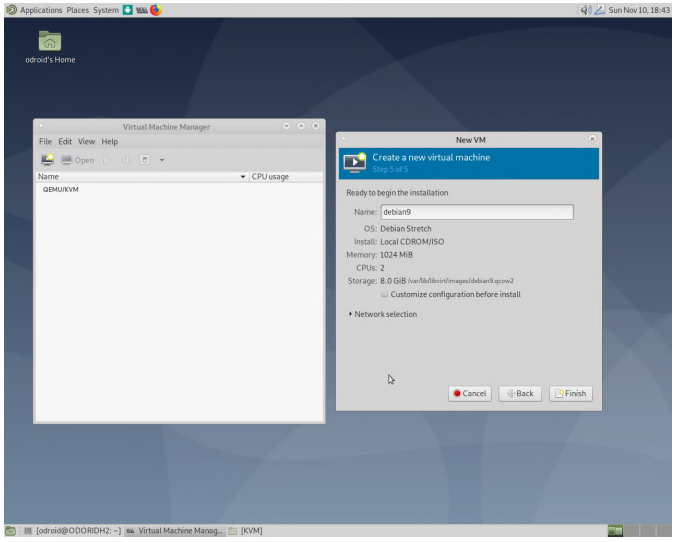


Figure 6 - Give the VM a name and we're done

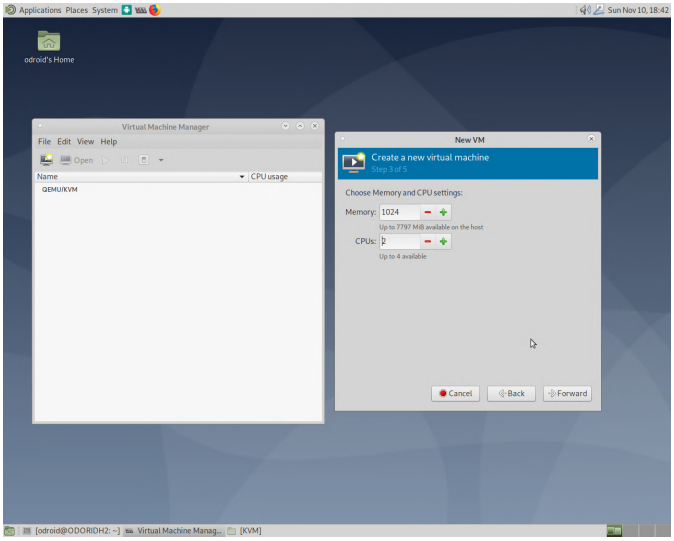


Figure 4 - I chose 2 CPUs to show that VMs can handle multiple cores

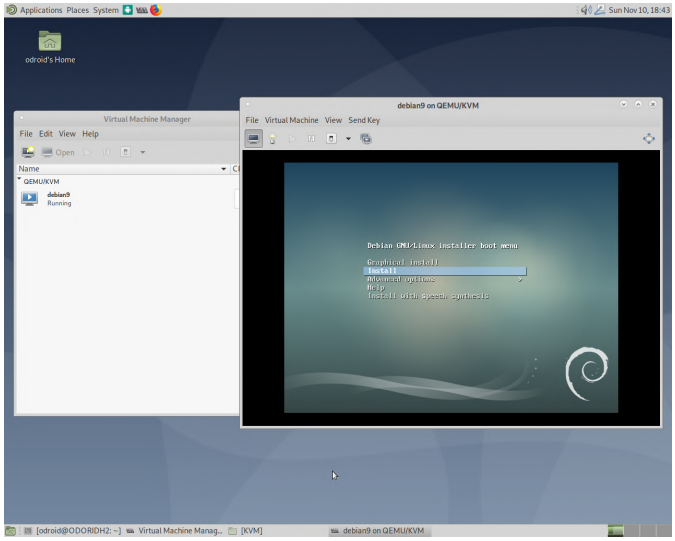


Figure 7 - The VM starts and boots the installer image for Debian 9

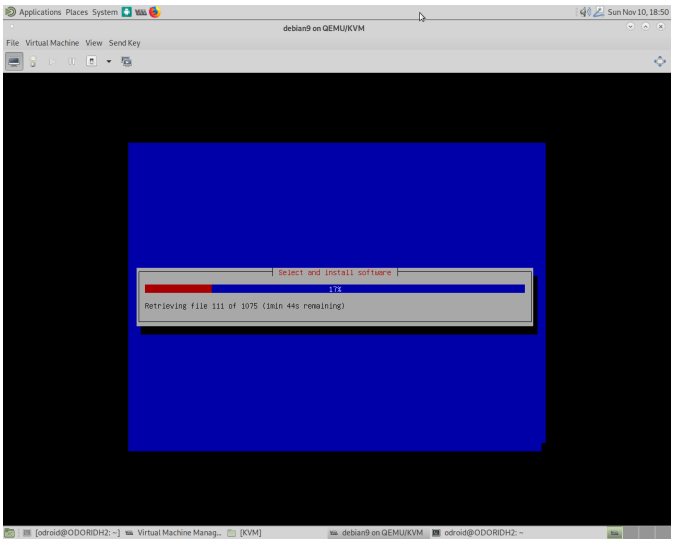


Figure 8 - Going though the standard setup and install MATE Desktop

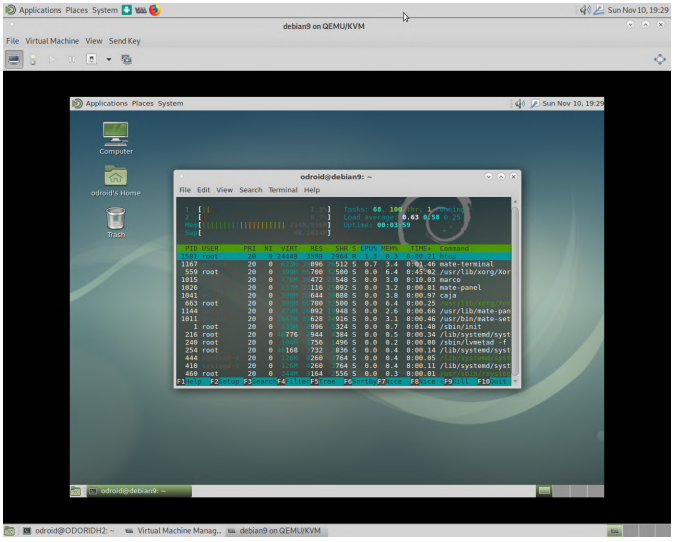


Figure 9 - Debian Stretch MATE Desktop running in a VM

The entire process is rather easy and straightforward. And as we can see, even an installation of a graphical desktop environment is quite easy, and can be used rather well. Instead, you could also use a pure terminal installation and connect to your VM via SSH.

Installing a Windows VM

Installing Windows is similar, but a bit different. While Linux is rather good at handling virtual systems and being a virtual system itself, Windows is not as good with those things, and while we could install a Windows machine pretty much the same way as we did with the Linux system, the performance would be not very good, as Windows does not handle virtualization well enough. For this, it's best to install Windows directly with driver support for virtualization, especially for KVM to get the best performance out of it. First, we need to download some drivers. Luckily for us, the Fedora project takes care of most of the stuff for us, and we can simply download a pre-made image for all the drivers we need. At https://bit.ly/2s3rAUK, there are several links for the stable images and the latest images with the drivers needed, either as a CD or even as a floppy disk image to download. I downloaded the stable CD image which at the point of this article was: virtio-win-0.1.171.iso and placed it in our images folder as explained earlier. Next, we can start the creation of a new VM same as we did with Linux, except for the last screen where we will check the option "Customize configuration before install". Here, we need to change some hardware settings for the VM. We start by adding new hardware, where "Storage" should already be selected. We change the device type to CDROM device and select our virtio image which has all of the drivers.

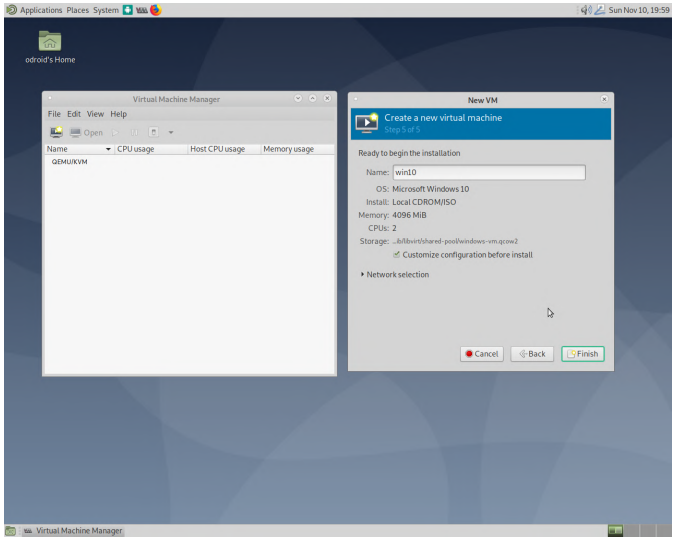


Figure 10 - Make sure to select to customize configuration

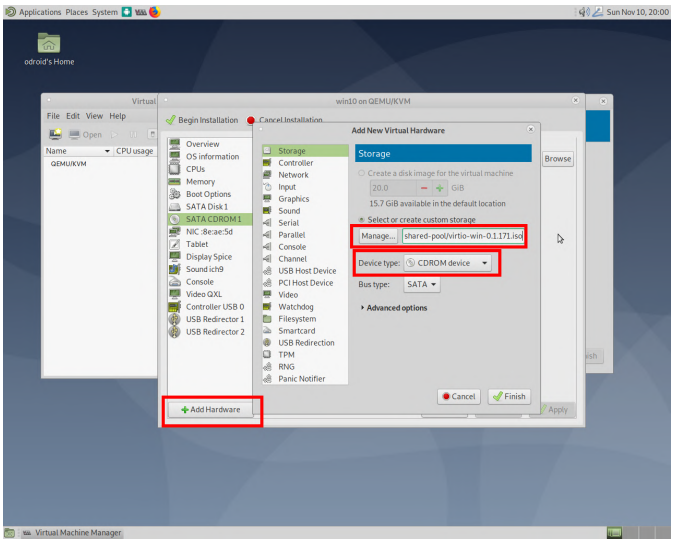


Figure 11 - Add a new CDROM device with virtio ISO

Next, we need to adjust some existing hardware; namely our network adapter (NIC) and the disk we want to use. Both devices tend to be slow on this type of Windows virtualization, especially the disk performance, which has been an ongoing problem for me for Windows guests under KVM. I encourage anyone that wants to use Windows VMs (server or client) to read up on this topic, and I'm open to other options and suggestions as to how to increase performance.

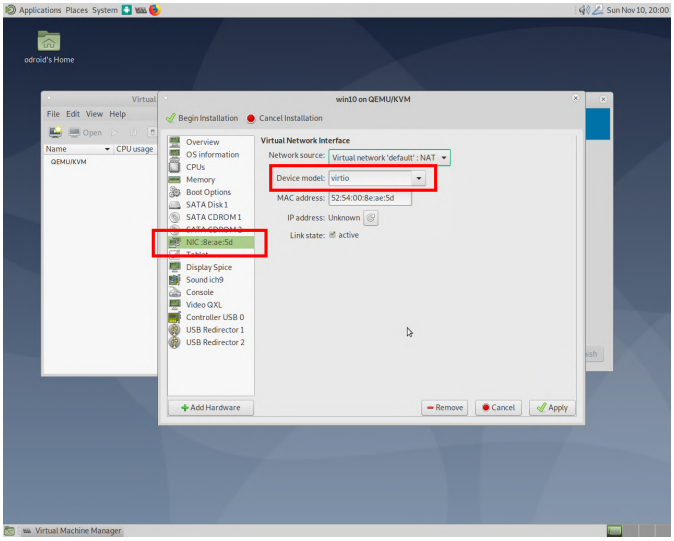


Figure 12 - NIC should be Device model: virtio for best performance

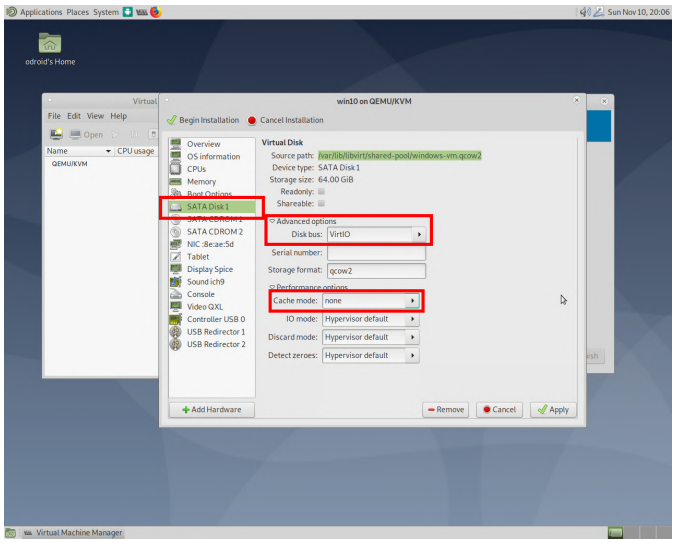


Figure 13 - Without VirtIO Disk bus the disk speed might be as slow as 1MB/sec

The Network adapter should be changed to virtio Device model for best performance. The same goes for the Disk bus of our virtual harddrive. Under performance options, you should set Cache mode "none". This should increase write performance, which is the biggest issue for Windows hosts. The storage format qcow2 is the default for VMs, but some suggest using "raw" as a Storage format instead. While this can increase performance, it also removes some features like snapshots for VMs, so make sure you don't need this. Finally, click the button "Begin Installation" in the upper left corner of the configuration window, and KVM will start our new VM and boots the Windows CD.

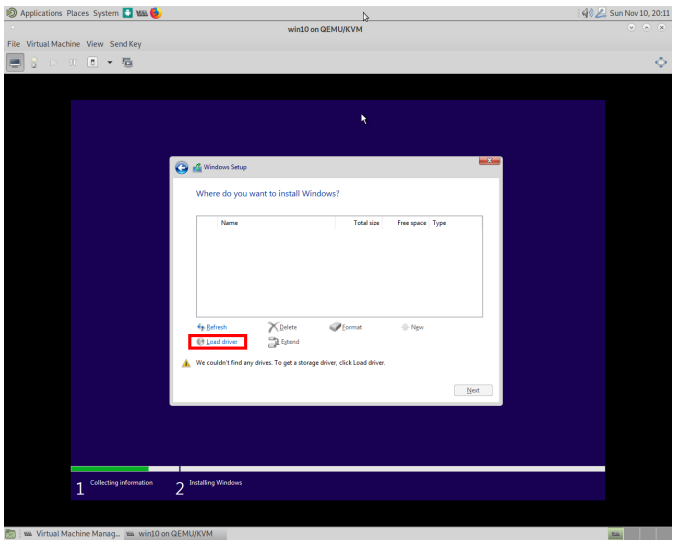


Figure 14 - Windows doesn't know how to handle VirtIO disks

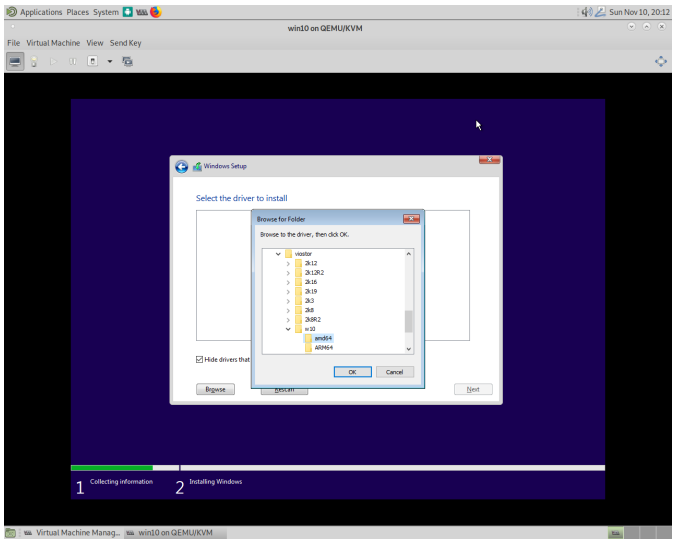


Figure 15 - Installing additional drivers during Windows Setup

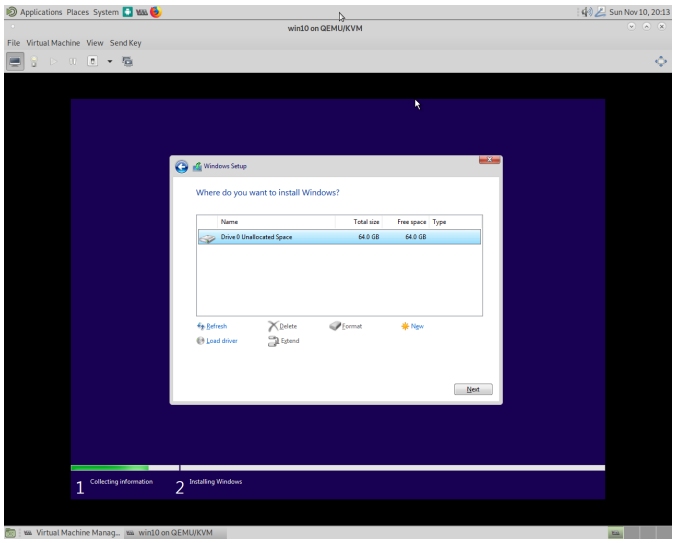


Figure 16 - After installing drivers for VirtIO disk bus Windows found the HDD

When you first start the setup of Windows and want to select the hard drive on which you want to install Windows, you will find the list is completely empty. This is due to the fact that Windows doesn't have any drivers for VirtIO based disk drives. One Windows feature heavily used under Windows 95 or 98 but nearly forgotten by now is required to get things working: the installation of additional drivers during the Windows Setup. For this, select the button "Load drivers" in the lower left corner of the installation window and navigate to the CD we mounted in the second CD drive. Navigate to the folder "viostor" and then to the OS and architecture you want to install. For me this was w10/amd64 as I installed Windows 10 on a 64bit board (the ODROID-H2). If your OS is not listed, take one that is closest to it and it should work anyway. After the driver was installed, the setup found the harddrive and I could continue to install Windows as usual. After the setup completed and the system rebooted, Windows loaded normally but stopped again when it tried to connect to the Internet. I did not install drivers for the network when I installed the drivers for the harddrive. To be honest, I don't know if I could have, and I haven't tried. I probably could have installed all the necessary drivers right then and there, but it's not important as you can continue without them, and install the drivers later.

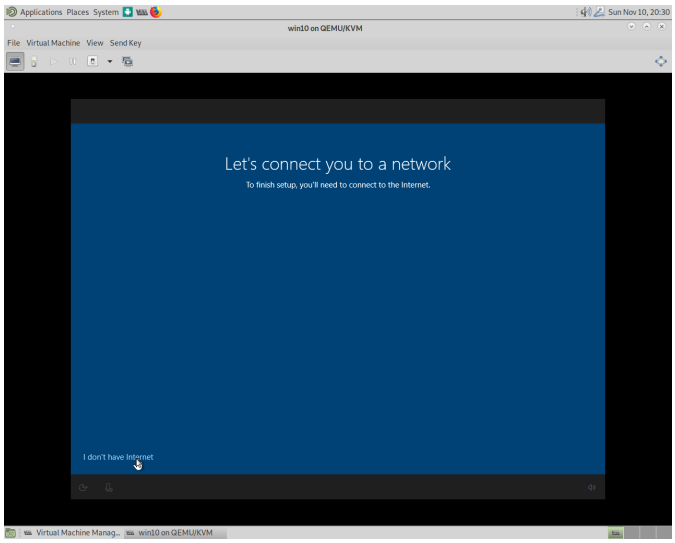


Figure 17 - The network was an afterthought for me

Once Windows is up and running, just navigate to your System → Device Manager in Windows, and you will find the devices where the drives are missing. Right click them and select update drivers, then navigate to the CD with the virtio drivers. You do not need to select the correct folder for the drivers, since Windows will find them on its own.

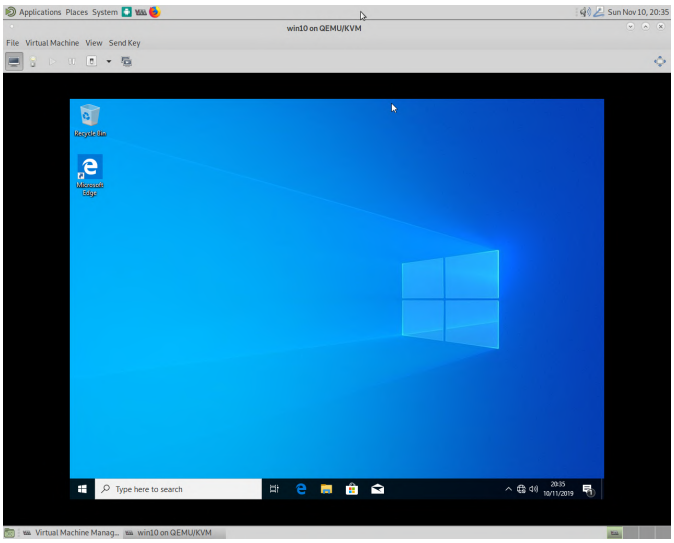


Figure 18 - Windows booted up just fine even without network

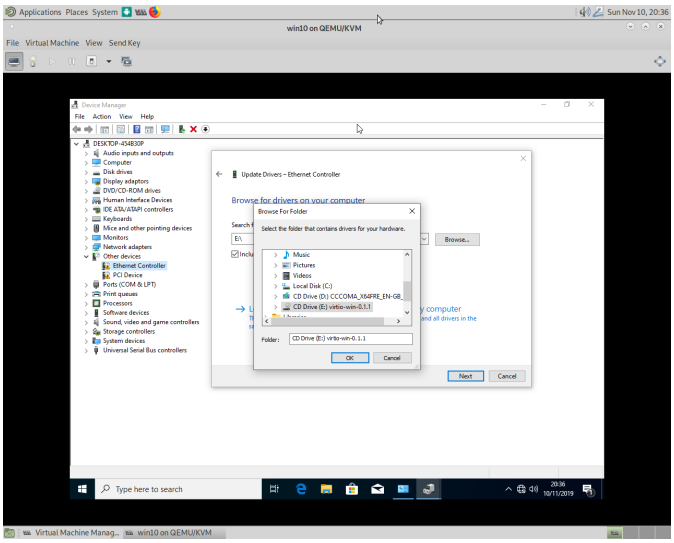


Figure 19 - Installing missing drivers; just select the CD Windows does the rest

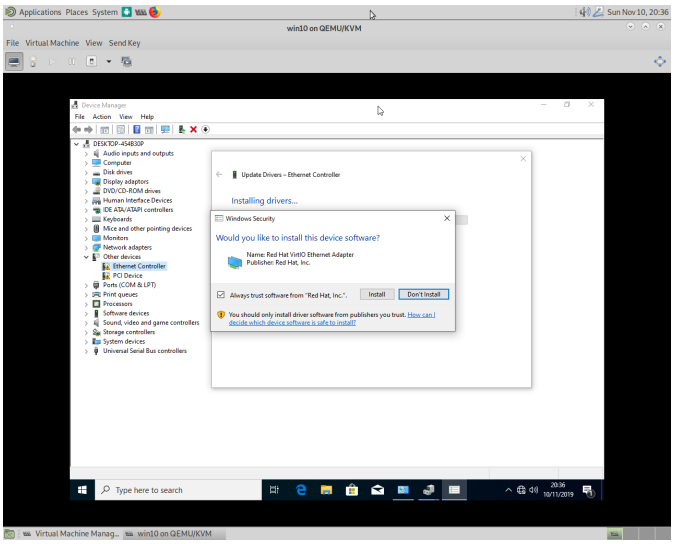


Figure 20 - Windows will ask if it's suppose to install the driver

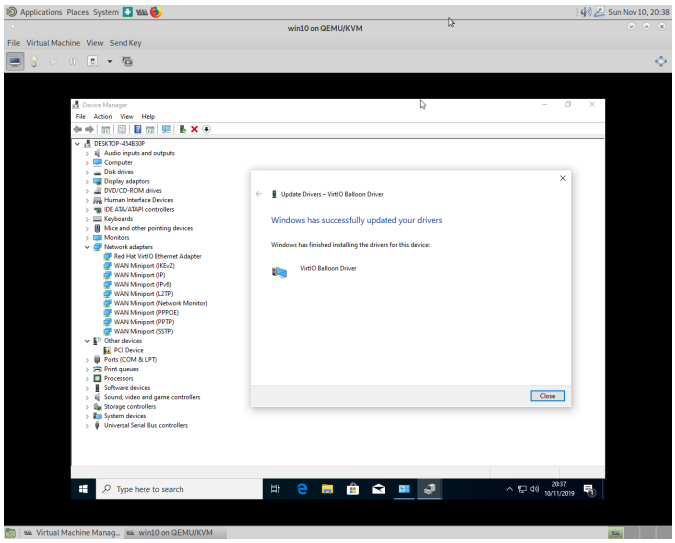


Figure 21 - Installing PCI Device Balloon Driver

Installing the "PCI Device" **Balloon Driver** is somewhat controversial. Some people claim the system runs better without, but I will let you decide about that. What it does is fill the RAM with a pseudo process, and depending on what RAM the Host OS or the Guest OS needs to run applications, it expands or reduces its size, and with that allows either the Host or the Guest to use more RAM, depending on their needs. At this point Windows is up and running, and should behave like any other Windows VM.

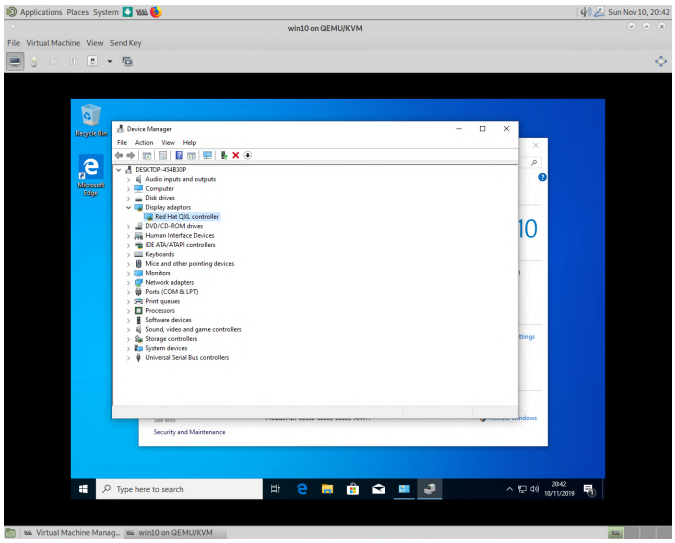


Figure 22 - Now we can watch Windows doing what Windows does best

You can install additional drivers, though. Spice is a graphical interface to connect to the display of the VM. Rather than using VNC, like other hypervisors, Spice is used by default in QEMU. It is faster and more responsive than VNC. For Windows there are spice-guest-tools that can improve graphics and other features on a Windows guest. Think of it like as

VMWare or VirtualBox guest tools/additions just for QEMU/Spice. You can download and install them from https://www.spice-space.org/download/binaries/spice-guest-tools/. You can check if it's installed correctly by checking your graphics adapter after the installation of the spice-guest-tools.

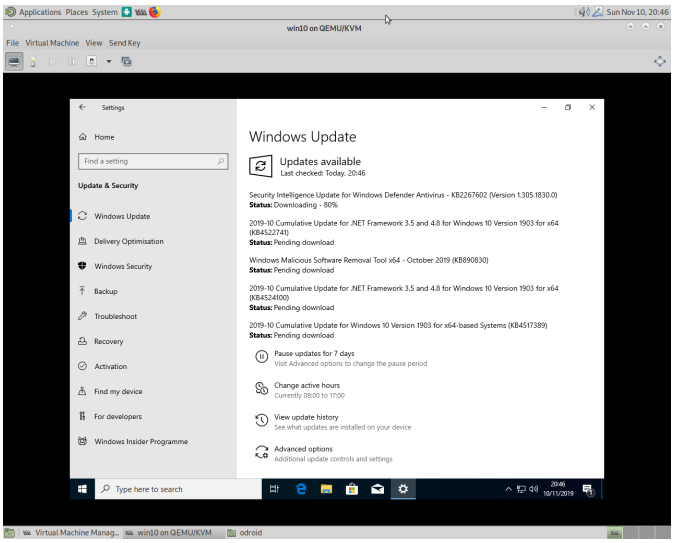


Figure 23 - Red Hat QXL controller is the new GPU driver for your VM with spice-guest-tools installed

There is one feature I want to talk about, as it's been used in other hypervisors as well: Snapshots. The Virtual Machine Manager can handle Snapshots of VMs just fine, same as VMWare, or VirtualBox or any other hypervisor would do. Simply open the Snapshot TAB in your VM click on the plus symbol and create a snapshot.

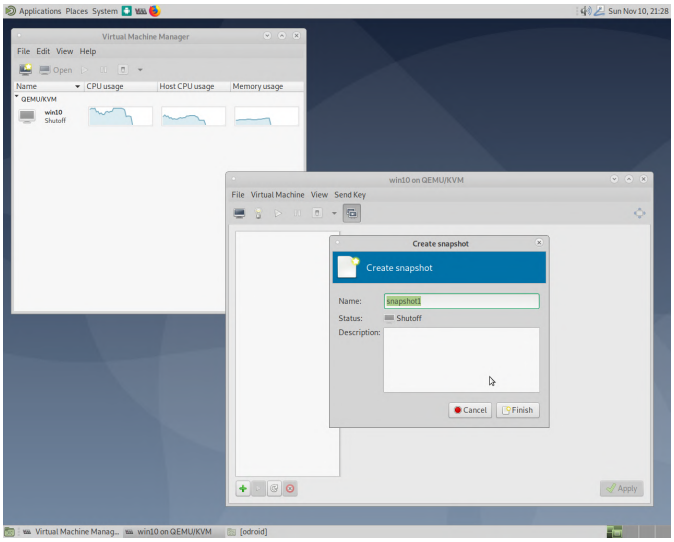


Figure 24 - Creating a new Snapshot for our Windows VM

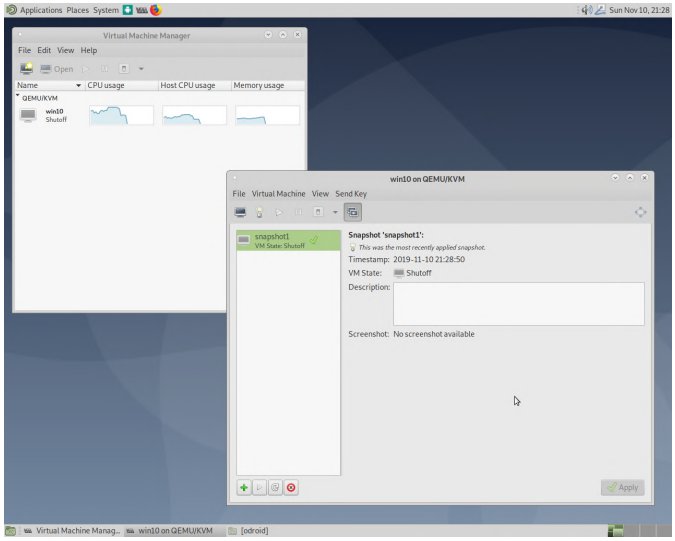


Figure 25 - Snapshot details - you can have multiple snapshots for one machine

While snapshots are a nice way to revert to a previous state, which makes it essential when you want to try out new things that might affect the system (e.g. OS upgrade), please consider that snapshots can reduce overall performance as well as highly increase disk usage. Let's assume you have a VM with two disks: one for the OS and a second with a size of 20 GB as a data store. You fill the data disk to its maximum (20GB) and make a snapshot. Then you exchange the entire data on the data disk. If you check on the host system, where your qcow2 file for the data disk is located, you will notice that the size of the file has increased to approximately 40GB. Although the size limit of the disk is 20GB, it now needs additional 20GB to store the difference between the snapshot and the new data. Keep this in mind when working with snapshots, and also that snapshots should only be a temporary solution, not a way to organize or backup your data.

Up Next

In the next article on this topic, I want to talk about advanced features, like shared storage pools and live migration of VMs from one host to another, which means moving a VM while running from one PC to another PC without interruption. I also want to introduce "virsh", which is the command line tool for libvirt, and also show how you can connect to your virtualization hosts from a remote system, so that you don't need a desktop installation to run your VMs on.

The G Spot: Your go-to destination for all things that are Android Gaming

December 1, 2019 By Dave Prochnow Android, Gaming, ODROID-C2, ODROID-N2

Our long wait is finally over: Google Stadia, this universal game-changing streaming service, has gone live now! It debuted late November, 2019.

As discussed in earlier articles from this column, on this launch date, Google Stadia will ONLY be available to subscribers of the Stadia Founder's Edition. As you may recall, this $129 package includes a controller, Chromecast Ultra for TV, 3 months of Stadia Pro and a "FREE" streamed copy of Destiny 2. If you are not already a Founder's Edition subscriber, then you will have to wait for three months before you can experience Stadia.

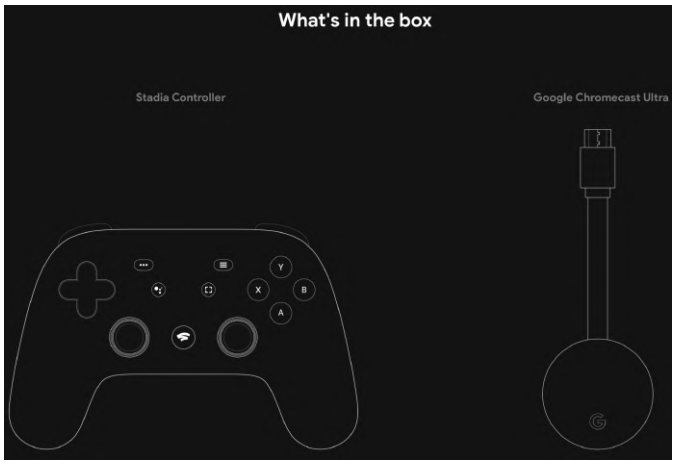


Figure 1 - Founder's Edition subscribers get some hardware with their purchase

After the three month Stadia perk for Founder's subscribers has lapsed, then regular subscribers can join in the game-streaming party. This Stadia Pro option will cost $9.99 per month (three months of this option are included with the Founder's Edition

subscription) PLUS the cost for purchasing any games. No free lunch here, yet.

At October's Made by Google event, there was a mention that a "free tier" would be launched in 2020. This option will include stereo sound and 1080p game streaming. The free option is in stark contrast with the previously mentioned Stadia Pro which will feature 4K game streaming, HDR color, 5.1 surround sound and one free game per month. Regardless of your subscription tier, you will receive at least 60 frames per second (fps) for game playback.

Figure 2 - The Stadia controller

Unfortunately, there were some surprising "details" mentioned at the October event that disappointed many of the Stadia faithful.666

Bundled inside these surprising caveats that caught some Stadia followers by surprise was the limitation on mobile devices that will be able to access Stadia at the time of launch. Only Pixel 3 and 3a phones (and presumably the new Pixel 4 family of smartphones) and Chrome OS tablets (e.g., Pixel Slate, HP Chromebook X2, etc.) will have access. Furthermore, the Stadia controller will only work in wireless mode with Chromecast Ultra. Any computer, however, with a Google Chrome browser will work with the Stadia controller using a wired USB connection and have access to Stadia. Additionally, you will be able to use third-party controllers with Chrome on your Stadia-streaming computer.

Figure 3 - Make your TV a Stadia "play" with Chromecast Ultra7

As for games, Google had previously stated that 31 games would be ready on the Stadia launch day. Since the bulk of these promised titles have already been released, this goal should be attainable (and do not forget two of these 31 titles are reliable DOOM standards). Luckily, lots of game developers want to jump onto the Stadia bandwagon, so this list of games is increasing. In fact, both Red Dead Redemption 2 and Orcs Must Die 3 should join the Stadia party in March 2020.

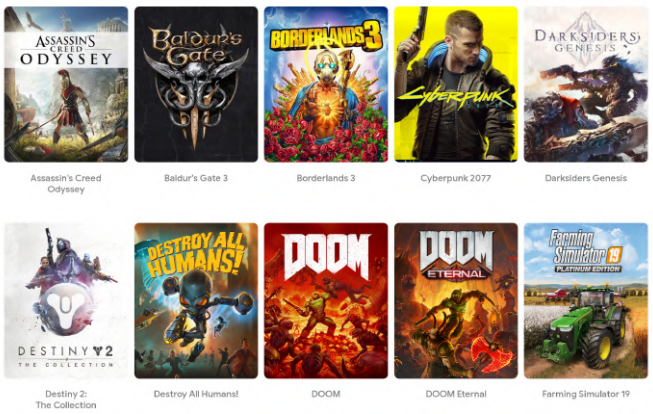


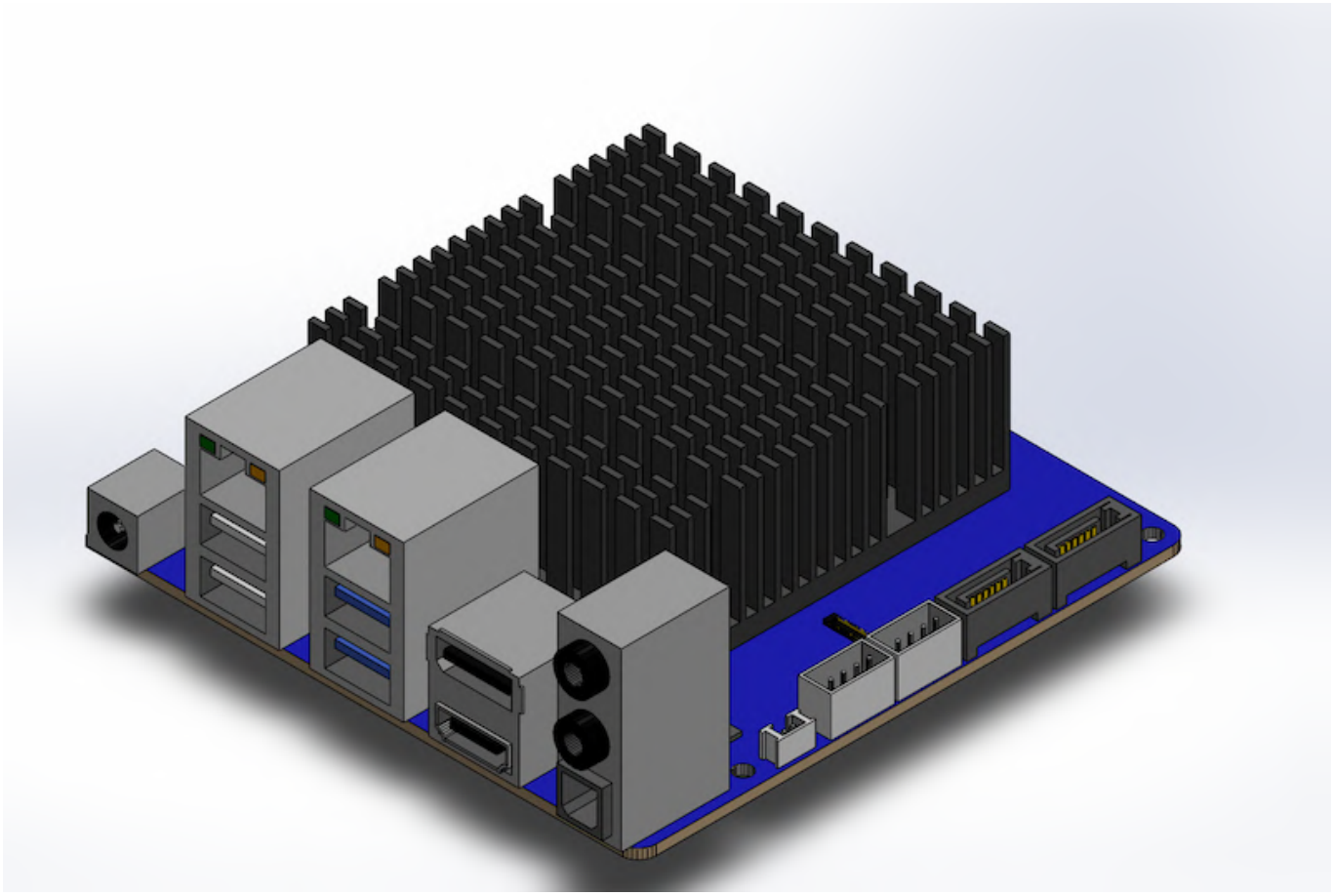
Figure 4 - A Stadia game sampler; Destiny 2: The Collection is included free for Founder's subscribers

Is this a bumpy start for Stadia or just a minor, forgettable glitch? According to Stadia chief, Rick Osterloh, "Stadia is aiming to deliver the best games ever made to just about any screen in your life". Enough said.

If you are still confused about all of this Stadia stuff, there is a Google-made video at https://www.youtube.com/watch?v=Pwb6d2wK3Qw that explains "how Stadia works" or, as its also known, "Stadia 101".

Building An ODROID-H2 BladeCenter: Create A Micro-footprint High Performance Computing Station

December 1, 2019 By @rvalle ODROID-H2, Tinkering



After receiving inspiration from the excellent OpenSCAD H2 Model posted at https://forum.odroid.com/viewtopic.php?f=172&t=33824, I created a remix of the fantastic Raspberry Blade Center to house 3 ODROID-H2 units. I made the following changes to the original project:

- Migrated the files to SolidWorks
- Built an assembly
- Aligned the fasteners
- Widened the cart by 2mm so that it would remain fit in its place
- Made pillars for PSM brass inserts by both Heatlok and Minitech

The SolidWorks files are available at https://www.thingiverse.com/thing:3929164.

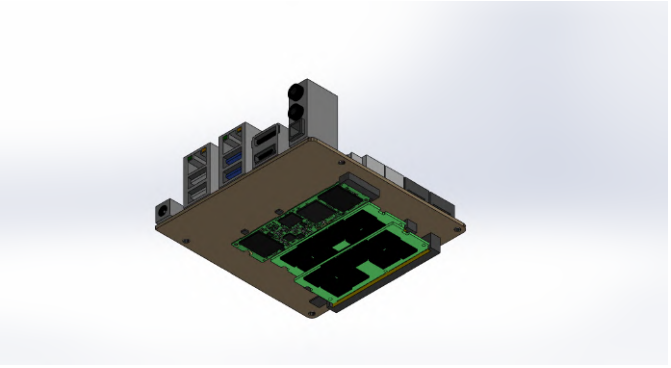


Figure 1 - For a more realistic model, here is an assembly of the ODROID-H2 with peripherals and heat-sink on

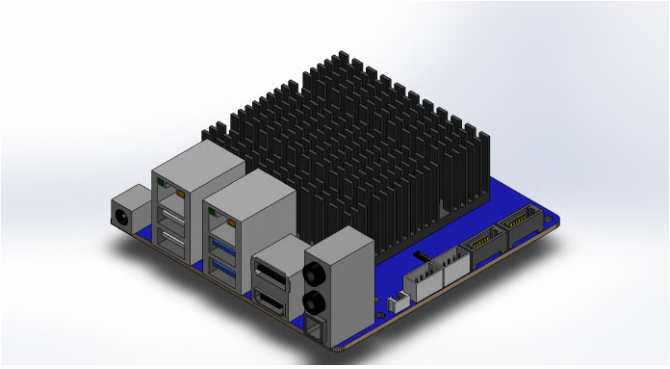


Figure 2 - Here is another view of the assembly of the ODROID-H2 with peripherals and heat-sink on

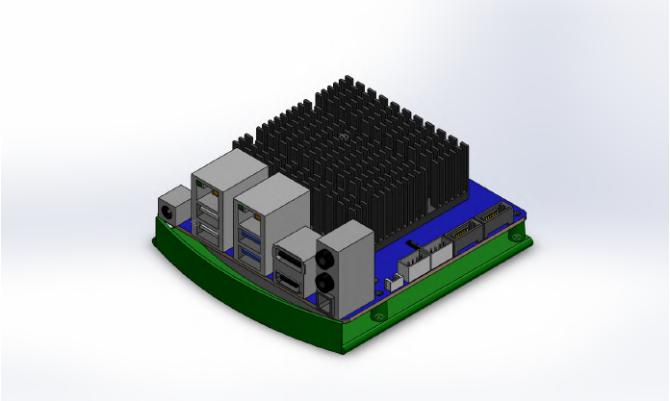


Figure 3 - The design of the caddy

Figure 4 - The design of the caddy

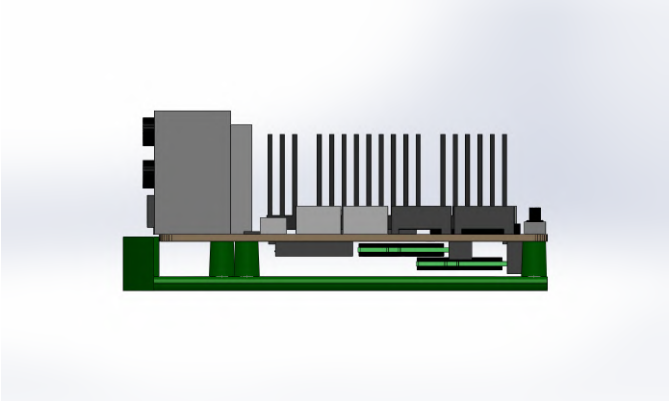


Figure 5 - The design of the caddy

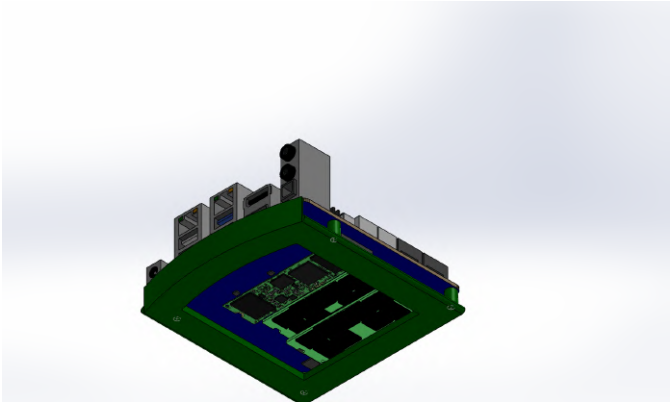


Figure 6 - The design of the caddy

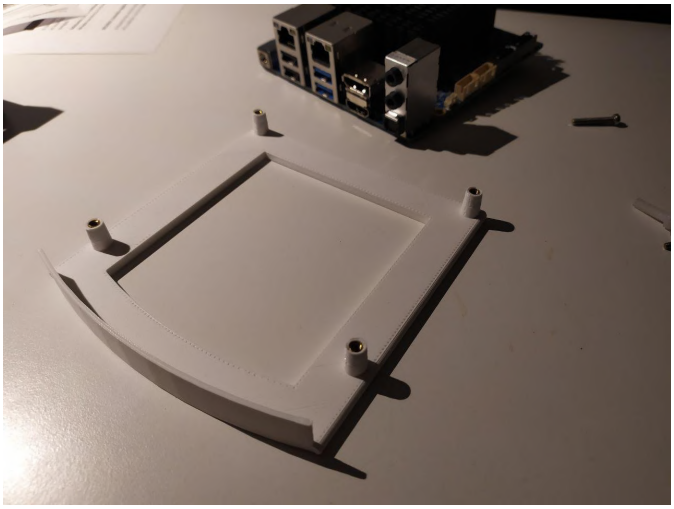


Figure 7 - The first 3D print, which pretty much works, but needs thicker pillars

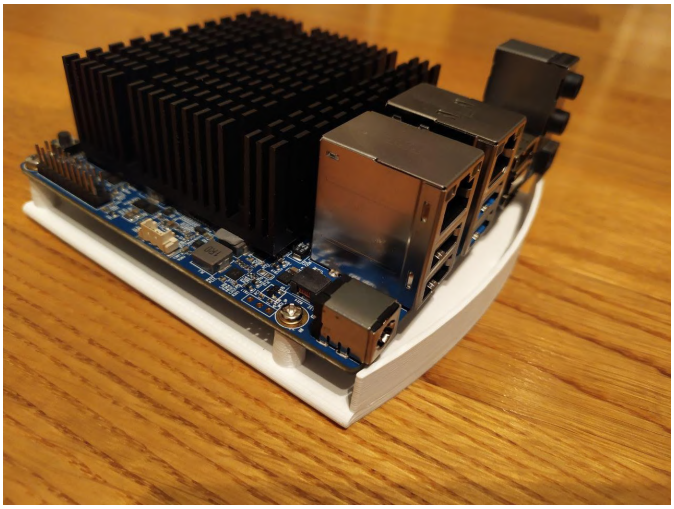


Figure 8 - The first 3D print, which pretty much works, but needs thicker pillars

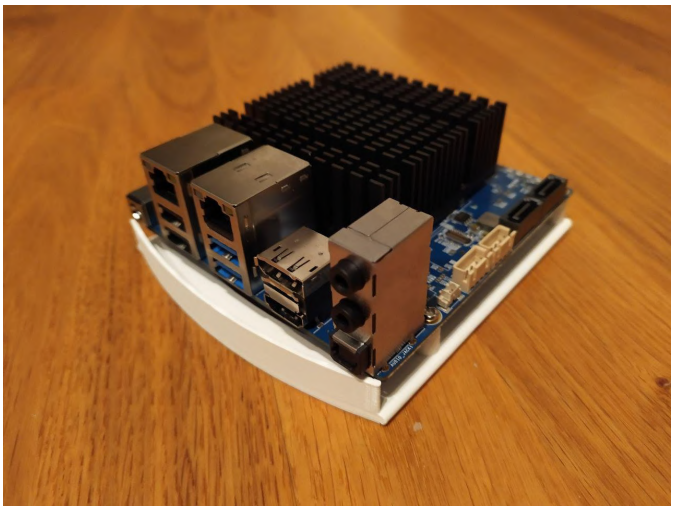


Figure 9 - The first 3D print, which pretty much works, but needs thicker pillars

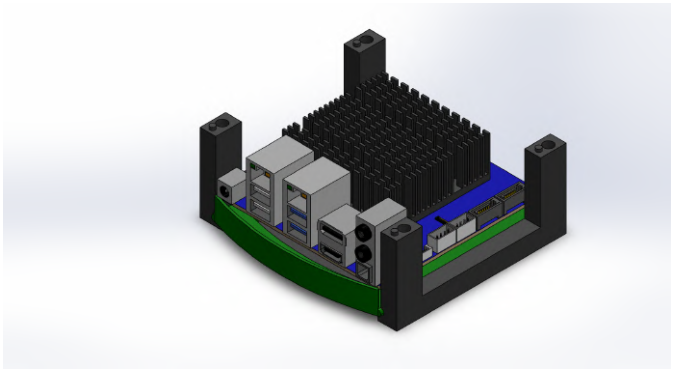


Figure 10 - I decided to make an unusual hack, letting each audio connector embed itself into the next block

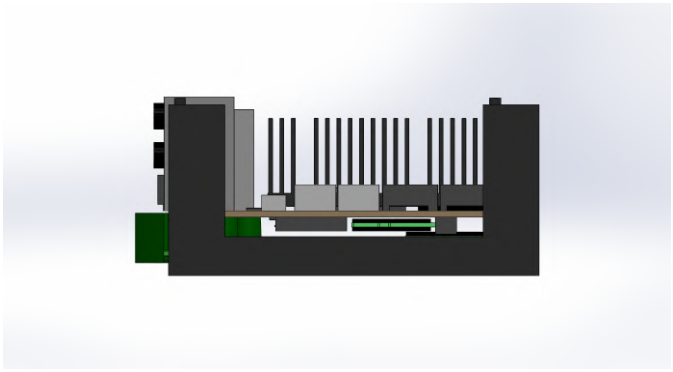


Figure 11 - The audio block is higher than the mount, as you can see on this side view

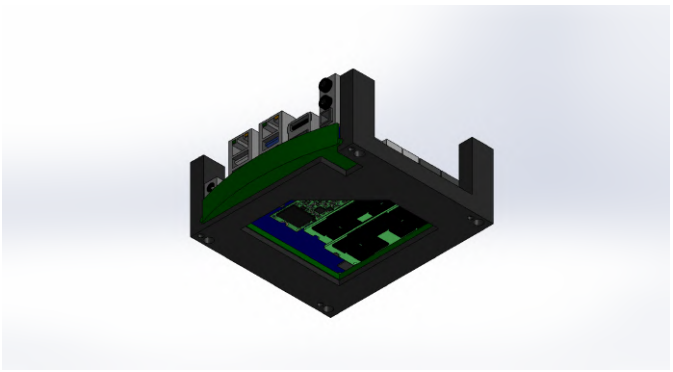


Figure 12 - A socket has been created on the base of the mount to embed the audio connector from the previous H2 block

Figure 13 - Here you can see how each audio connector embeds in the next block, saving space to allow 8 units in 3Us

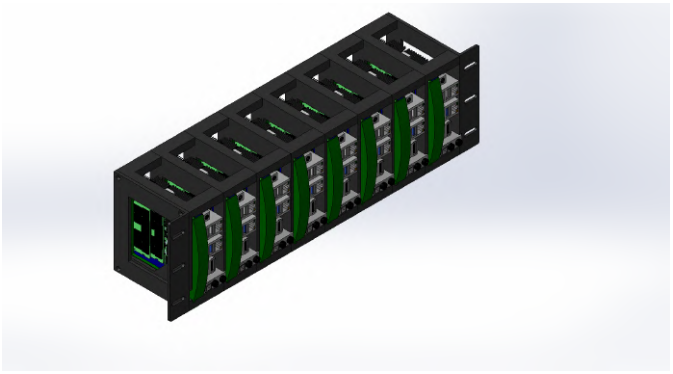


Figure 14 - Once you put them together, the blade center looks like this

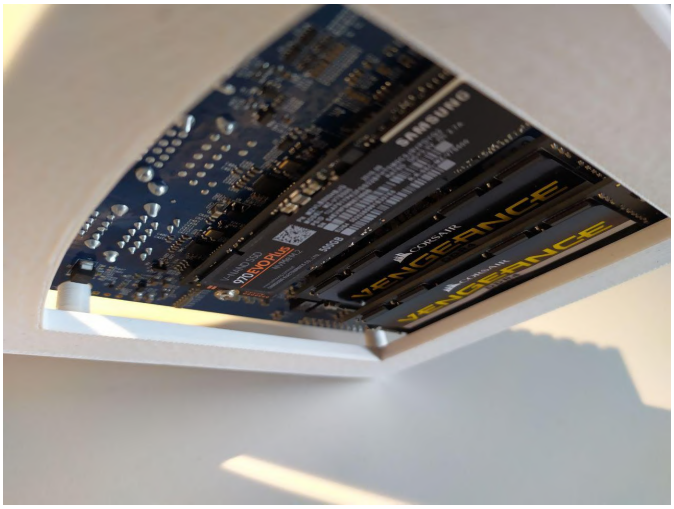


Figure 15 - After receiving the RAM and SSD

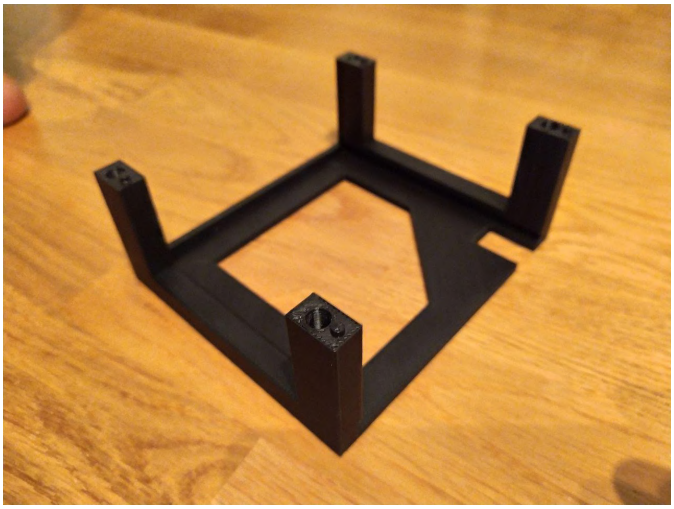


Figure 16 - The first mount print

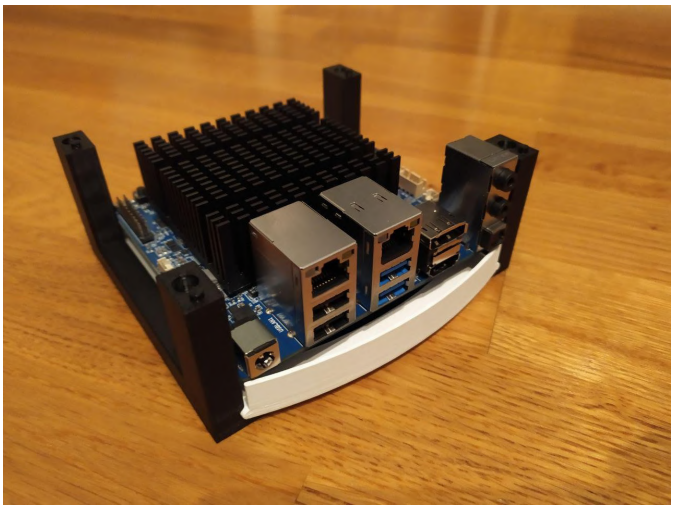


Figure 17 - The first mount print

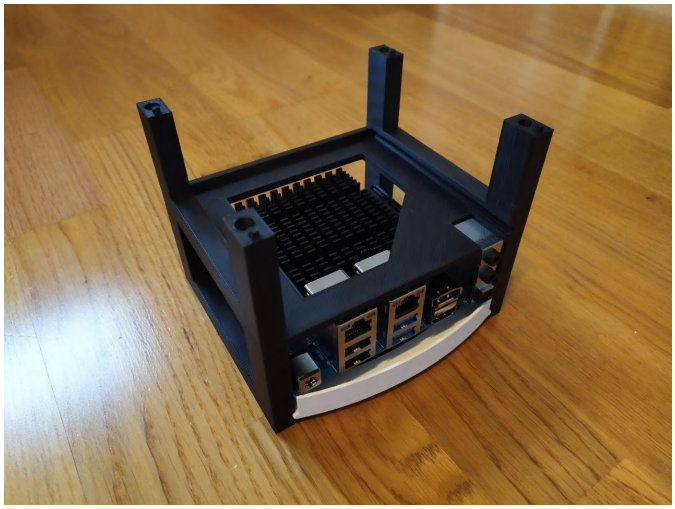


Figure 18 - The second mount print, where the alignment knobs work well, but the holes need to be cleaned of plastic for them to do their job properly

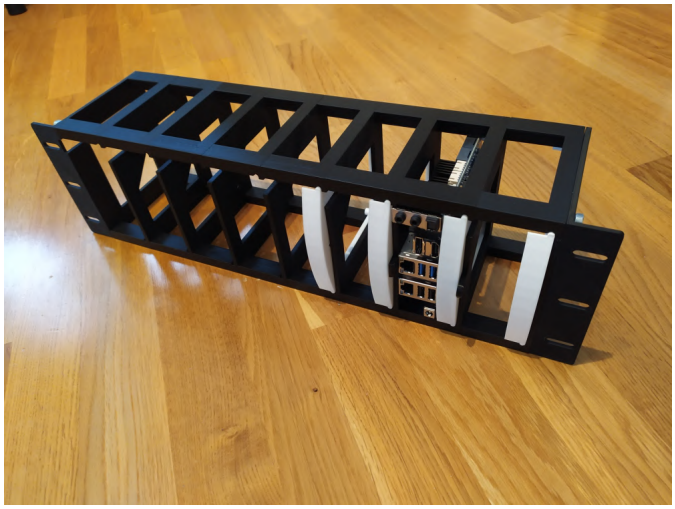


Figure 19 - The entire rack assembled, where I implemented a press fit system so that the blades don't come off the mount easily, with a click and a helper tab to get them out, similar to the Ethernet cables

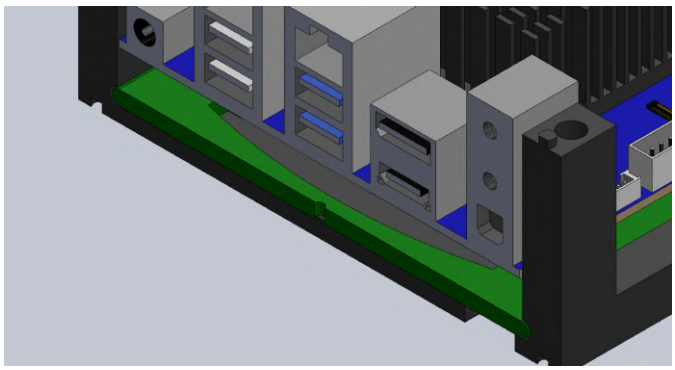


Figure 20 - Here is a section to see the lock system working

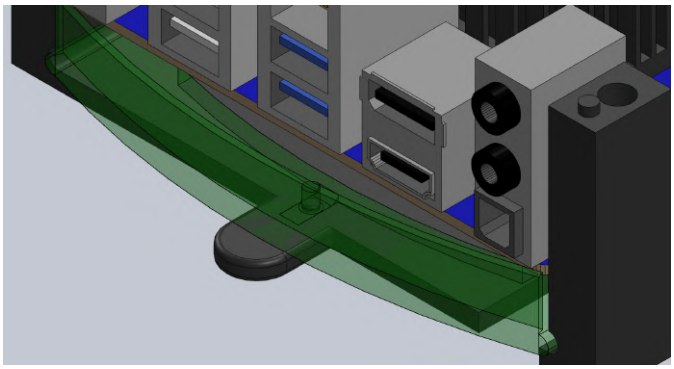


Figure 21 - Here you can see how the caddy and tab interact

Figure 22 - The final assembly mounted into a rack, with 3 ODROID-H2 units installed

Required Hardware

- 4x Rods, M6x44.5 cm (Metal or Nylon)
- 4x M6 Nuts
- Optional 32x M3 HeatLok Brass Inserts from PSM International or similar.
- Optional 32x M3x5mm screws for Insert Version
- Optional 32x self tapping screws 5-6mm long (hole is 2.5mm wide)

Printed Parts

- 8x H2 Caddy (either Inserts or Self tapping version) Version with inserts recommended, but requires brass inserts.
- 8x H2 Mount part (vanilla, pressfit or pressfit&tab version) recommended the one with tab.
- 1x Ear Left
- 1x Ear Right

Equipment

- 8x H2 Boards
- 3U available in a rack (small depth/network rack is OK)
- PSU

Pending Issues

I have not tested the Self-tapping screws version of the caddy The distributed PSU is not rack friendly.

ODROID-H2 Design Feedback For the particular use case of using the H2 as a rack blade, the following issues have been found:

- The audio connectors are too tall, and should stay within the limits of the other connectors
- The positioning of the power/reset switches is inconvenient, and it would be better to have them in the front of the caddy
- The positioning of the LEDs is also inconvenient

For more information, comments, suggestions, and questions, please visit the project page at https://www.thingiverse.com/thing:3929164 or the ODROID Forum post at https://forum.odroid.com/viewtopic.php?t=36780&p=272149.

Surviving A Power Outage: Operating e-Commerce Business During Regional Power Outage

December 1, 2019 By www.ameridroid.com Tutorial

Imagine if the electricity to your business was cut off. But not just your business -- your city, your county and your entire region! That is what ameriDroid was faced with due to PG&E's Public Safety Power Shutoff that started on Saturday, October 26, 2019, and lasted until the afternoon of Wednesday, October 30 affecting our warehouse in Northern California.

Figure 1

Our Challenge

ameriDroid's staff did not want to delay the shipping of packages to our customers, and also did not want to have customers confronted with dead phone lines when trying to call in. So we knew we could not just

kick back in our beach chairs with tropical drinks until the power came back on!

Figure 2

Our Infrastructure

Because ameriDroid is a single board computer (**https://ameridroid.com/collections/single-board-computer**) distributor, retailer (**https://ameridroid.com/collections/all**), and wholesaler (**https://ameridroid.com/pages/corporate-orders**), we want to be intimately familiar with the products we sell. In addition to research and theory, using our products in the real-world is one of the best ways to do this. We decided from the beginning that we would operate as much as possible using only the products we sell. We are also committed to minimizing our impact on the environment. We do this in several ways:

- We are completely paperless in our operations other than: cardboard boxes and packing material we use for shipping, thermal shipping labels, thermal packing receipts
- We recycle any excess cardboard, paper and packing material we get from other sources
- Our shipping staff use Android tablets for the majority of their work in the warehouse
- Our warehouse infrastructure runs on ARM-based ODROID SBCs (**https://ameridroid.com/collections/odroid**) with one low-power embedded Intel system (**https://ameridroid.com/products/beelink-sii-mini-pc**) (for shipping peripherals that run on Windows, like digital scales and address label printers) - this allows for redundancy and low power requirements
- 

Figure 3

- 
- Our video security system runs on custom-designed Raspberry Pi camera units with an ODROID-U3 acting as a DVR, which is an extremely power-efficient system
- 

Figure 5

- Our site premises security system runs on an ODROID-XU4Q and battery-powered wireless door and motion sensors, again extremely power-efficient
- Our peripheral storage facility runs on an off-grid solar array, but is too far from our warehouse to act as a power supply for the main shipping location

We employ Internet connectivity from 3 different providers for redundancy, but only one ended up being viable during the regional outage:

- Comcast - Down due to regional equipment failure
- Verizon - Marginally operational with high latency and about 50% packet loss due to overloaded networks
- Pacific.net Bonded DSL - Fully operational

Because our parent company, Respectech (**http://respectech.com/**), provides our main infrastructure on industry-standard equipment, we had to set up a temporary network to allow ameriDroid to operate on the Pacific.net connection. Respectech's server rack takes 30A of power to operate, so it was not feasible to power these Windows- and CentOS-based servers on our off-grid solution. ameriDroid's phone systems also operate off Respectech's infrastructure. Fortunately, ameriDroid's calls automatically fail-over to our staff's mobile phones during an outage.

Our Solution

Figure 6

On our first day of off-grid operations, we enlisted the ameriBus to provide power with its 4000W peak pure sine wave inverter. The ameriBus has limo-style perimeter seating, so we installed this inverter to allow the ameriBus to act as a portable conference room and demo facility for our SBC solutions at fairs, conferences, and when visiting our west-coast clients. It also came in handy for our unexpected off-grid requirements. Although this system worked perfectly well, the ameriBus consumed about one gallon of fuel for every 4 hours of idling due to the big v10 engine. We knew we could do better. The following day, we took a 1000W pure sine wave inverter being used on our peripheral storage facility's solar power system and connected it to ameriDroid's shipping van. The shipping van has a much smaller 6 cylinder engine. This change allowed us to operate on approximately 1 gallon of fuel for every 8 hours of operation. Because we operate on SBCs, tablets, and low-power thermal printers, 1000W of current was more than enough. An ironic challenge was that even during the day, our shipping staff had to use LED-powered headlamps to work in the warehouse as the overhead lighting wasn't able to be powered up.

Figure 7

The Outcome

We lost a few hours of standard productivity by having to set up off-grid power distribution solutions and building a temporary network for our shipping infrastructure. Other than that, we were nearly fully operational for the extent of the 5-day-long Public Safety Power Shutoff event, and most customer orders went out on schedule. The few that didn't make the shipping cut-off went out the very next day. We'll be even more prepared for the next challenge, thanks to the possibilities provided by single-board computers!

Reference

https://ameridroid.com/blogs/ameriblogs/news-operating-e-commerce-business-during-regional-power-outage

Repairing Your ODROID-N2: How To Recover From An Accidental Short Circuit

December 1, 2019 By Justin Lee ODROID-N2, Tinkering

In this guide we will show you how to repair Q1 on your ODROID-N2. The original article was taken from the Hardkernel wiki page available here: https://wiki.odroid.com/odroid-n2/hardware/repairs

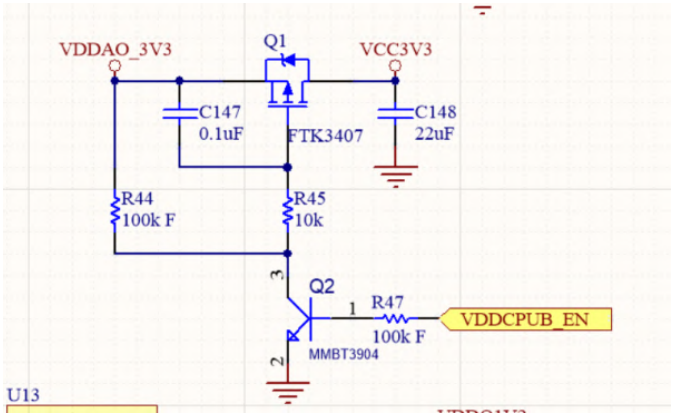


Figure 1 - Discrete Load Switch Schematic with FTK3407

Causes of Q1 Transistor Failure

Over-currents, even for a short duration, can cause progressive damage to a MOSFET, often with little noticeable temperature rise before failure. MOSFETs often carry a high peak-current rating, but these ratings typically assume peak currents only lasting 300 µsec or so. And cumulative over-current may cause damage.

The causes of Q1 failure are generally from built-in current limiting circuits in integrated power switches to prevent the load SW from being destroyed during over-currents. In the case of discrete load SW, the current limiting circuit is not built in, so the P-CH FET FTK3407 is damaged due to over-currents as shown above.

- When the eMMC is mounted to reverse, over-current flows to VCC3V3 due to the FLASH_1V8 node short circuit to GND.
- GPIO connection error or GPIO short-circuit on the 40-pin IO port.

Failure to function -- due to failure

As shown below, the load switch does not supply the VCC3V3 node due to damage and its power supply is not supplied, so booting to the eMMC, USB, Ethernet and IO pin etc. does not work.

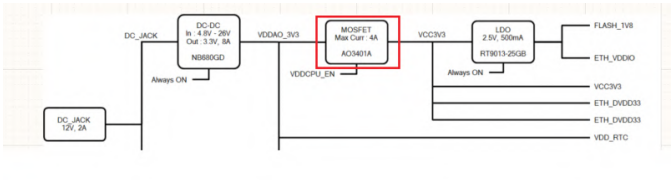


Figure 2 - Block diagram of MOSFET that is damaged

Repair procedure

Note:

- You can also replace AO3407A alternatively since FTK3470 is not easily accessible.
- Make sure to leave the components R44, R45, and C147 nearby Q1 alone. Small parts could go flying by your wayward soldering iron.

Step 1. Basically, you could see the Q1 damaged by its appearance as shown in figure 3.

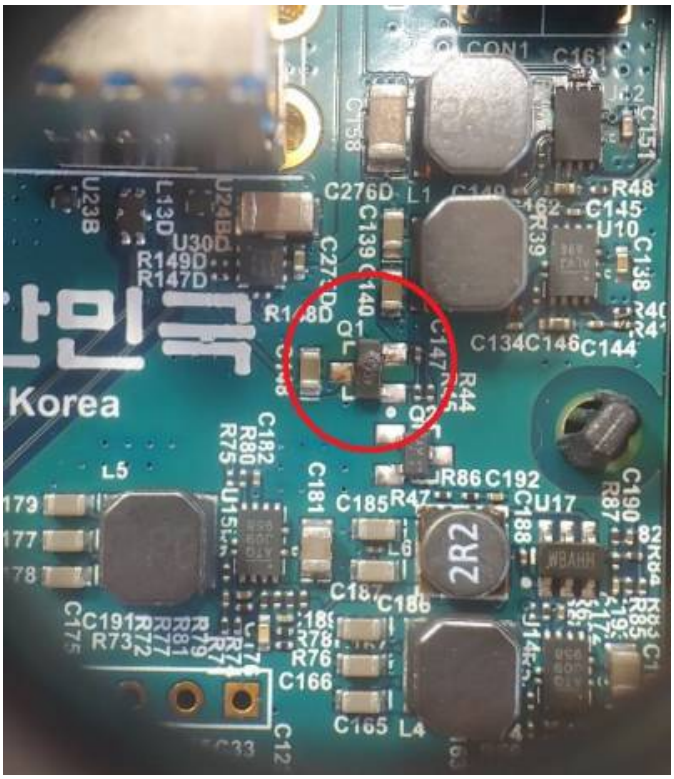


Figure 3 - Q1 outlined in red

Step 2. To easily detach Q1, you should add a little solder on the pad of Q1.

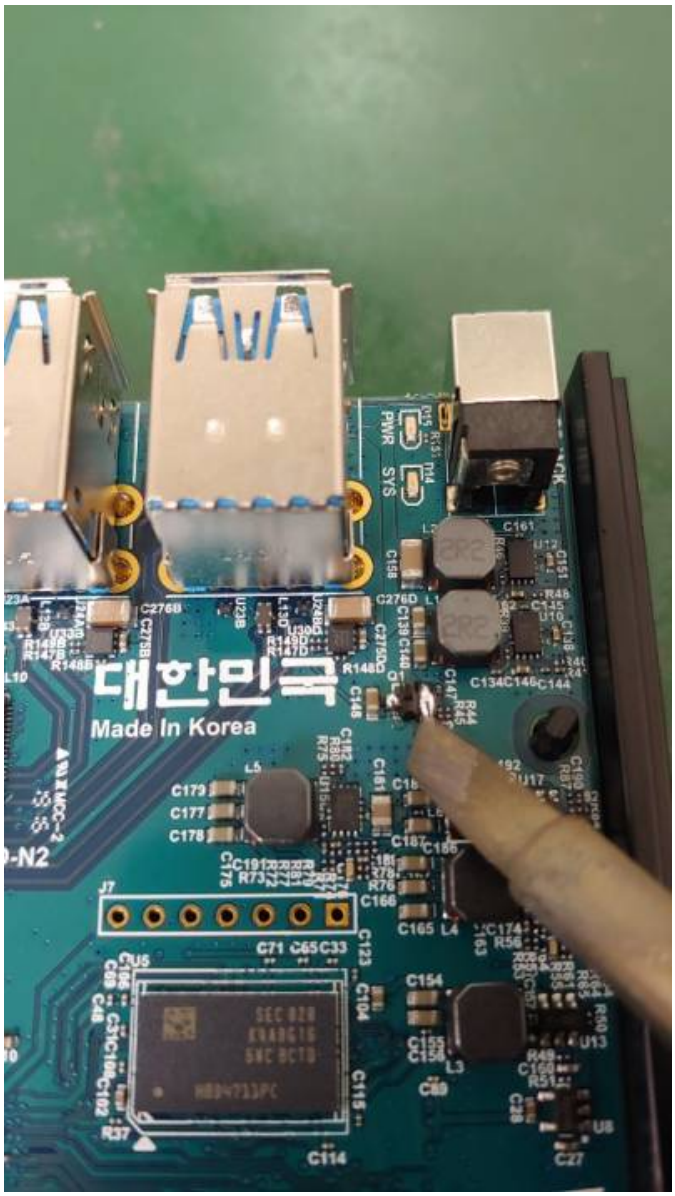


Figure 4 - Solder added to Q1 for easier removal

Step 3. If you have a spare soldering iron, use it as shown in the image. If you don't have a spare soldering iron, it will be harder than using a single soldering iron, but you could try melting solder on the three pads at the same time and then pull part out.

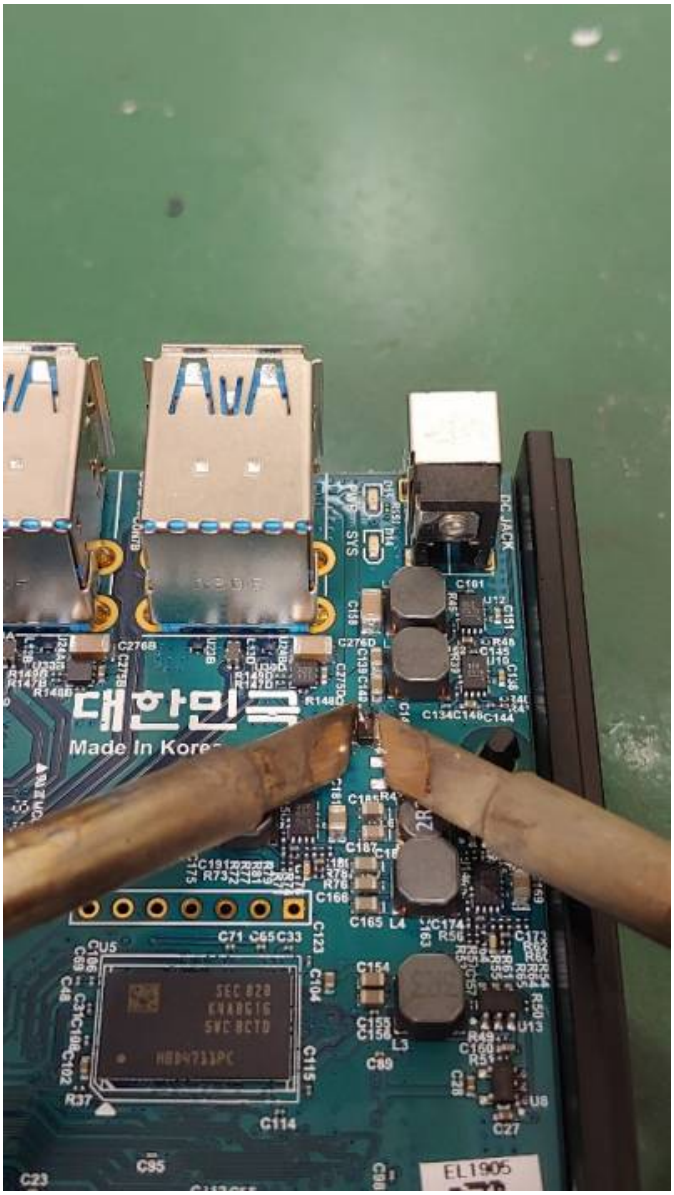


Figure 5 - Using 2 soldering irons to remove Q1, heating all of the pads at the same time

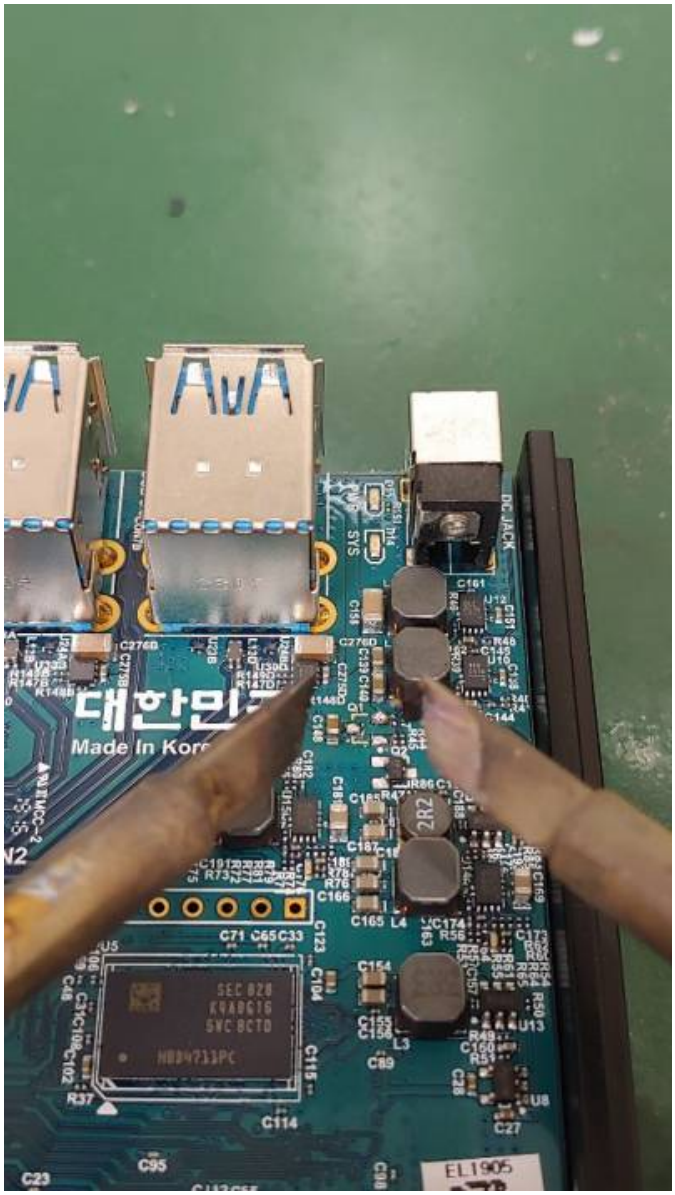


Figure 6 - Q1 removed

Step 4. Remove the rest of the solder on the pad. Use a solder wick for solder removal.

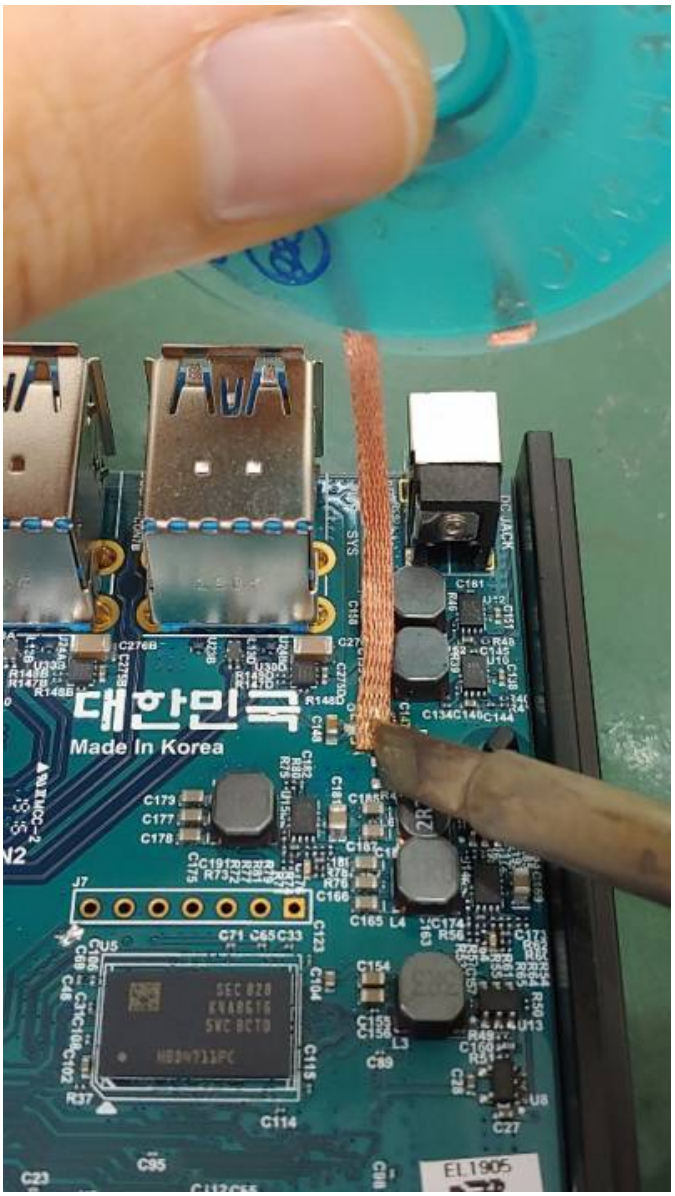


Figure 7 - Copper wick used to remove any residual solder

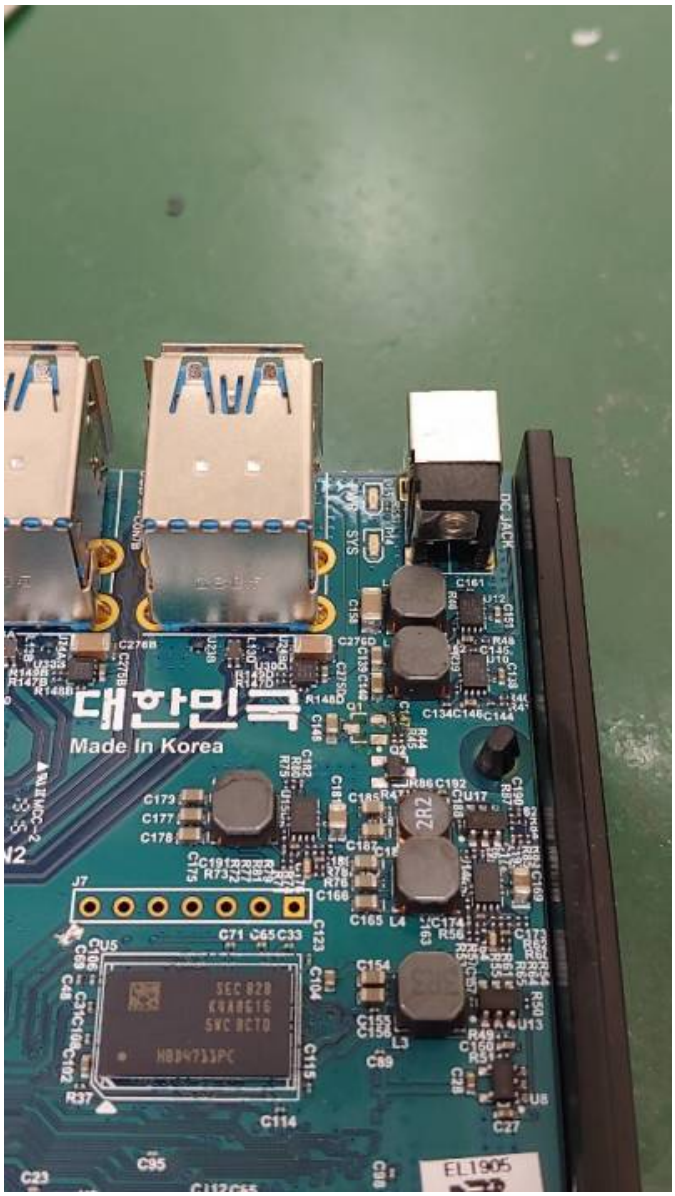


Figure 8 - Solder removed

Step 5. Use SMD tweezers and soldering the new Q1 into place.

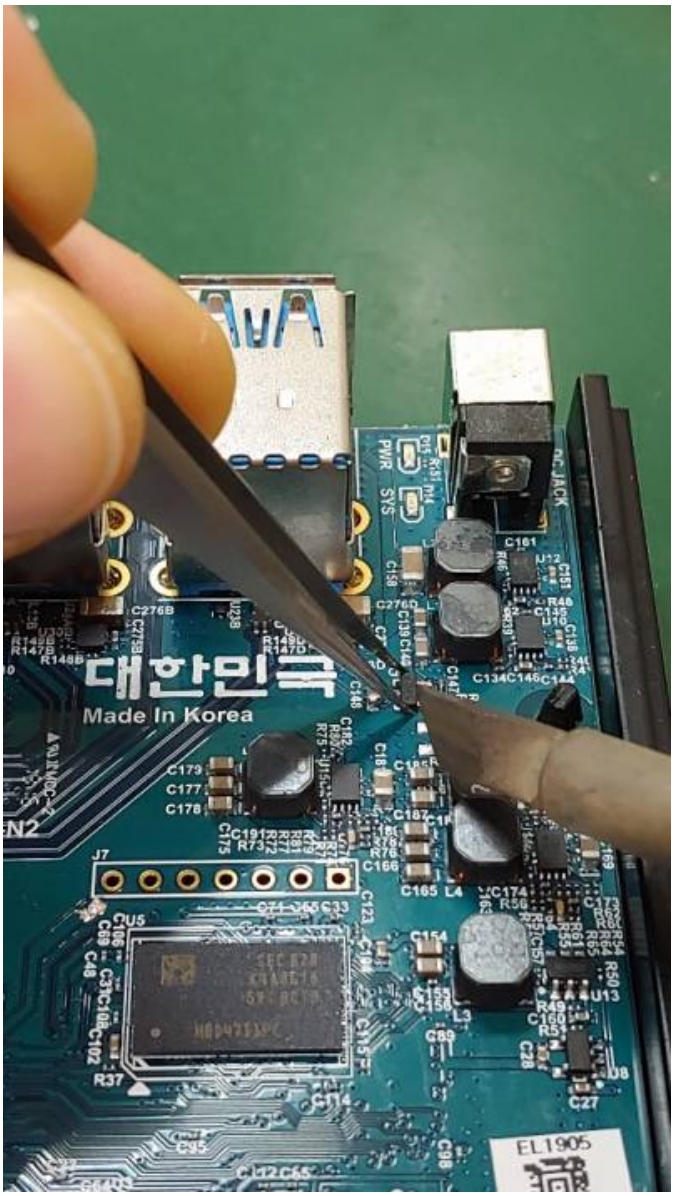


Figure 9 - New MOSFET held in place with tweezers

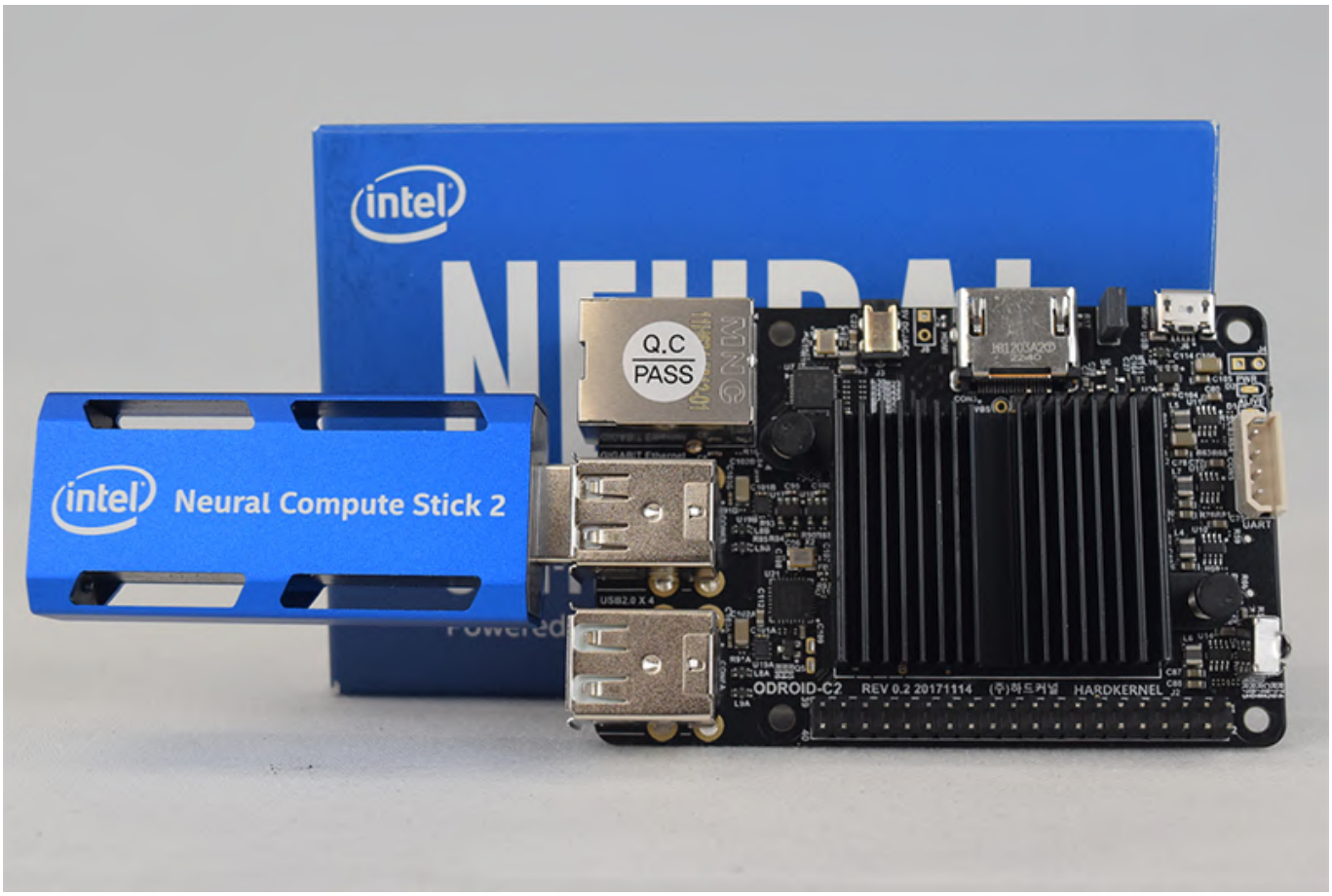
Figure 10 - New MOSFET solder in place

Conclusion

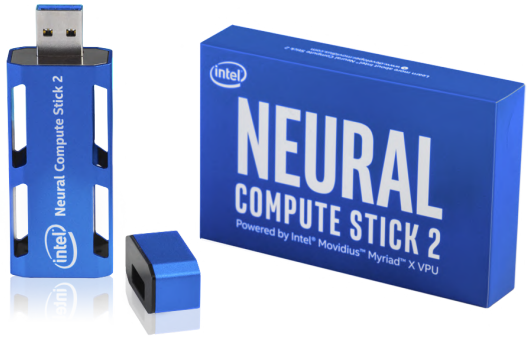
We hope that this guide helped you with repairing your ODROID-N2. After replacing Q1 (P-CH FET), your USB, Ethernet, Expansion Connector 3.3V power, and eMMC booting should work perfectly.

Creating a Vision Application In Low Power Situations: Using OpenVino and OpenCV With The ODROID-C2

December 1, 2019 By software.intel.com Linux, ODROID-C2, Tutorial



The Intel® Distribution of the OpenVINO™ toolkit and the Intel® Neural Compute Stick 2 (Intel® NCS 2) are the perfect complement for vision applications in low-power development environments. Getting setup on so many different architectures presents opportunities. ARMl platforms such as ARM64 are becoming increasingly common for developers building and porting solutions with low-powered single-board computers (SBCs). These can have widely varying requirements compared to traditional x86 computing environments. While the Intel® Distribution of the OpenVINO™ toolkit provides a binary installation for multiple environments, including the popular Raspberry Pi* SBC, the open-source version of the Intel® OpenVINO™ toolkit offers developers the opportunity to build the toolkit and port application(s) for various environments. HARDKERNEL CO., LTD's ODROID-C2 is a microcomputer similar to the Raspberry Pi. The ODROID-C2 is an ARM64 platform with a powerful quad-core processor and plenty of RAM (2 GB) for multiple applications. This article will guide you on your journey of setting up an ODROID-C2 with Ubuntu* 16.04 (LTS), building CMake*, OpenCV, and Intel® OpenVINO™ toolkit, setting up your Intel® NCS 2, and running a few samples to make sure everything is ready for you to build and deploy your Intel® OpenVINO™ toolkit applications.



Neural Compute Stick Package

Although, these instructions were written for the ODROID-C2*, the steps should be similar for other ARM* 64 SBCs such as ODROID-XU4 as long as your environment is using a 64-bit operating system. If your device uses a 32-bit operating system supporting at least the ARMv7 instruction set, visit this ARMv7 article: **https://intel.ly/2DyJjpt**. For general instructions on building and using the open source distribution of the OpenVINO™ toolkit with the Intel® Neural Compute Stick 2 and the original Intel® Movidius™ Neural Compute Stick please take a look at the article at: **https://intel.ly/2P9kQga**.

Hardware

Make sure that you satisfy the following requirements before beginning. This will ensure that the entire install process goes smoothly:

- ARMv7 SBC such as the Orange pi PC Plus
- AT LEAST an 8GB microSD Card. You may utilize the onboard eMMC module if one is attached, but you will need a microSD card to write the operating system to the board
- Intel® Neural Compute Stick 2
- Ethernet Internet connection or compatible wireless network
- Dedicated DC Power Adapter
- Keyboard
- HDMI Monitor
- HDMI Cable
- USB Storage Device
- Separate Windows*, Ubuntu*, or macOS* computer (like the one you're using right now) for writing the installer image to device with a compatible microSD card reader

Setting Up Your Build Environment

This guide assumes you are using the root user and does not include sudo in its commands. If you have created another user and are logged in as that user, run these commands as root to install them correctly.

Make sure your device software is up to date:

```
$ apt update && apt upgrade -y
```

Some of the toolkit's dependencies do not have prebuilt ARMv7 binaries and need to be built from source – this can increase the build time significantly compared to other platforms. Preparing to build the toolkit requires the following steps:

- Installing build tools
- Installing CMake* from source
- Installing OpenCV from source
- Cloning the toolkit

These steps are outlined below:

Installing Build Tools

Install build-essential:

```
$ apt install build-essential
```

This will install and setup the GNU C and GNU CPlusPlus compilers. If everything completes successfully, move on to install CMake* from source.

Install CMake* from Source

The open-source version of Intel® OpenVINO™ toolkit (and OpenCV, below) use **CMake*** as their build system. The version of CMake in the package repositories for both Ubuntu 16.04 (LTS) and Ubuntu 18.04 (LTS) is too out of date for our uses and no official binary exists for the platform – as such we must build the tool from source. As of writing, the most recent stable supported version of CMake is 3.14.4. To begin, fetch CMake from the Kitware* GitHub* release page, extract it, and enter the extracted folder:

```
$ wget
https://github.com/Kitware/CMake/releases/down
load/v3.14.4/cmake-3.14.4.tar.gz
```

```
$ tar xvzf cmake-3.14.4.tar.gz
$ cd ~/cmake-3.14.466
```

Run the bootstrap script to install additional dependencies and begin the build:

```
$ ./bootstrap
$ make –j4
$ make install
```

The install step is optional, but recommended. Without it, CMake will run from the build directory. The number of jobs the make command uses can be adjusted with the –j flag – it is recommended to set the number of jobs at the number of cores on your platform. You can check the number of cores on your system by using the command grep –c ^processor /proc/cpuinfo. Be aware that setting the number too high can lead to memory overruns and the build will fail. If time permits, it is recommended to run 1 to 2 jobs. CMake is now fully installed.

Install OpenCV from Source

Intel® OpenVINO™ toolkit uses the power of OpenCV to accelerate vision-based inferencing. While the CMake process for Intel® OpenVINO™ toolkit downloads OpenCV, if no version is installed for supported platforms, no specific version exists for ARMv7 platforms. As such, we must build OpenCV from source. OpenCV requires some additional dependencies. Install the following from your package manager (in this case, apt):

- git
- libgtk2.0-dev
- pkg-config
- libavcodec-dev
- libavformat-dev
- libswscale-dev

Clone the repository from OpenCV GitHub* page, prepare the build environment, and build:

```
$ git clone
https://github.com/opencv/opencv.git
$ cd opencv && mkdir build && cd build
$ cmake –DCMAKE_BUILD_TYPE=Release –
DCMAKE_INSTALL_PREFIX=/usr/local ..
$ make –j4
$ make install
```

OpenCV is now fully installed.

Download Source Code and Install Dependencies

The open-source version of Intel® OpenVINO™ toolkit is available through GitHub. The repository folder is titled dldt, for Deep Learning Development Toolkit.

```
$ git clone https://github.com/opencv/dldt.git
```

The repository also has submodules that must be fetched:

```
$ cd ~/dldt/inference-engine
$ git submodule init
$ git submodule update --recursive
```

Intel® OpenVINO™ toolkit has a number of build dependencies. The install_dependencies.sh script fetches these for you. There must be some changes made to the script to run properly on ARM* platforms. If any issues arise when trying to run the script, then you must install each dependency, individually. For images that ship with a non-Bash POSIX-Compliant shell, this script (as of 2019 R1.1) includes the use of the function keyword and a set of double brackets which do not work for non-Bash shells. Using your favorite text editor, make the following changes.

Original Line 8 :

```
function yes_or_no {
```

Line 8 Edit:

```
yes_or_no() {
```

Original Line 23:

```
if [[ -f /etc/lsb-release ]]; then
```

Line 23 Edit:

```
if [ -f /etc/lsb-release ]; then
```

The script also tries to install two packages that are not needed for ARM: gcc-multilib and gPlusPlus-multilib. They should be removed from the script, or

all other packages will need to be installed independently.

Run the script to install:

```
$ sh ./install_dependencies.sh
```

If the script finished successfully, you are ready to build the toolkit. If something has failed at this point, make sure that you install any listed dependencies and try again.

Building

The first step, for beginning the build, is telling the system the location of the installation of OpenCV. Use the following command:

```
$ export OpenCV_DIR=/usr/local/opencv4
```

The toolkit uses a CMake building system to guide and simplify this building process. To build both the inference engine and the MYRIAD plugin for Intel® NCS 2, use the following commands:

```
$ cd ~/dldt/inference-engine
$ mkdir build && cd build
$ 6cmake -DCMAKE_BUILD_TYPE=Release
      -DENABLE_MKL_DNN=OFF
      -DENABLE_CLDNN=OFF
      -DENABLE_GNA=OFF
      -DENABLE_SSE42=OFF
      -DTHREADING=SEQ
      ..
$ make
```

If the make command fails because of an issue with an OpenCV library, make sure that you've told the system the location of your installation of OpenCV. If the build completes at this point, Intel® OpenVINO™ toolkit is ready to run. The builds are placed in:

```
/inference-engine/bin/armv7/Release/
```

Verifying Installation

After successfully completing the inference engine build, you should verify that everything is set up correctly. To verify that the toolkit and Intel® NCS 2 works on your device, complete the following steps:

- Run the sample program benchmark_app to confirm that all libraries load correctly
- Download a trained model
- Select an input for the neural network
- Configure the Intel® NCS 2 Linux* USB driver
- Run benchmark_app with selected model and input.

```
Sample Programs: benchmark_app
```

The Intel® OpenVINO™ toolkit includes some sample programs that utilize the inference engine and Intel® NCS 2. One of the programs is benchmark_app, a tool for estimating deep learning inference performance. It can be found in:

```
~/dldt/inference-engine/bin/intel64/Release6
```

Run the following command in the folder to test benchmark_app:

```
$ ./benchmark_app –h
```

It should print a help dialog, describing the available options for the program.

Downloading a Model

The program needs a model to pass into the input. Models for Intel® OpenVINO™ toolkit in IR format can be obtained by:

- Using the Model Optimizer to convert an existing model from one of the supported frameworks into IR format for the Inference Engine
- Using the Model Downloader tool to download a file from the Open Model Zoo
- Download the IR files directly from download.01.org

For our purposes, downloading the files directly is easiest. Use the following commands to grab an age and gender recognition model:

```
$ cd ~
$ mkdir models
$ cd models
$ wget
https://download.01.org/opencv/2019/open_model
_zoo/R1/models_bin/age-gender-recognition-
retail-0013/FP16/age-gender-recognition-
retail-0013.xml
$ wget
https://download.01.org/opencv/2019/open_model
_zoo/R1/models_bin/age-gender-recognition-
```

```
retail-0013/FP16/age-gender-recognition-
retail-0013.bin
```

The Intel® NCS 2 requires models that are optimized for the 16-bit floating point format known as FP16. Your model, if it differs from the example, may require conversion using the Model Optimizer to FP16.

Input for the Neural Network

The last required item is input for the neural network. For the model we have downloaded, you need a 62x62 image with 3 channels of color. This article includes an archive that contains an image that you can use, and it is used in the example below. Copy the archive to a USB Storage Device, connect the device to your board, and use the following commands to mount the drive and copy its contents to a folder called OpenVINO in your home directory:

```
$ lsblk
```

Use the lsblk command to list the available block devices, and make a note of your connected USB drive. Use its name in place of sdX in the next command:

```
$ mkdir /media/usb
$ mount /dev/sdX /media/usb
$ mkdir ~/OpenVINO
$ cp /media/archive_openvino.tar.gz ~/OpenVINO
$ tar xvzf ~/OpenVINO/archive_openvino.tar.gz
```

The OpenVINO folder should now contain two images, a text file, and a folder named squeezenet. Note that the name of the archive may differ – it should match what you have downloaded from this article.

Configure the Intel® NCS 2 Linux* USB Driver

Some udev rules need to be added to allow the system to recognize Intel® NCS 2 USB devices. Inside the attached tar.gz file there is a file called 97-myriad-usbboot.rules_.txt. It should be downloaded to the user's home directory. Follow the commands below to add the rules to your device:

If the current user is not a member of the users group then run the following command and reboot your device:

```
$ sudo usermod –a –G users "$(whoami)"
```

While logged in as a user in the users group:

```
$ cd ~
$ cp 97-myriad-usbboot.rules_.txt
/etc/udev/rules.d/97-myriad-usbboot.rules
$ udevadm control --reload-rules
$ udevadm trigger
$ ldconfig
```

The USB driver should be installed correctly now. If the Intel® NCS 2 is not detected when running demos, restart your device and try again.

Running benchmark_app

When the model is downloaded, an input image is available, and the Intel® NCS 2 is plugged into a USB port, use the following commands to run the benchmark_app:

```
$ cd ~/dldt/inference-
engine/bin/intel64/Release
$ ./benchmark_app –I ~/president_reagan-
62x62.png –m
  ~/models/age-gender-recognition-retail-
0013.xml
$ –pp ./lib –api async –d MYRIAD
```

This will run the application with the selected options. The –d flag tells the program which device to use for inferencing – MYRIAD activates the MYRAID plugin, utilizing the Intel® NCS 2. After the command successfully executes the terminal will display statistics for inferencing. If the application ran successfully on your Intel® NCS 2, then Intel® OpenVINO™ toolkit and Intel® NCS 2 are set up correctly for use on your device.

Inferencing at the Edge

Now that you have confirmed that your ARMv7 is setup and working with Intel® NCS 2, you can start building and deploying your AI applications or use one of the prebuilt sample applications to test your use-case. Next, we will try to do a simple image classification using SqueezeNetv1.1 and an image downloaded to the board. To simplify things the attached archive contains both the image and the network. The SqueezeNetv1.1 network has already

been converted to IR format for use by the Inference Engine.

The following command will take the cat.jpg image that was included in the archive, use the squeezenet1.1 network model, load the model with the MYRIAD plugin into the connected Intel® NCS 2, and infer the output. As before, the location of the sample application is:

```
/inference-engine/bin/armv7/Release/

$ ./classification_sample -i
~/OpenVINO/cat.jpg -m
  ~/OpenVINO/squeezenet/squeezenet1.1.xml -d
MYRIAD
```

The program will output a list of the top 10 results of the inferencing and an average of the image throughput.

If you have come this far, then your device is setup, verified, and ready to begin prototyping and deploying your own AI applications using the power of Intel® OpenVINO™ toolkit.

For more complete information about compiler optimizations, see our Optimization Notice at: **https://intel.ly/33FbQUU**.

Reference

https://software.intel.com/en-us/articles/ARM64-sbc-and-NCS2

How-To Set Up A Basic NAS: Using Samba To Share Files

December 1, 2019 By Miguel Alatorre, ameriDroid Technician Linux, Tutorial

While having a piece of hardware made specifically for NAS applications is ideal, that doesn't mean that it's impossible to make one out of any old computer [or new single-board computer]! Using Samba, anyone can turn an old tower or SBC into a file server!

NOTE: While a Samba server can be setup and run on Windows operating systems, this guide will focus on the Linux OS.

Now boot up your favorite distro of Linux on your machine Editor's note: In case you're curious, the blog picture shows an ODROID-XU4 and 2.5" SSD. Install the latest version of Samba with the following commands:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install samba
```

Before configuring Samba, make sure that the drive is mounted and accessible. To mount it, enter sudo mount {Drive} {Mounting Point}, though make sure to substitute {Drive} and {Mounting Point} with the correct drive and directory respectively. [Something like this: sudo mount /dev/sda1 /mnt/extdrive]

NOTE: To find the correct drive, use the fdisk -l command.

From here we need to dive into the smb.conf file to setup the server. Before playing with the configuration, it is a good idea to make a backup in-case anything goes wrong. To do so, enter:

```
$ sudo cp /etc/samba/smb.comf
/etc/samba/smb.comf.bak
```

Next comes the fun part, to edit the "*.conf" file, use your favorite text editor, or type:

```
$ sudo nano /etc/samba/smb.conf
```

For a simple server enter something similar to the following at the end of the file:

```
[Share]
comment = Shared Files
path = /path/to/share/destination
writable = yes
guest ok = yes
browsable = yes
```

That's all, to access the share on Windows, enter \Samba.Server.IP.Address in the file explorer search bar; for macOS, select the "Go" tab and enter smb://Samba.Server.IP.Address [in either case, replace Samba.Server.IP.Address with the IP address of your Samba server]. For Linux, the process may be different for different distributions, though you should be able to open the file manager and select a "Connect to Remote Host" option and enter smb://Samba.Server.IP.Address.

[Editor's note: Please note that Samba has multiple security options, and the above configuration is a basic configuration for storing files that don't contain sensitive data. If you wish to store and access sensitive data, please study Samba security in more depth.]

The source of this article is available at: **https://ameridroid.com/blogs/ameriblogs/how-to-set-up-a-basic-nas-using-samba**For more product updates and how-to tutorials, follow us on YouTube, our ameriDroid blog, Facebook, Twitter and Instagram!