Linux Gaming • Android Development • Recalbox for ODROID-XU4

ODROID Magazine

Year Three
Issue #32
Aug 2016

Understanding oCam's *Global Shutter*

To improve your camera skills

- Breaking WPA networks using an ODROID-XU4
- Build an ODROID-XU4 cluster using a simple topology for projects

What we stand for.

We strive to symbolize the edge of technology,
future, youth, humanity, and engineering.

Our philosophy is based on Developers.
And our efforts to keep close relationships with
developers around the world.

For that, you can always count on having the quality
and sophistication that is the hallmark of our products.

Simple, modern and distinctive.
So you can have the best to accomplish
everything you can dream of.

HARDKERNEL

Pollin ELECTRONIC

We are now shipping the ODROID-C2 and ODROID-XU4 to EU countries! Come and visit our online store to shop

Address: Max-Pollin-Straße 1
85104 Pförring Germany

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

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

EDITORIAL

The oCam peripheral, which is a high-end 720p camera made specifically for use with ODROID devices, will soon have a new version available, with the main feature being a global shutter instead of a rolling shutter. This allows you to take more professional pictures without distortion. Our feature article this month demonstrates the difference between the shutter types so that you can decide which will work best for your application. There is also a new power supply available for the ODROID-XU4 which delivers much more power to the peripherals. It's available for purchase at http://bit.ly/2aM8yHi.

Volumio also has an upcoming release with many improvements, and we are lucky enough to have a sneak peek at its new features. Adrian continues his series on improving wireless network security with the first of a two-part article on WPA hacking, Nanik details the Android Hardware Abstraction Layer, Nicolas gives an overview of Recalbox, which turns your ODROID-XU4 into a great set-top box, and Michael presents his fantastic high performance coputing cluster. Additionally, we highlight several games for both Android and Linux, including the Löve Engine, which lets you create your own games!

ODROID Magazine, published monthly at http://magazine.odroid.com, is your source for all things ODROIDian.
Hard Kernel, Ltd. • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815
Hardkernel manufactures the ODROID family of quad-core development boards and the world's first ARM big.LITTLE single board computer.
For information on submitting articles, contact odroidmagazine@gmail.com, or visit http://bit.ly/IyplmXs.
You can join the growing ODROID community with members from over 135 countries at http://forum.odroid.com.
Explore the new technologies offered by Hardkernel at http://www.hardkernel.com.

HARDKERNEL

OUR AMAZING ODROIDIAN STAFF:

Rob Roy, Chief Editor

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

Bruno Doiche, Senior Art Editor

Our fellow nutjob have the most strange quirk, whenever he is zapping through TV channels, if The Matrix is on any channels, he stops and watches it until the end. Even if it is one of the sequels, he doesn't mind watching one of his favorite movie series. Having lately being so lucky as to get a TV weekend special where all three movies were being played and on the anime channel The Animatrix as well, he had a blast that although he enjoyed greatly, his wife don't understand what does he see on the movie and just shrugged. And so it came the column on page 12 came to life.

Manuel Adamuz, Spanish Editor

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

Nicole Scott, Art Editor

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

James LeFevour, Art Editor

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

Andrew Ruggeri, Assistant Editor

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

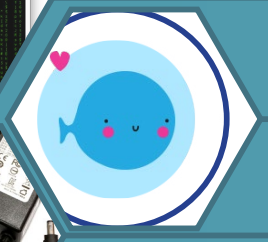
Venkat Bommakanti, Assistant Editor

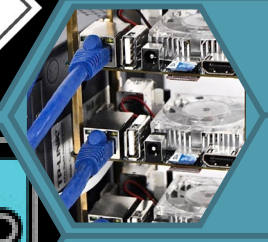
I'm a computer enthusiast from the San Francisco Bay Area in California. I try to incorporate many of my interests into single board computer projects, such as hardware tinkering, metal and woodworking, reusing salvaged materials, software development, and creating audiophile music recordings. I enjoy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

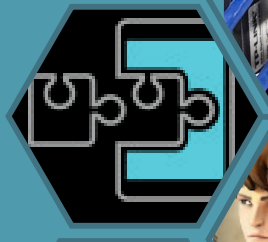
Josh Sherman, Assistant Editor

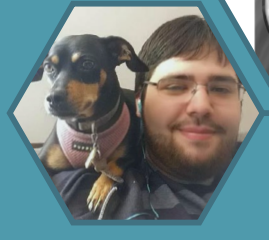
I'm from the New York area, and volunteer my time as a writer and editor for ODROID Magazine. I tinker with computers of all shapes and sizes: tearing apart tablets, turning Raspberry Pis into PlayStations, and experimenting with ODROIDs and other SoCs. I love getting into the nitty gritty in order to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

INDEX









WHY DOES THE PENCIL LOOK BENT?

THE BEHAVIOR OF DIFFERENT TYPES OF CAMERA SHUTTERS

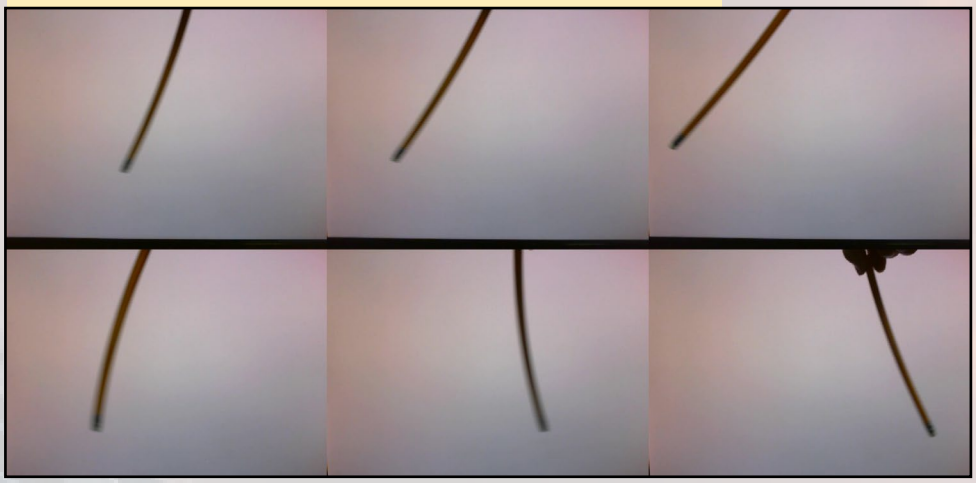
by withrobot@withrobot.com

Let's begin our investigation of camera shutters with a simple experiment using a webcam and a pencil. Hold the pencil directly in front of the webcam and against a white background. Now, try to swing the pencil rapidly back and forth in front of the webcam. As you can see in the captured frames below, the straight pencil suddenly appears to be bent as if by magic.

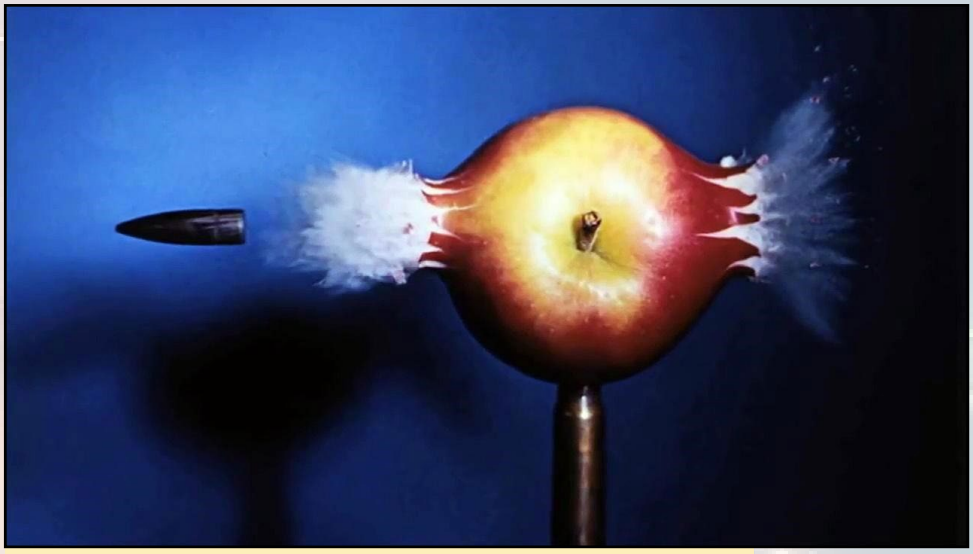
Camera image of a stationary object

What makes the straight pencil to appear bent when it's swinging? To answer this, we need to go one step further into the camera shutter mechanism.

Web Camera frames of a moving pencil

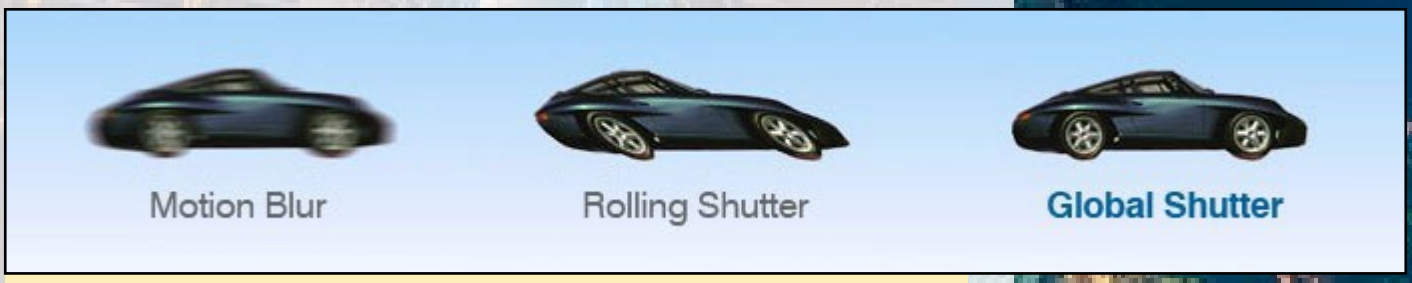


You may have seen interesting photographs taken by a high speed camera. One classic example is of a bullet going through an apple.



High speed cameras are used to capture fast-moving objects

High speed cameras are especially useful for capturing still images of rapidly moving objects such as athletes at sporting events. Additionally, high speed cameras are beneficial when the camera is itself moving, and the object being captured is stationary, such as taking an image from a drone. Unfortunately, we encounter a few unwelcome phenomena during high speed imaging.

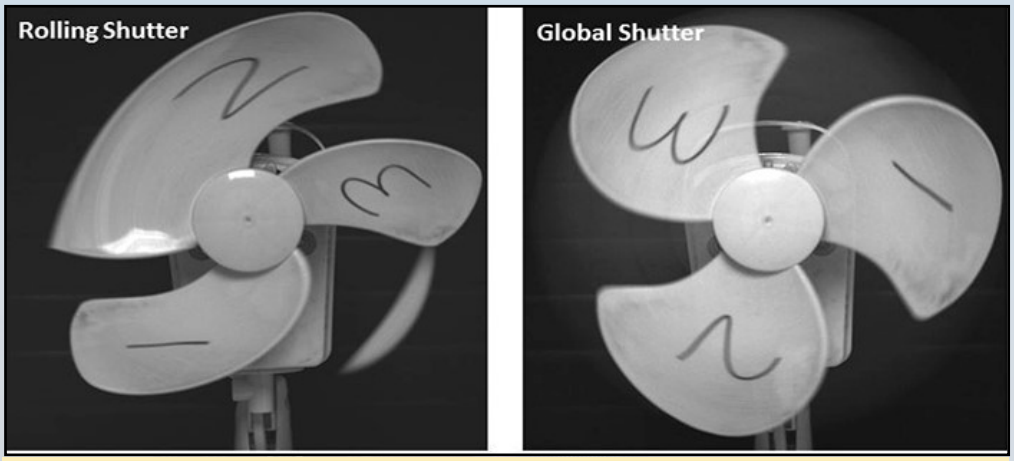


Typical problems in capturing high speed objects. (source: Teledyne DALSA)

The first image, "motion blur", shown above, occurs when a specific point of a moving object appears on more than one pixel due to the excessive exposure time. To reduce motion blur, we need to reduce the exposure time to be as quick as possible. However, a faster exposure time requires additional lighting.

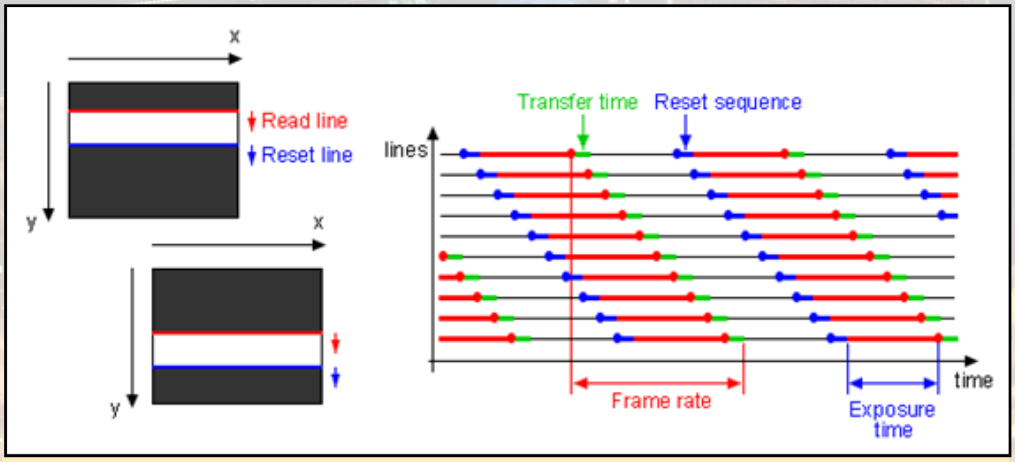
The second phenomenon, and the main topic of this article, is the effect of a rolling shutter. There are two types of electronic camera shutters; one is a rolling shutter and the other one is a global shutter. The two photographs of a rotating fan next page show the difference vividly.

The fan looks deformed in the image taken by a rolling shutter camera while the other image, taken by a global shutter camera, appears to be normal. What causes the difference? The answer lies in understanding how the two shutters works.



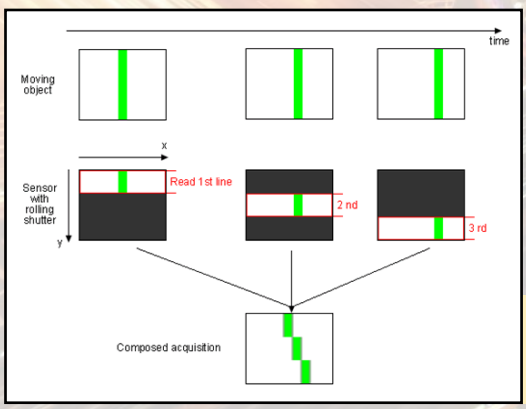
Rotating fan images of two types of shutters: rolling shutter and global shutter. (source: www.andor.com)

A camera with a rolling shutter captures light and sends it out one line after line. The diagram depicted below can help us to understand what is happening. Here, the top line of the image is exposed to light, this only captures a "slice" of the image. Little by little, lines of the image are captured until all the lines form the whole image. Since each line is captured one after another, this means each line was captured as slightly later time.



Exposure of every line starts and ends at different times for rolling shutter. (source: https://www.matrix-vision.com)

How do these different exposure timings deform an image of a fast moving object? For example, if we have a pole moving fast from left to right, the positions captured will be different at each line, as shown here.



We observe this deformation of a fast moving object most dramatically in a photo shot of a swinging golf club.

On the contrary, the cameras using global shutter to expose the whole image at the same time, so

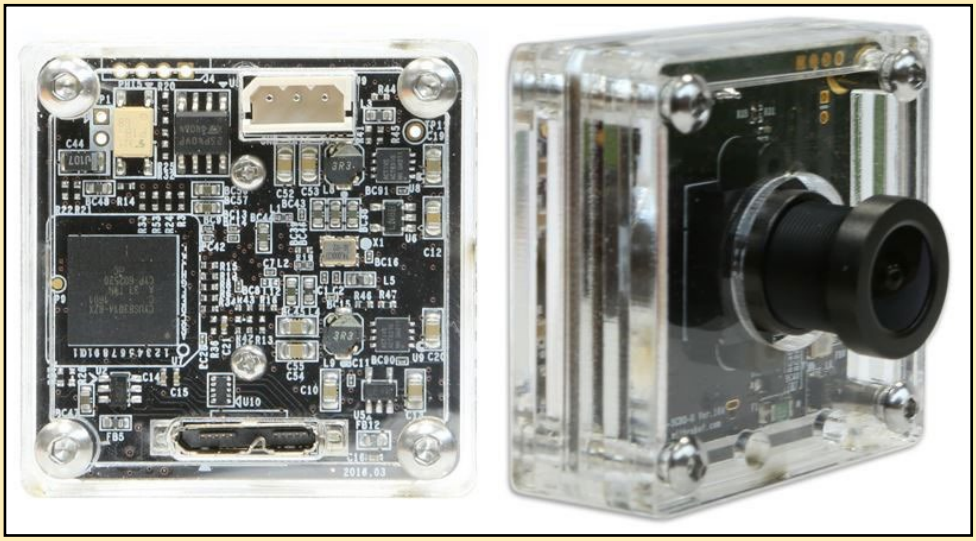
Shifted image of rolling shutter for a fast horizontal object movement

Bent golf club captured by a rolling shutter camera.

they won't have the same deformation problem. In the near future, a new camera will be available to use with ODROIDs. This camera, model oCam-1MGN-U, will have the same form factor as oCam-5CRO-U.

Specifications

Sensor: OnSemi MT9M031 CMOS image sensor
Lens: Standard M12 lens
Image sensor size: 1/3 inch
Pixel size: 3.75um x 3.75um
Shutter: Global shutter
Interface: USB 3.0 super-speed
Frame rate: 1280x960 @45fps, 1280x720 @60fps, 40x480 @80fps, 320x240 @160fps



Global shutter camera for ODROIDs - oCam-IMGN-U

To see the the improvement this camera brings, we captured the same image of the swinging pencil from the first experiment, as shown in Figure 10. Clearly, this camera offers an improved image without any deformation. In the next article, we'll look at an interesting example application that uses this new global shutter camera attached to an ODROID.

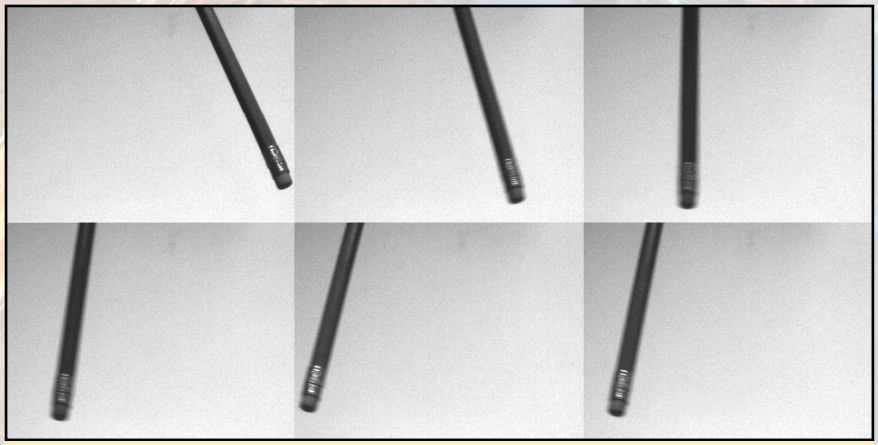


Image of the global shutter camera oCam-IMGN-U for a fast swinging object.

ANDROID DEVELOPMENT

HARDWARE ABSTRACTION LAYER (HAL)

by Nanik Tolaram

When we hear about newest Android devices, we always gets excited and want to be the first to play with them. Android is used in many different consumer devices from cell phones to TVs. Android's flexibility comes from its software stack, which is a layered architecture that makes it easy to port software to different kind of hardware and devices. One of the key layers that plays an integral part is the Hardware Abstraction Layer, or HAL for short. Having strong code support in place for Android's HAL allows boards such as ODROIDs to run Android, and at the same time, allow support for various kind of sensors. In this article, we will look at the HAL layer with a focus on the Bluetooth HAL that is used in the ODROID-C2.

Basic HAL

The Android HAL is the abstraction layer provided by Google that allows for various vendors to support different hardware implementations. The HAL layer is like a translator between Android and the hardware, which makes it convenient for the Android framework to support a range of hardware configurations. This makes it easy for developers to create an app that uses a device's sensors that will work across different Android devices.

Developers don't need to interface with the HAL, as this is taken care internally by the framework. There are possibilities for different hardware running Android and performing differently using the same applications, this is something that developers have no control over.

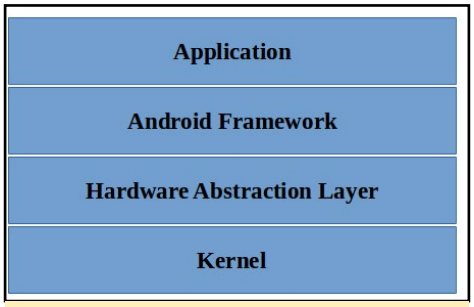


Figure 1 : Android Stack

Figure 1 is a simplified stack of Android's layers that interact with the HAL. The application layer is the Android "app" that is interacting with the HAL via the APIs exposed in the framework. The applications here include the built-in application such as the Settings app that give you the freedom to configure your device, such as Bluetooth functionality, screen brightness, or WiFi settings.

Inside HAL

As previously mentioned, the HAL is a contract or binding between the Android framework and the device hardware. The main code for this binding can be found as in Figure 2. The "/hardware" folder contains several subfolders

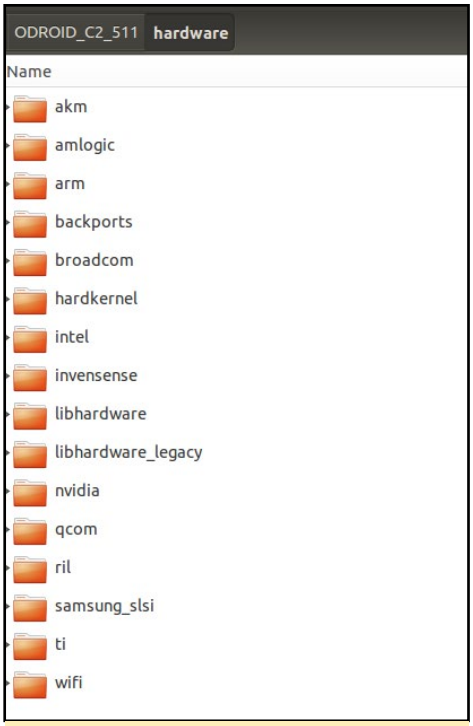


Figure 2 : hardware folder

that contain code for various hardware implementations and the binding code defined in Android.

The folder that is of interest is "libhardware". As seen in Figure 3, there are multiple C header files that corresponds to different hardware bindings defined by Android for devices, such as Fingerprint reader, GPS, Camera, NFC, USB Audio, and many more. We will explore the "bluetooth.h" file that contains the bindings for a Bluetooth device.

Each HAL module uses a constant that is defined as part of the binding.

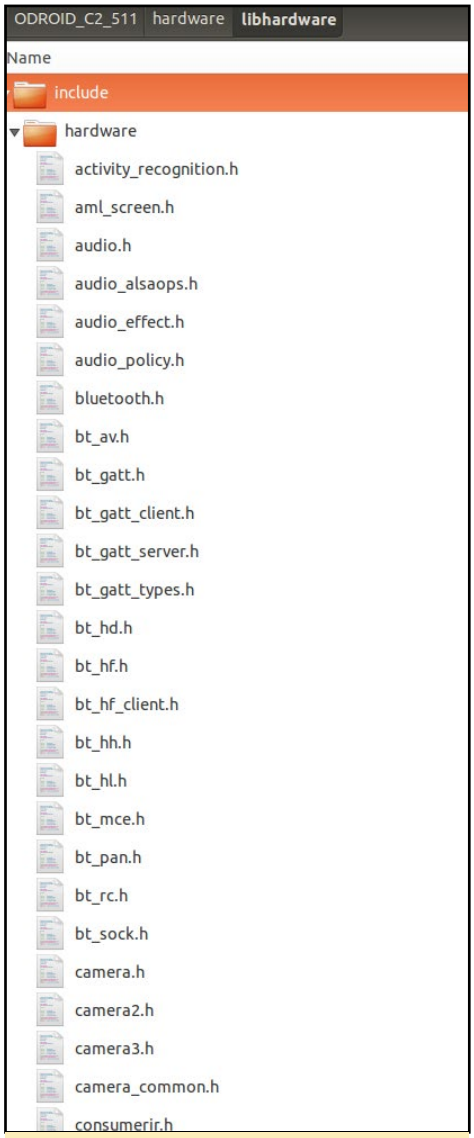


Figure 3 : HAL Binding Header Files

This constant is used by the framework to load the correct library for the relevant hardware. As is seen in Figure 4, the constant used for Bluetooth is defined by a constant named BT_HARDWARE_MODULE_ID.

Figure4 : Bluetooth Header File

```
 * Not a Contribution
 *
 * Copyright (C) 2012 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

#ifndef ANDROID_INCLUDE_BLUETOOTH_H
#define ANDROID_INCLUDE_BLUETOOTH_H

#include <stdbool.h>
#include <stdint.h>
#include <sys/cdefs.h>
#include <sys/types.h>

#include <hardware/hardware.h>

__BEGIN_DECLS

/**
 * The Bluetooth Hardware Module ID
 */

#define BT_HARDWARE_MODULE_ID "bluetooth"
#define BT_STACK_MODULE_ID "bluetooth"
#define BT_STACK_TEST_MODULE_ID "bluetooth_test"
```

The Bluetooth HAL layer binding must implement the bt_interface_t structure defined inside the bluetooth.h header file as shown in the following code segment. In the next section, you will see that the Bluetooth HAL will implement this structure along with the constant defined above.

```
typedef struct {

// .. break in code .. //

    /** Enable Bluetooth. */
    int (*enable)(void);

    /** Disable Bluetooth. */
    int (*disable)(void);

    /** Closes the interface. */
    void (*cleanup)(void);

    int (*ssp_reply)(const bt_
bdaddr_t *bd_addr,
bt_ssp_variant_t variant,
                     uint8_t ac-
cept,
uint32_t passkey);

    /** Get Bluetooth profile in-
terface */
    const void* (*get_profile_in-
terface) (const char *profile_id);
    int (*dut_mode_configure)
(uint8_t enable);

// .. break in code .. //

    int (*read_energy_info)();
    /** BT stack Test interface
*/
    const void* (*get_testapp_in-
terface)(int test_app_profile);
    /** rssi monitoring */
    bt_status_t (*le_lpp_write_
rssi_threshold)(const bt_bdaddr_t
*remote_bda,
char min, char max);
        bt_status_t (*le_lpp_
read_rssi_threshold)(const bt_
bdaddr_t *remote_bda);
```

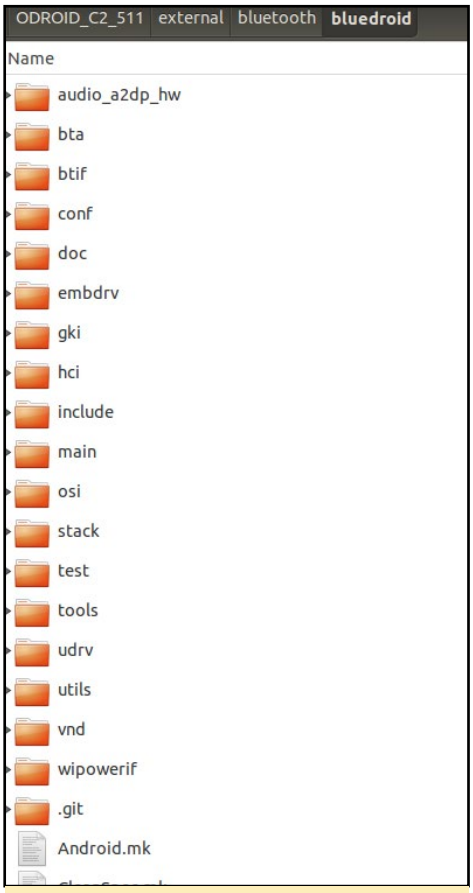


Figure 5 : Bluedroid Bluetooth Implementation

```
// .. break in code .. //

} bt_interface_t;
```

Bluedroid Implementation

Bluedroid is the open source project this is implemented in the Android Bluetooth HAL which resides inside the "external/bluetooth" directory, as shown in Figure 5.

Figure 6 : Bluedroid Bluetooth HAL Implementation

```
07          stack->common.tag = HARDWARE_DEVICE_TAG;
08          stack->common.version = 0;
09          stack->common.module = (struct hw_module_t*)m
10          stack->common.close = close_bluetooth_stack;
11          stack->get_bluetooth_interface = bluetooth__g
12      }
13      else
14      {
15          ALOGE("%s: malloc() returned NULL", __FUNCTIO
16      }
17      *abstraction = (struct hw_device_t*)stack;
18      return 0;
19  }
20
21
22  static struct hw_module_methods_t bt_stack_module_met
23      .open = open_bluetooth_stack,
24  };
25
26  struct hw_module_t HAL_MODULE_INFO_SYM = {
27      .tag = HARDWARE_MODULE_TAG,
28      .version_major = 1,
29      .version_minor = 0,
30      .id = BT_HARDWARE_MODULE_ID,
31      .name = "Bluetooth Stack",
32      .author = "The Android Open Source Project",
33      .methods = &bt_stack_module_methods
```

THE MATRIX

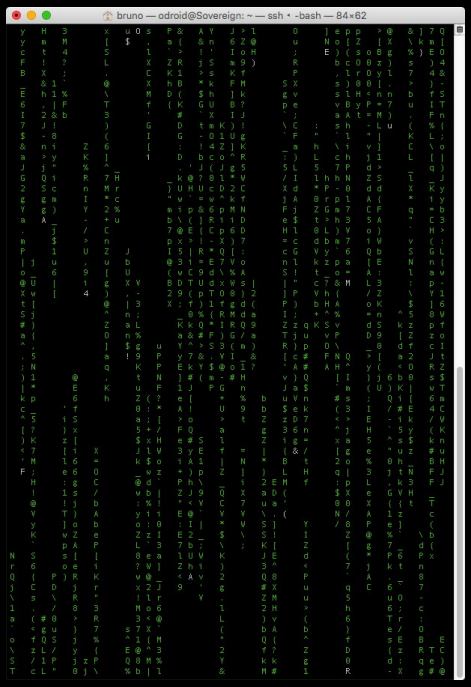
IT WILL ALWAYS BE COOL

by Bruno Doiche

It was the end of that distant age called the Nineties, when boy bands roamed the world and girls wore chokers and sky-high platform shoes.

At that time, a movie hit the theatres that was different and cool, called The Matrix. Along with it, everybody began installing screensavers to emulate the iconic terminals with the Matrix code running on it. If your inner nerd wants to see it again on your screen, just run this command and let your ODROID take you back to your teenage years:

```
$ sudo apt-get install cmatrix
```



It never lost it's cool.

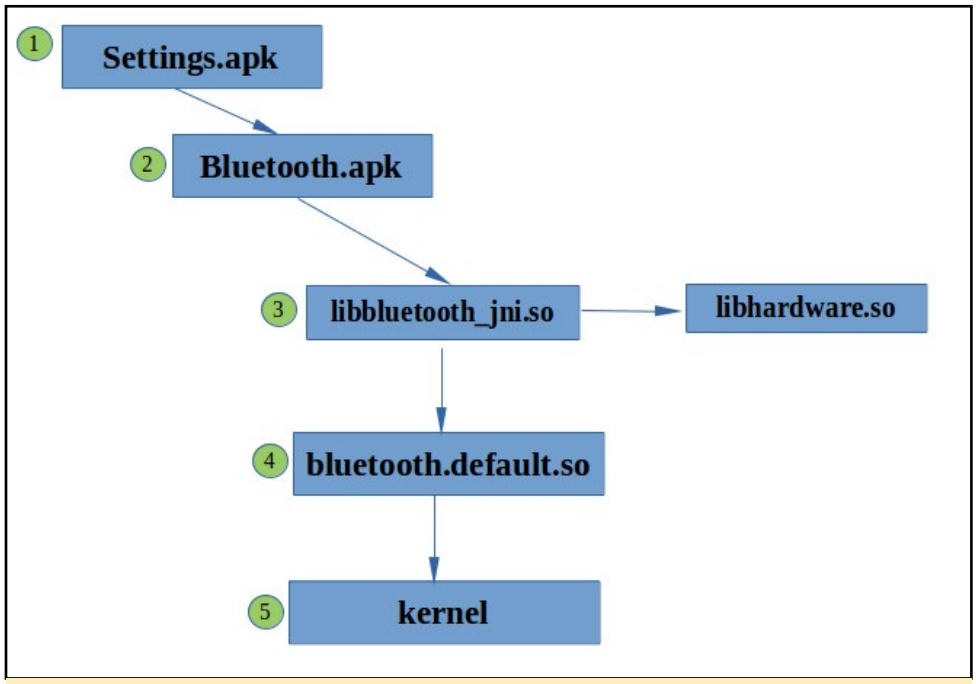


Figure 7 : Bluetooth HAL Interaction Flow

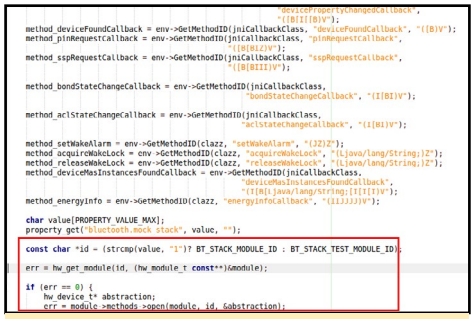
The Bluetooth binding implementation can be found inside the bluetooth.c file that is found inside the "bluedroid/btif/src" folder. The implementation uses the hw_module_t structure, defined by HAL_MODULE_INFO_SYM, which is the standard way to define a HAL structure. This allows the framework to easily find and bind to the HAL implementation. The methods field contains the open function that will be called when initializing the HAL.

Bluetooth HAL

We will now look through the different application layers inside and outside the framework to expose the way it uses the HAL.

The Settings.apk is the an Android application that you normally use to turn the Bluetooth radio on and off, as well as to pair to nearby devices. The Bluetooth.apk is the "bridge" that provides a service for apps to communicate with the Bluetooth HAL and the Bluetooth stack.

The code inside Bluetooth.apk is responsible for loading the libbluetooth_jni.so shared library with the help of libhardware.so, along with making services available, allowing other Android applications to interact with the low level Bluetooth stack. This app is responsible for calling the framework library to load the Bluedroid library, which can be seen inside the file packages/apps/Bluetooth/



```
method_deviceFoundCallback = env->GetMethodID(jniCallbackClass, "deviceFoundCallback", "([B)V");
method_pinRequestCallback = env->GetMethodID(jniCallbackClass, "pinRequestCallback",
                                             "([B[BII)V");
method_sspRequestCallback = env->GetMethodID(jniCallbackClass, "sspRequestCallback",
                                             "([B[BII)V");
method_bondStateChangeCallback = env->GetMethodID(jniCallbackClass,
                                                  "bondStateChangeCallback", "([BI)V");
method_aclStateChangeCallback = env->GetMethodID(jniCallbackClass,
                                                 "aclStateChangeCallback", "([BI)V");
method_setWakeAlarm = env->GetMethodID(clazz, "setWakeAlarm", "(JZ)Z");
method_acquireWakeLock = env->GetMethodID(clazz, "acquireWakeLock", "(Ljava/lang/String;)Z");
method_releaseWakeLock = env->GetMethodID(clazz, "releaseWakeLock", "(Ljava/lang/String;)Z");
method_deviceMasInstancesFoundCallback = env->GetMethodID(jniCallbackClass,
                                                          "deviceMasInstancesFoundCallback",
                                                          "([B[Ljava/lang/String;[I[I)V");
method_energyInfo = env->GetMethodID(clazz, "energyInfoCallback", "(IIJJJJ)V");
char value[PROPERTY_VALUE_MAX];
property_get("bluetooth.mock.stack", value, "");
const char *id = (strcmp(value, "1")? BT_STACK_MODULE_ID : BT_STACK_TEST_MODULE_ID);
err = hw_get_module(id, (hw_module_t const**)&module);
if (err == 0) {
    hw_device_t* abstraction;
    err = module->methods->open(module, id, &abstraction);
```

Figure 8 : Loading Bluedroid Shared Library

jni/com_android_bluetooth_btservice_AdapterService.cpp, as shown in Figure 8.

The Bluedroid bluetooth stack inside the bluetooth.default.so shared library contains the completed Bluetooth specification implementation such as pairing, security, and more. This stack is the only layer responsible to communicate with the kernel layer.

LINUX GAMING

SPREAD SOME LÖVE (ENGINE)

by Tobias Schaaf

In my last article, I talked about a nice little engine that allowed you to play old style RPG games. This month, I want to talk about a different engine called Löve, which facilitates 2D game development. The Löve engine (often called "love") is based on Lua scripts which allows you to easily create games of your own.

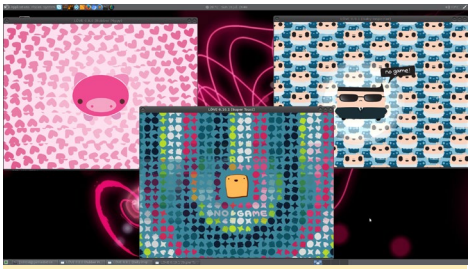


Figure 1 - Löve engine 0.8.0, 0.9.1 and 0.10.1 running on an ODROID-XU3 with no games up

For the ODROID, I created 3 versions of the Löve engine, 0.8.0 (from Debian Wheezy), 0.9.1 from Debian Jessie, and 0.10.1 from git repository. The reason for this is simple. Löve is sadly not as compatible as it should be. A game made for Löve 0.8.0 might no longer work on 0.9.1 or later versions.

In fact, all versions of Löve up to 0.10 required OpenGL, which means they require GLshim to run on the ODROID, which is limited. Starting with 0.10, Löve supports OpenGL ES 1 and 2, which allows for a better portability of games.

Installation

As usual, you can install Löve engine from my repository using the following commands:

```
$ apt-get install love-0.8-odroid
$ apt-get install love-0.9-odroid
$ apt-get install \
  love-0.10-odroid
```

I will add new versions in time when they are released.

Löve version 0.8 and 0.9 require GLshim to work which will be installed as well as a dependency. Löve 0.10 is running on it's own using OpenGL ES 2.

Games

There are many games out there using the Löve engine, in fact many of them you can find directly on the Löve homepage.

Wiki: http://bit.ly/2a3rG0d

Forum: http://bit.ly/2atSJHz

Or in other places such as indiedb.com: http://bit.ly/2afQjdr

There, you can find a large library of Löve games, and it seems that new games come out quite often. There are even some very good commercial Löve game that have been published to Steam and other paid platforms, and some which are still in the making, such as http://bit.ly/2afHs8U. While getting these paid games to work on the ODROID might be a little bit complicated, there are plenty of other games that can be played. I took the time to package some of these games and put them into my repository for easy installation and playing experience. Since I also offer different versions of Löve you can play other games that were made for older versions of Löve as well, side by side with newer games.

MariO

Mari0 is a crossover between Super Mario Bros and Portal, allowing for some crazy game play twists.

Figure 2 - MariO crossover between Super Mario Bros and Portal, even with it's own version of Portal puzzles

It's a little bit hard at first playing Mario with a mouse to aim in the right direction, but using the portal gun can be really fun and allows for some cool moves. The developers, known as "Stabyourself" (http://bit.ly/2afPLEq) even included their own levels of Portal, where you have to solve door puzzles with different cubes and through the use

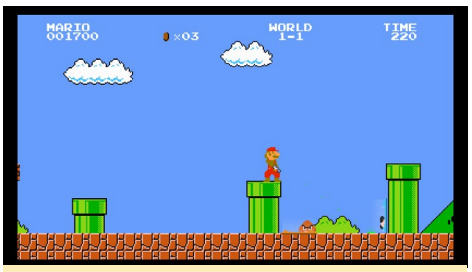


Figure 3 - Having fun with a Gumba! Can you catch them all?

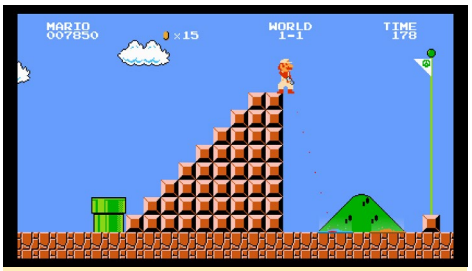


Figure 4 - Making that "high jump" was never easier

of portals.

The game is very fun to play, and although it uses the old Löve 0.8 engine, which requires OpenGL, it runs quite nicely on ODROIDs using GLshim, and also supports multiplayer mode. Did I mention that it has its own build in level editor as well?

Duck Marines

Duck Marines is a reimplementation of the Game ChuChu Rocket from Sonic Team, which is available for consoles such as the Dreamcast.

The game is very fun to play, especially since you can have up to four players on one ODROID. The game is very fast paced, and you have to react quickly on the often changing situations and mini-games. It offers plenty of different maps, and you can create your own levels if you

Figure 5 - Duck Marines from tangram is a remake of ChuChu Rocket

Figure 6 - Get as many ducks to your place as you can

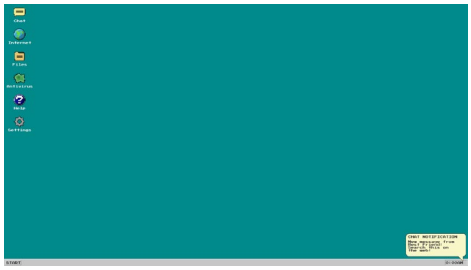
want.

The game runs perfectly fine on ODROIDs, and uses the current 0.10 version of Löve, which means that it runs on native OpenGL ES.

Don't Get a Virus

This game is really unique. It shows what can happen if you go to these shady websites on the Internet that everyone is talking about.

Isn't that how it always starts? A



Figures 7 and 8 - This is how it all starts, talking to a friend or visiting a totally harmless web-site

before you can even do something, suddenly your computer is being attacked by an deadly virus in a "giant space craft with deadly laser guns crafted out of defeated PCs"!

Well at least it's just a game, and you're luckily prepared to defend yourself with your antivirus software.

The game is a fun little shooter and comes in different difficulty levels, but you only have five minutes before the

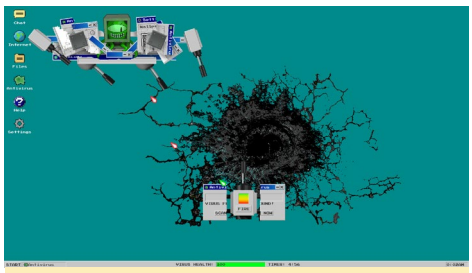


Figure 9 - Evil viruses come in giant space ships as we all know!

virus takes over your computer.

Mr. Rescue

In Mr. Rescue, you're a firefighter going into a burning building in order to rescue the people inside.

Your only weapon is a fire hose with

Figure 10 - Mr. Rescue, do you have what it takes to be a fire fighter?

which you can open doors and windows and push back the flames. The only protection is your fireman suit, which will protect you from the heat, at least for a little while.

The game offers many interesting

Figure 11 - Rescue civilians by throwing them out of the Window before they burn

features, like the line of sight, where you only have a limited view, depending on where you stand and the location of the fires. This makes it harder to find any civilians for rescue, and you never know what awaits you behind the next door, or on the next floor.

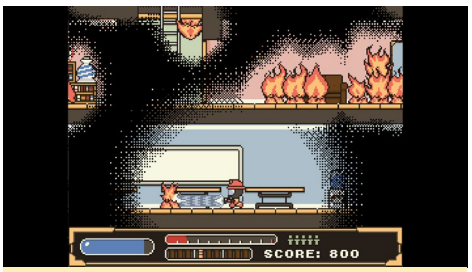


Figure 12 - Extinguish some flames with your trusty fire hose

Your mission is to rescue survivors, and although your instincts tell you to put out every fire you see, that's not why you're in there, and you will lose if you try to extinguish them all.

The blue bar on the bottom left of your screen is your water bar and will reduce when you fire your hose, but will also recover slowly when you stop. If you stop before it's empty, it will recover, and you may shoot again at any given time, even if it's not fully recharged again. However, if you empty that bar completely, you have to wait until it's fully refilled before you can shoot again.

The red bar in the middle of the screen is the most important bar. It displays the heat inside your suit. If it rises to high, you'll die and the game ends. It's really tough to keep your heat in check.

Final thoughts

Löve is a very interesting Engine, and being written in Lua makes it easy for developers to create their own games. If you're interested in building games using Löve, you should check their website at http://love2d.org, where you can find different tutorials and hints, both on the Wiki and the forum. There are some good tutorials as well, such as http://bit.ly/2afRgCN.

Follow my discussion thread in the forum about Löve engine under Games and Emulators for more exciting games, or contact me if you developed your own game and want me to package it for others.

HARDKERNEL 5V/6A POWER SUPPLY

A HIGH QUALITY STABLE AND LOW-NOISE POWER SUPPLY FOR THE ODROID-XU4

by Bruno Doiche

A bunch of ODROID users are always thinking about new things to do with their machines, constantly inventing new configurations, and enjoying their portability. But there are others that use them for the simple and pratical use of making ODROIDs into a multi disk NAS and plugging USB disks into every port available.

This takes an enormous toll on the regular power supply that comes from Hardkernel, and when all disks are running on full throttle powered by the USB ports, we tend to get I/O errors that can easily be mistaken as a disk gone awry, when there is nothing actually wrong with it.

Listening to our needs, Hardkernel has provided us with their newest 5V/6A Power Supply that will keep your ODROID-XU4 (and any plugged USB powered disks) working perfectly.

With EU - Asia - US adapters, you can use the correct cable for your ODROID

TAKING A CRACK AT BREAKING WPA NETWORKS - PART I

BULLETPROOFING YOUR OWN SYSTEM

by Adrian Popa

In our previous articles, we've attacked WEP and WPS enabled networks, but now it's the time to attack the most secure wireless network technology out there: WPA encryption. As always, ask for the network owner's consent before attempting to break their network to save you from legal trouble afterwards. Better to be safe than sorry when testing things out!

WPA fundamentals

There are two flavors of WPA encryption: WPA1 uses TKIP (which is like a beefier WEP encryption but these days is considered out of date and deprecated) and WPA2 uses AES-CCMP. We will focus mostly on WPA2, but the techniques we show in this guide can be used with WPA1 as well. From a key management point of view, there are two types of WPA networks:

- WPA-PSK (Pre-Shared Key) - In this scenario, the same network key is known to all the network users and any user can see (decrypt) the neighbor's traffic with this key. This is generally used for home networks like the one you're probably using with any off-the-shelf router.
- WPA-Enterprise - This uses a RADIUS server to authenticate clients either with usernames and passwords or via individual certificates on each connected device. The advantage here is that when an employee leaves the company, his account is disabled on the server instead of having to reconfigure all access points and all WiFi clients. An added bonus is that an authenticated client can't sniff/decrypt the other clients' traffic because each client has a different key, separating each device from attack isolating compromised devices.

WPA-Personal encryption can be broken by brute-forcing the four-way handshake, but to break WPA-Enterprise, you need to set up a fake AP and set up a special RADIUS server to get the user's credentials (more details on that at http://bit.ly/2adge8d). We will be focusing on WPA-Personal from now on.

The four-way handshake

The PSK (Pre-Shared Key) or the username and password combination (with WPA-Enterprise networks) are not used to actually encrypt data in WPA networks. They are only used to authenticate to the network. After authentication, the network devices generate temporary encryption keys that are used to encrypt the actual data.

The WPA authentication mechanism involves the exchange of four EAPOL messages in order to set up the encryption keys for your session. The input information that both the access point and the mobile device need in this process include:

- Pairwise Master Key (PMK) - For WPA-Personal this is computed by concatenating the network SSID and your plaintext passphrase and running it through a SHA1 algorithm 4096 times as defined in the PBKDF2 function in RFC2898 http://www.ietf.org/rfc/rfc2898.txt .
- ANonce - A 256 bit pseudo-random number generated by the access point (Authenticator)
- SNonce - A 256 bit pseudo-random number generated by the mobile device (Supplicant)
- AA - The MAC address of the access point (Authenticator Address)
- SA - The MAC address of the mobile device (Supplicant Address)

After a device finishes the Open Network association process, both the access point (Authenticator) and the mobile de-

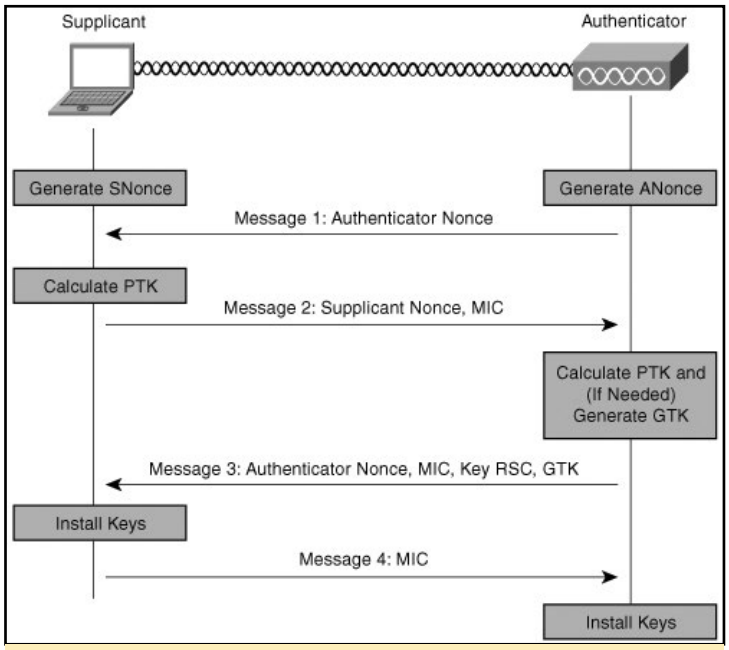


Figure 1 - A diagram of the four-way handshake in WPA encryption

vice (Supplicant) generate the temporary nonces. These are the ANonce and SNonce respectively.

During the handshake process, the access point goes first and sends Message 1 containing the Authenticator Nonce. The mobile device uses the ANonce and combines it with the known Pairwise Master Key (PMK) and calculates a Pairwise Transient Key (PTK). The PTK is used to encrypt unicast data and also to calculate Message Integrity Codes (MIC) and is 256 bits in length.

The mobile device then sends out Message 2 in the handshake process, containing its SNonce, and the message is protected with a MIC which is calculated from the PTK derived from Message 1. The access point can calculate its PTK and check the validity of Message 2. If the MIC is valid, it means that the Supplicant knows the same password and the process continues.

Next, the access point generates keys for multicast traffic that are shared between all authenticated hosts (GTK - Group Temporal Key), and it sends these keys in Message 3 of the handshake process. Note that Message 3 contains a MIC calculated by the access point (derived from SNonce and PMK) which is different in value from the MIC calculated by the supplicant. This allows the mobile device to authenticate the access point as well. The client confirms and sends back the last handshake message as an acknowledgement of this MIC and starts using the PTK and GTK for data transfer. More details about the four-way handshake process are available from this YouTube video at http://bit.ly/2aojWdD.

Taking a crack at WPA

Now that we know how authentication works, let's see what we need to do in order to crack WPA networks. First of all, the network password is only used in the authentication process. Subsequent authentications will generate different temporary keys which are used to encrypt the data. So, getting one of the keys will only allow you to decrypt part of the data until the client reauthenticates or the key is recalculated.

The problem is, you can't actually get the PTK because it's not being sent in any packet. It is just known to the end points. You can recalculate it to decrypt traffic only if you know the input information needed to authenticate to the network. An attacker listening in could easily get the Authenticator MAC address and the Supplicant MAC address because they are sent in the clear, but it needs to capture a four-way handshake in order to get the ANonce and SNonce. Even with this information, the attacker lacks the PMK, which is known only to the legitimate clients and access point.

The problem can be reduced to:

- Capturing a full four-way handshake between a legitimate client and the target access point
- Using a brute force method to calculate the PMK that combined with the ANonce can produce the same MIC the client sends in Message 2.

Capturing the handshake

In order to capture a four-way handshake you will need a network interface in monitor mode and airodump-ng:

```
$ sudo airmon-ng start wlan0
$ sudo iw dev wlan0 del
$ sudo sudo airodump-ng --channel 11 --write nasa
--output-format pcap --bssid BC:EE:7B:8F:6C:B2 --ig-
nore-negative-one mon0
```

The airmon-ng command puts your wireless interface in monitor mode, while the iw command deletes the managed interface (otherwise I wasn't capturing much traffic). Airodump-ng takes the channel as parameter, the BSSID of the target access-point, and writes a packet capture in pcap format that is prefixed with the string "nasa" in your current directory. If you haven't figured it out by now, we're still trying to hack into NASA to get our Haxt0r license!

Naturally, this means that you have to wait and listen until a client connects, but if you notice that a client is already con-

Figure 2 - Airodump-ng is kind enough to show you the last network it captured a handshake from

```
CH 11 ][ Elapsed: 40 s ][ 2016-06-04 19:03 ][ WPA handshake: BC:EE:7B:8F:6C:B2

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

BC:EE:7B:8F:6C:B2  -65  38      408      162   42  11  54e  WPA2 CCMP   PSK  NASA-HQ-WPA

BSSID              STATION            PWR   Rate    Lost    Frames  Probe
BC:EE:7B:8F:6C:B2  88:30:8A:3F:44:B7  -44    0e- 0e  1654      342  NASA-HQ-WPA
```

nected, you can run a deauthentication attack to kick it off the network and force it to reauthenticate:

```
$ sudo aireplay-ng -0 3 -a BC:EE:7B:8F:6C:B2 -c
88:30:8A:3F:44:B7 mon0
```

You can find more details about it in the Injection article from Odroid Magazine issue May 2016 (http://bit.ly/2azqi7U).

Aircrack will save the handshake inside a pcap file, such as the one in Figure 3 (http://bit.ly/2as3uaL). Next, we can use cracking tools such as aircrack-ng to look for the password.

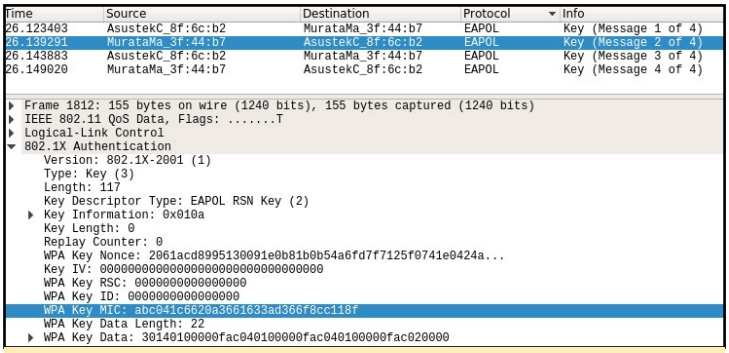


Figure 3 - The 4 way handshake

Note that I tried to capture handshakes with all of the different HardKernel WiFi modules, but I was unable to capture any handshake packets. I tried different positions, different kernels, different channels and different devices; I must have re-authenticated my mobile client a hundred times, but the capture showed no EAPOL packets. I also tried with my laptop's internal WiFi (Intel 6205/iwlwifi), and it was able to capture the handshake on the first attempt! This means that either there's a hardware limitation with HardKernel's WiFi modules, such as the antenna, or there are driver issues that prevents me from capturing handshakes. If you manage to use them for this purpose, please leave a comment in the support thread.

Brute force attacks

But here is the problem: even with a handshake captured, you need to look through all combinations of possible passwords until you find the correct one, because the MIC is not reversible. Maybe when HardKernel will release their Quantum Computer add-on board things might improve, as this really tests the limits of conventional computing!

How many combinations are there? Time for some math: the PSK is an ASCII string between 8 and 63 characters long. A US keyboard has 95 printable characters (10 digits, 26 lowercase letters, 26 uppercase letter and 33 special characters such as punctuation) so this means that the total address space of all PSK possible passwords is really big (see Figure 4). It's so big that for each atom in the known universe you'd have 1044 PSKs. Exponential growth has a way of getting out of hand

$$\sum_{n=8}^{63} 95^n =$$
39 919 297 033 102 270 412 781 965 613 433 199 719 545 223 215 933 635 382 877 ⋱
568 640 629 314 237 842 806 568 803 331 259 120 261 378 523 130 762 379 661 ⋱
850 000 000

Figure 4 - Number of PSK possibilities

really quickly!

Figure 5 shows a graph with the number of combinations for various string lengths and types of characters. The number of combinations grows exponentially with the length of the string (the graph's y axis is logarithmic, making the graph look linear). The allowed characters in the string are represented in regular expression syntax.

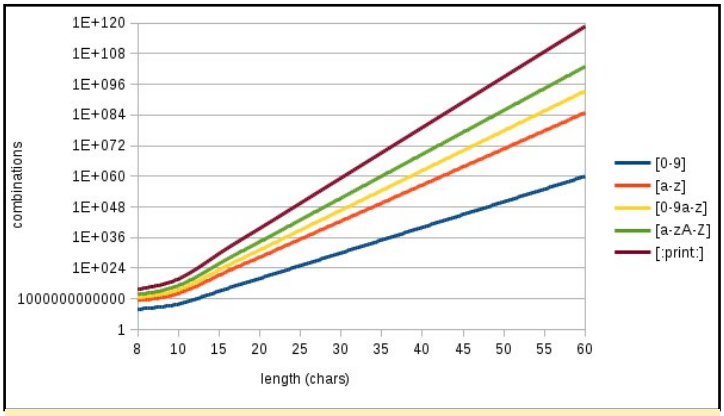


Figure 5 - Number of combinations vs string length

Are you discouraged yet? If you know the pattern of the PSK or if you can limit the characters used and their number, such as someone's birthday in ISO 8601 format, then you can try the brute force approach.

A tool which is designed for brute-force attacks and can benefit from Open-CL GPU acceleration is oclHashcat. Although it's a very powerful tool and allows you to set a password pattern (e.g. letter + letter + capital + number), it was written only for Intel architecture, so we can't use it on our Odroids - bummer.

But we can use "crunch" to generate the list of combinations and aircrack-ng to crack via the CPU. You should keep in mind that the total list of possible combinations will not be stored because all 8 character lowercase passwords, for example, take up almost 2TB of data. Crunch is very flexible and allows you to specify the list of characters you want to use and also their position in the string. Refer to Crunch's main SourceForge page for examples.

```
$ sudo apt-get install crunch
```

If you're on Ubuntu 14.04, you can install Crunch from github instead:

```
$ git clone git://git.code.sf.net/p/crunch-wordlist/\
```

```
code crunch
$ cd crunch
$ make
$ sudo make install
```

You can download here a capture file from http://bit.ly/2as3uaL for the network with SSID NASA-HQ-WPA that has an 8-character lowercase password. I'm going to let you find the password yourself, so please don't give it away on the forums!

To iterate through all possibilities with crunch and aircrack-ng you can do this:

```
$ crunch 8 8 + -t @@@@@@@@
| aircrack-ng -w - -b
BC:EE:7B:8F:6C:B2 nasa-aaaazzzz-
handshake.pcap
```

The syntax states that Crunch will generate strings of 8 characters that have a pattern of lowercase letters. This list is piped into aircrack-ng, which is asked to try every combination for the network key. Aircrack will start as many threads as it can and it will tax your CPU, so expect your ODROID to get really hot.

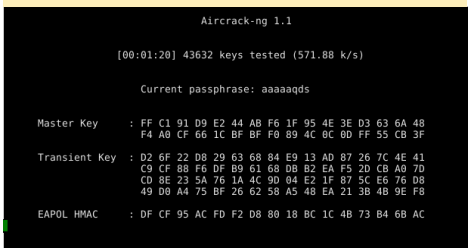
To get some OpenCL support for bruteforce cracking, we can use Pyrit:

```
$ sudo apt-get install pyrit
pyrit-opencl
$ pyrit list_cores
$ crunch 8 8 + -t @@@@@@@@ | py-
rit -r nasa-aaaazzzz-handshake.
pcap -b BC:EE:7B:8F:6C:B2 -i -
attack_passthrough
```

Performance

Although all ODROIDs have GPUs, the Mali 450 on the ODROID-C1 and ODROID-C2 does not support OpenCL, so the ODROID-C1 can only do CPU cracking, which is slow (~300 PMK/s). The C2 can also crack hashes on the CPU, and can achieve a rate of ~555 PMK/s, but it also has a dedicated crypto unit in the CPU which can offload some of the computations. Pyrit seems to use openssl as a backend to do the heavy lifting and openssl should support the crypto unit via the cryptodev kernel extension. Unfortunately, neither the stock kernel nor the stock version of OpenSSL come with cryptodev support, so further experimenting is necessary, but preliminary results look promising (http://bit.ly/2adiMTB).

Figure 6 - Crunch + aircrack-ng in action



The XU4 has a Mali T628 GPU, which supports OpenCL 1.1, so it can be used to crack hashes as well. For OpenCL support, you need to set up the environment by running the following commands:

```
$ sudo mkdir -p /etc/OpenCL/ven-
dors
$ sudo echo /usr/lib/arm-linux-
gnueabihf/libOpenCL.so > /etc/
OpenCL/vendors/mali.icd
```

This will enable the XU4 to use 2 Mali cores instead of 2 little cores with OpenCL. Note that performance-wise, you will see one Mali core have double the performance of the other since The Mali T628 has its 6 processing cores split 4/2 as far as OpenCL is concerned.

The XU4 can do CPU-only cracking with a performance of around 750 PMK/s limited mostly by heat. The best performance I got was with the conservative governor and the big cores limited to 1.6GHz. The trick is not to let the big cores overheat, which throttles them. If you use the GPU, performance increases to 1238 PMK/s - almost doubling the total performance. Perhaps the blue cooler sold by Hardkernel (http://bit.ly/2aolJQ2) with a Noctua fan can do an even better job.

Other systems I tested were my work PC (Intel G3220 @2x3.00GHz), my work laptop (Intel i7-3612QM @ 8x2.10GHz), a friend's gaming rig (Intel i7-4790k @ 8x4GHz) and an ESX server (Intel Xeon X5570 @ 32x3GHz). Figure 7 illustrates the performance values. However, even the top CPU cracker is blown out of the water by a high end gaming GPU (e.g. Nvidia GTX 980) which can do about 200000 PMK/s! You can find expected values for GPU cracking at http://bit.ly/26v53e7. As you can see, for serious cracking, or bitcoin mining, the ODROID devices are out of their league for brute-force attacks, but can be effective for dictionary attacks.

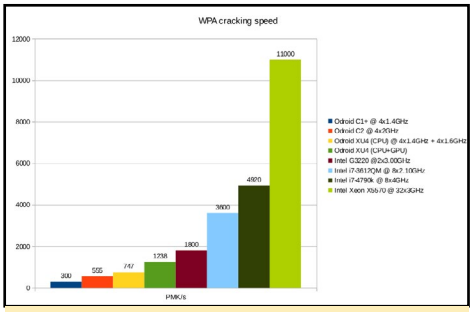


Figure 7 - Cracking performance by device: GPUs are off the scale!

Real attackers who are not afraid to invest a little money in cracking a password can employ the help of Amazon's server farms and use Amazon's EC2 clouds to do GPU cracking and pay only for the time they use too. More details about industrial cracking is available at http://bit.ly/1H4VYtH. As always, feel free to share your own best practices, ask questions, or share problems in the support thread at http://bit.ly/2azoM5N.

RECALBOX FOR ODROID-XU4

THE ULTIMATE MULTIMEDIA AND GAMING SYSTEM

by Nicolas Adenis-Lamarre

Recalbox is a lightweight Linux distribution that allows you to re-play a variety of video-game consoles and platforms when a computer such as the ODROID-XU4 is attached to a TV.

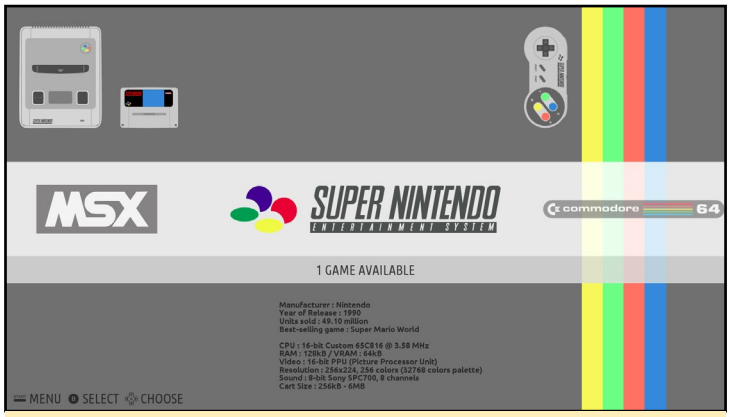


Figure 1 - Recalbox main screen

It includes the media center software Kodi and console emulators like Retroarch, Reicast, Mupen64 and even the streaming gaming system Moonlight. Refer to http://bit.ly/2a8jAaT for a complete list of compatible systems.

The system is plug and play and user friendly and hence attractive to any gaming enthusiast. You just attach your supported joysticks, add your own ROMs and videos and ask your friends to join you. It supports up to 5 players.

Features

On the media center side, Kodi 16.1 comes already configured. You can control it with your joystick (bluetooth or USB), your TV-remote or a device compatible with the LIRC software library. For starters, included by default, are some Kodi repositories, as well as some applications like Youtube and FilmOn. They help you to directly watch channels such as BBC, as shown in Figure 2.

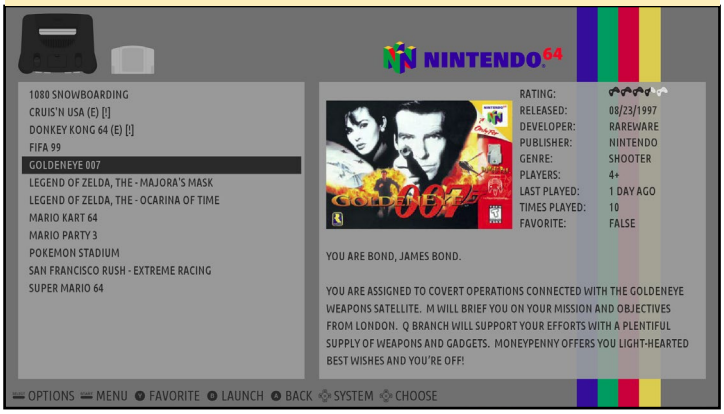
On the gaming side, about 50 systems are available for play.

Figure 2 - Streaming with the BBC Addon

The Hardkernel system of choice for Recalbox is the powerful and capable ODROID-XU4. It makes Nintendo 64, Sega DreamCast and Sony PSP games come alive, with very good performance.

The games must be copied onto a compatible microSD card directly from your computer or from a network share (Samba). An external USB stick or a HDD/SSD drive can be used as well. You can even use a NAS via NFS or samba-share if you wish to use a larger storage area. In the future, it will even be possible to use cloud storage such as DropBox or Google Drive

Figure 3 - Browsing a Nintendo game library



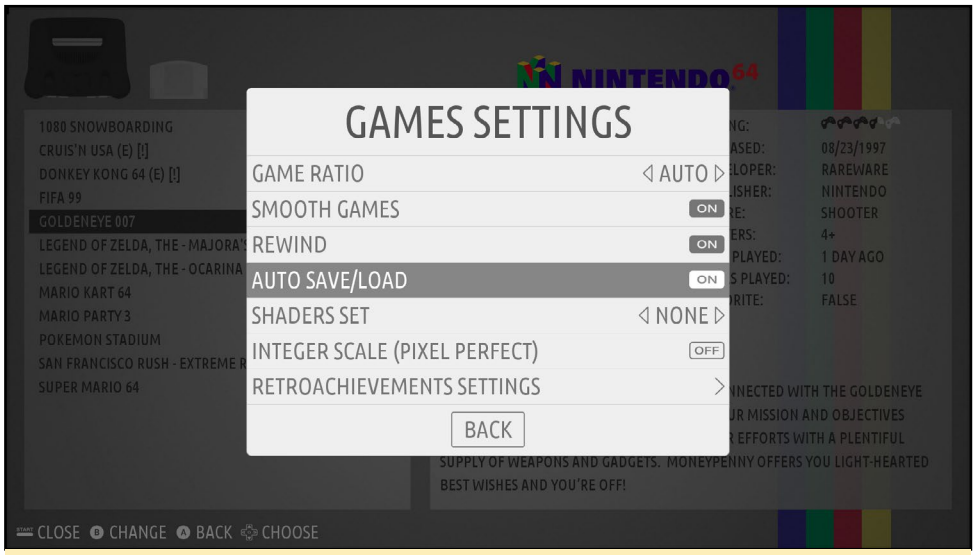


Figure 4 - The Recalbox Games Settings menu

to save all or part of the user's game data. Other features include shaders, auto-saves, retro-achievements and favorites, to name a few, as shown in Figure 4.

Implementation

The Recalbox is built using the Buildroot tool, which makes it possible to create a Linux image from scratch. Contrary to standard Linux distribution based systems, this system is more robust. For example, the root filesystem is read-only. So when using say, an SD card-based system, a power outage will not brick the device. The system is kept lightweight, with no Xorg server or other non-essential Linux features. This allows the system to boot very quickly.

The main interface is based on a custom version of EmulationStation. It lists games, metadata and game jackets. EmulationStation is the configuration user interface to configure the language, system updates, network settings, retro-achievements account and the joystick. On the emulation side, Retroarch makes integration of a new system easier. However, an appropriate emulator needs to be used to ensure good performance.

The ODROID-XU4 had the power button, Mali GPU and the CEC-specific modules already implemented. The more complex parts needing additional work, including patching SDL2, configuring RetroArch, and Kodi framebuffer patches.

Community

The RecalBox community is very knowledgeable and helpful. Support is available through their forums, irc, wiki, Facebook and Twitter outlets. My thanks go to this ODROID community and others including the Buildroot and Lakka communities for making it possible for me to make Recalbox work on the ODROID-XU4.

References

www.recalbox.com

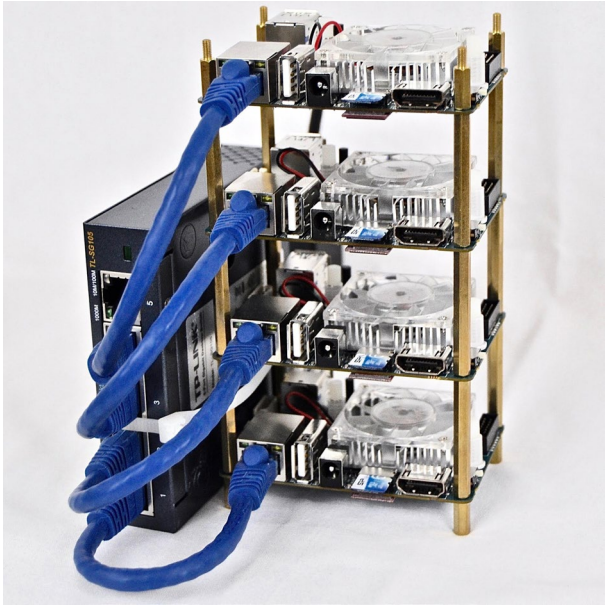
https://kodi.tv/about/

http://bit.ly/2a8jAaT

ODROID-XU4 CLUSTER

THE HOLY GRAIL OF CORE PER WATT vs PERFORMANCE vs PRICE

by Michael Kamprath



In the past few years, the topics of big data and data science have grown into mainstream prominence across countless industries. No longer are high tech companies in Silicon Valley the sole purveyors of topics like Hadoop, logistic regression, and machine learning. Being familiar with big data technologies is becoming an increasingly necessary requirement for tech jobs everywhere. Unfortunately, getting real, hands-on experience with big data technologies typically means having access to an expensive computer cluster to run your queries. However, the recent single board computer revolution has made true distributed computing accessible for personal use and education for tasks such as these and more.

I have worked in the big data space for eight years. While I have had access to a cluster to crunch petabytes of data for some time, I have never had the opportunity to design and build a cluster of my own. I decided to build a small cluster primarily to become more familiar with the underlying setup and operations of big data software and an underlying cluster. My price goal was to build a four-node cluster for under USD$600. I also wanted build a cluster powerful enough to be reasonably able to process data on the 10s of gigabyte scale in size.

Key elements of consideration when selecting the cluster technology is data storage and I/O, networking performance, CPU cores, and available RAM. Fortunately, Hardkernel makes a single board computer that excels in these spec needs: the ODROID-XU4. With a 2GHz Samsung Exynos 5422 8-core processor, onboard Gigabit ethernet, multiple USB 3.0 ports, 2 GB of RAM, and availability of high performance data storage with both eMMC drives and UHS-1 microSD cards, the XU4 is a formidable single board computer for a relatively low cost.

With the node hardware selected, our first task is to design the cluster topology, or how the nodes will be connected to each other. Several things influence this, most notably the type of distributed computing you expect to do. Distributed computing paradigms can be categorized roughly as either big CPU or big data. For this project, we are focusing on the big data use case, specifically for data analysis. The most common big data paradigm in use today for data analysis is mapreduce, which is implemented famously by both Apache Hadoop and Apache Spark, both very popular data warehousing technologies in use by many of the big tech companies out there.

In most commercial scale MapReduce clusters, the general cluster topology has any number of edge nodes that a user logs into to use the cluster, one or more head nodes which are used by the cluster to coordinate both compute activity and data storage, and any number of slave nodes which are used for compute tasks or data storage or both (see Figure 1). Think of it like dividing a large project between multiple people to improve everyone's speed: a director issues the project request (the edge node) with several managers coordinating what to do (the head nodes), and employees taking those tasks and combining their work (the slave nodes) into a final solution for the director.

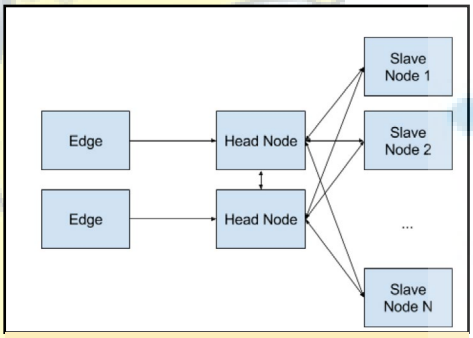


Figure 1 - Typical MapReduce Cluster Topology

For our XU4 cluster, we are going to combine the concept of an edge node and a head node into one master node, and then link slaves to the master node. This means the master node will be the node users log into to use the cluster and the node that coordinates the slaves. This also implies that the cluster's node-to-node communication would

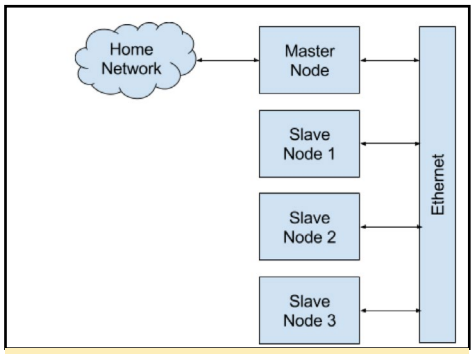


Figure 2 - ODROID-XU4 Cluster Topology

occur over a private network, while the master node needs to have connection to the outside network. Given that, the XU4's networking design for a four node cluster would need to resemble the one shown in Figure 2.

This topology requires the master node to be able to connect to two separate networks. However, the XU4 has only one ethernet port. A second network connection will need to be added to the master node with a USB 3 ethernet dongle.

The XU4 offers two storage options: an eMMC drive and a microSD card. Both have their pros and cons. The eMMC drive is extremely fast, while the microSD card cost per gigabyte is very affordable, but slower than the eMMC drive. The good news is that a UHS-1 microSD card's read and write performance can be on par with spinning hard drives, which are typically used in large commercial clusters. This makes the microSD card a good option for bulk data storage. However, the speed of the eMMC drive is attractive for using as a boot drive from which software is executed. Given that, each node in our cluster will have both an eMMC drive for booting from and a microSD card for bulk data storage. I recommend getting at least a 16GB eMMC drive for the master node, since it will be where you, as a user, will work from, while money can be saved by getting the cheaper 8GB eMMC drives for the slave nodes. For data storage, find some fast 64GB or greater microSD cards for each of the nodes.

The final set of materials necessary for the project include a small Ethernet switch for the cluster's internal network, a number of 6 inch Ethernet cables, and PCB standoffs to stack the XU4s together. I also picked up a serial UART for the XU4 in case I needed to connect to a device directly to sort out any issues, although I never needed it. One item which I did not purchase that would be nice to have in retrospect was a single power supply that could provide 5V power at 4 amps simultaneously to all the nodes, rather than a messy and inefficient collection of wall adapters plugged into a power strip. That will be a future improvement to the project.

Once all the materials are collected and the cluster is constructed, our first task is to configure the operating system and networking on all nodes. I chose to go with ODROID's current Ubuntu 15.10 distribution for the XU4. I flashed this OS onto each of the eMMC modules, and then one-by-one booted each device without the additional microSD card (which will be used for later storage after provisioning) and while directly connected to my home network. This allowed me to directly SSH into the device after the first boot. After the device booted, I found the IP address each XU4 grabbed from my home's DHCP server and logged in. The default user account is "odroid" with a password of "odroid". After connecting, I installed the ODROID Utility to further configure the OS. This can be done by directly downloading the utility from Github:

```
$ sudo -s
$ wget -O /usr/local/bin/odroid-utility.sh \
  https://raw.githubusercontent.com/\
  mdrjr/odroid-utility/master/odroid-utility.sh
$ chmod +x /usr/local/bin/odroid-utility.sh
$ odroid-utility.sh
```

The three tasks the ODROID Utility is used to accomplish is to name the node, disable Xorg, and maximize the partition size of the eMMC drive. I named the master node master, and the other three: slave1, slave2, and slave3.

The master node needs to be configured further to use the USB 3 ethernet dongle as it's external network. To configure the master node for getting its external internet connection from the network attached to USB dongle, you will need to create a file named "eth1" in the /etc/network/interfaces.d/ directory with the following contents (assuming that network has a DHCP server):

```
auto eth1
iface eth1 inet dhcp
```

Similarly, to have the onboard ethernet be used for the internal cluster network, a file named eth0 needs to be created in the same folder indicating a static IP address:

```
auto eth0
iface eth0 inet static
        address 10.10.10.1
        netmask 255.255.255.0
        network 10.10.10.0
        broadcast 10.10.10.255
```

A DHCP server needs to be set up on the master node in order to provide an IP address to the slave nodes on the internal network, and the master node will need to provide NAT services between the external and internal networks. Furthermore, all nodes will need their /etc/hosts file edited to allow mnemonic addressing of nodes by their name without needing a DNS service. Detailed instructions for accomplishing these tasks can be found at my blog at http://bit.ly/2aJdAmi.

Once the nodes are configured for the desired networking design, the nodes can be shut down and disconnected from the home networking. The nodes' onboard Ethernet should be connected to

the internal network's Ethernet switch, and your home network should connect to the master node's USB 3 Ethernet dongle.

Before restarting each node, format the microSD cards with an ext4 file system, and attach one to each node. Boot up all the devices. You should be able to SSH into the master node, and from there you can SSH into each slave. Your final setup task is to configure the /etc/fstab file on each device such that the microSD card is mounted to a /data mount point. To do this, you need to find the UUID of the microSD card's volume after mountain it for the first time with the blkid command, then adding a line to the /etc/fstab file that looks like:

```
UUID=c1f7210a-293a-423e-9bde-
1eba3bcc9c34 /data ext4 defaults
0 0
```

Replacing your microSD card's UUID with the one listed above, which is also detailed on my blog. Once these steps are completed, you will have a fully configured cluster that is ready to have big data software such as Hadoop installed. Installing Hadoop is a fairly involved process, and I will cover that in a future article. For now, we have successfully provisioned an XU4 cluster that can be used for any sort of complex data processing. Further information about this ODROID-XU4 cluster can be found at http://bit.ly/2aJdAmi.

XPOSED FRAMEWORK

TAME YOUR ANDROID UPGRADES AND HANDLE SYSTEM LEVEL CHANGES

by Jörg Wolff

Consider the issue of making system level changes to you Android operating system. This can be achieved by the laborious process of creating custom ROMs, installing them, re-working upgrades and re-installing them. This is unsustainable.

Frameworks such as the Xposed framework can come to your rescue, in many cases, as long as the original code/behavior was not changed extensively. Given root access, it allows you to make system level changes, without installing custom ROMs. Reversal of the changes can be done by simply deactivating the module and rebooting.

Installation

An ODROID-C2 is used in this example. To begin, open a terminal window. To a new working directory on the device, download the ARM/Lollipop version of the framework (xposed-v84-sdk22-arm.zip) from http://bit.ly/1VOp1wW. Unzip the framework file. From the same working directory, download the installer apk (XposedInstaller_3.0_alpha4.apk) from http://bit.ly/1GODrBg.

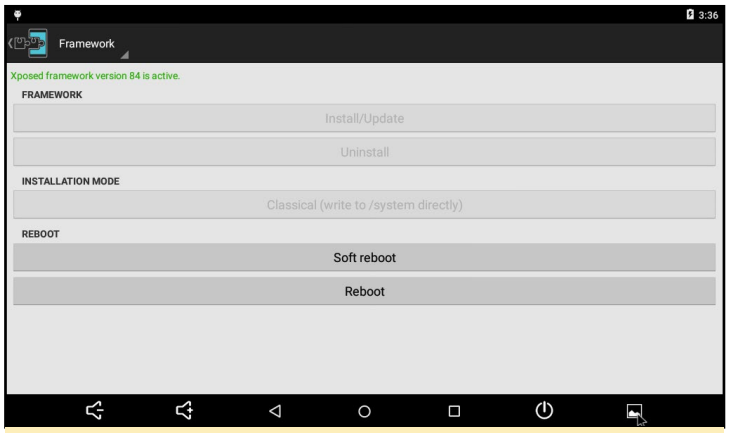
Run the following commands:

```
$ adb install XposedInstaller_3.0_alpha4.apk
$ adb shell
$ mount -o remount, rw /
$ mkdir tmp
$ cp -av sdcard/xposed-v84-sdk22-arm /tmp/
$ cd tmp/xposed-v84-sdk22-arm
$ cp -av META-INF/com/google/android/* .
$ chmod 755 flash-script.sh
$ ./flash-script.sh

...
******************************
Xposed framework installer zip
******************************
- Mounting /system and /vendor read-write
```

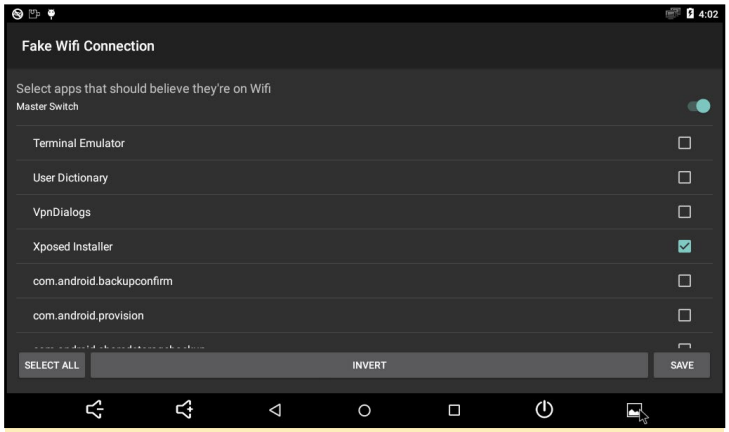
```
- Checking environment
  Xposed version: 84
- Placing files
- Done
```

The Xposed framework configuration screen is shown in Figure 1. Note that the Install/Update option is disabled. In my case, since I was on the wired LAN, I suspected that the lack of a Wi-Fi connection caused the disabling of that option. The Fake Wi-Fi Connection tool came to my rescue.



Xposed Framework UI

Download the Fake Wi-Fi apk from http://bit.ly/1rdfT7B, then sideload it using ADB tools. Start the Fake Wi-Fi Connection and select the Xposed Installer. After the installation has completed, you should be able to run the Install/Update option in Xposed.



Fake Wi-Fi setup

References

http://bit.ly/2ao423N

http://bit.ly/1hCaq32

http://bit.ly/1VOp1wW

http://bit.ly/1GODrBg

http://bit.ly/1rdfT7B

BROTHERS: A TALE OF TWO SONS

A FANTASTIC GAME FINALLY PORTED TO ANDROID

by Bruno Doiche

The opportunity to play a game that was vastly acclaimed as a masterpiece, and directed by an award-winning director does not present itself every day, so get yourself and your control ready to enjoy a game that was previously only available for console games.

With some of what I believe that was some of the most wonderful storytelling from the last generation of games, it reminds me so much of the PS2 classic, Shadow of Colossus. You are inserted straight into a quest where, although you cannot understand which language the brothers are speaking, you are immersed in the game world based on the small details of their interactions with the environment, characters and objects.

A great adventure awaits you and you will wish that it never ends

If you have brothers, you will get yourself identifying with the relationship between the older brother and the younger one. Seeing this loving contrast of responsibility and playfulness instills a sense of awe in me for the beautiful Nordic-inspired land of this game.

```
https://play.google.com/store/apps/details?id=com.
and.games505.brothers
```

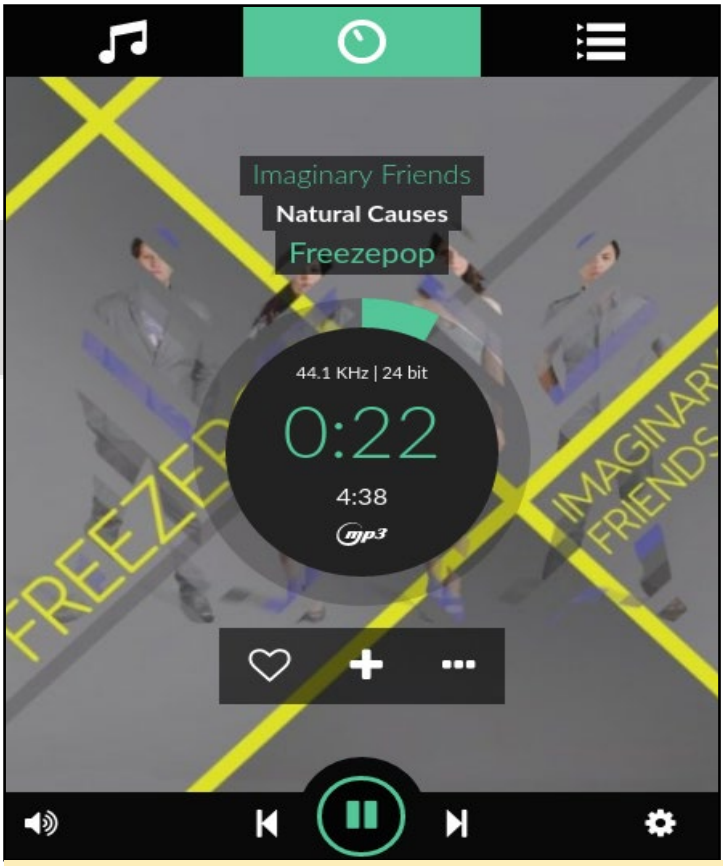
VOLUMIO 2.0

YOUR FAVORITE INTEGRATED MUSIC PLAYER JUST GOT BETTER

by @synportack24

If you haven't heard of Volumio, it's a great feature-packed OS that can turn your ODROID into a WiFi-controlled jukebox. If you are familiar with Volumio, you might be wondering why things have been rather quiet from them recently when it comes to updates.

Volumio 2 marks a significant step forward and is far from being a simple version number increment. The entire web interface has been rewritten in Node.js, and is accompanied by a reworked OS as well. The result of these changes is a more modular interface which makes writing plugins much easier. Another of the major changes you'll notice coming from version 1 is that it's fast! Almost every part of the using Volumio 2 seems quicker. It boots in seconds and there is nearly no lag when moving around the web UI. The OS is available for a range of ODROID platforms, including the C0, C1, C1+, C2, XU3, XU4, and X2. A link to the prebuilt binaries and to the project's source are available at the end of this article.



Volumio 2.0 was rewritten in Node.js and is much faster

Setup

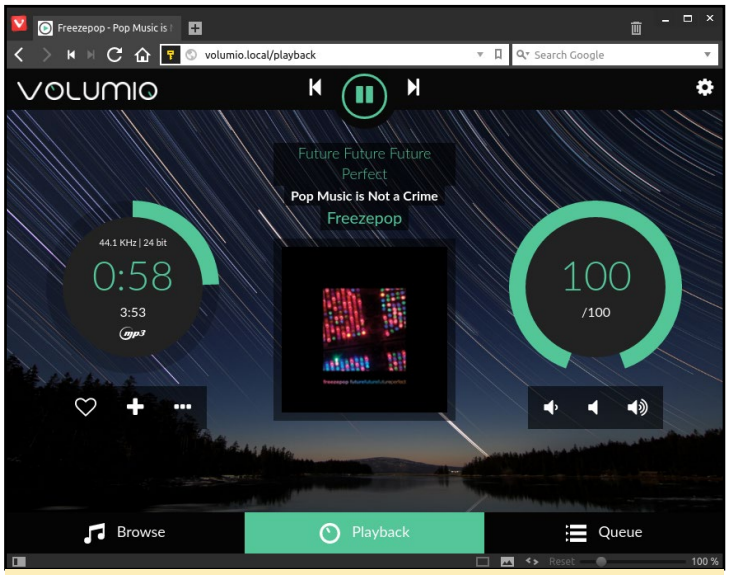
For the past month, I've been running Volumio 2 RC2 on a ODROID-C2 with a HiFi Shield and am eagerly waiting for their full release. The setup for Volumio 2 is extremely simple and takes just a few clicks to get things started. Simple flash the Volumio image to a micro SD card or eMMC module and connect your ODROID to your network via Ethernet. Note you have to use Ethernet at first, rather than WiFi, just to set things up. Open up a browser, and on your computer a type "volumio.local". Volumio 2 uses a protocol known as Bonjour, so if you have a browser that support it, you don't need to know the device's IP address. I had no problems using this address with an iOS and several browsers in Linux, but Android Marshmallow would only work if I entered the device's IP. Clicking on the "gear" settings icon in the upper right corner brings up a list of settings pages. To add a WiFi connection, click on the "Network" tab and scroll down to find a list of local networks. With the current version (RC2), I noticed that not all of my usb WiFi dongles were supported, but I had good luck using Hardkernel's WiFi module 0.

After the WiFi setting are finished, you can disconnect the Ethernet cable. The next step will likely be adding a NAS or some music source. Configuring a NAS is amazingly simple, since Volumio will automatically find and list all network and local storage devices. Once a device is added, it will search and add all music. If you have the HiFi shield, which I highly recommend, you can switch the audio output between the shield and HDMI on the "Playback Options" tab. That's all you need to do get started.

Review

While reading this review, keep in mind that I've been running a release candidate, so bugs and issues are likely to be corrected before the final release. Even over the last month or so, there has been tremendous progress in crossing things off their "todo" list. Updates can be simply applied via OTA (over the air) directly from the web interface.

I was unable to get my primary Samba share to mount and be scanned, but a quickly made share from my Debian laptop worked without a hitch. I'm certain that with a little more in-



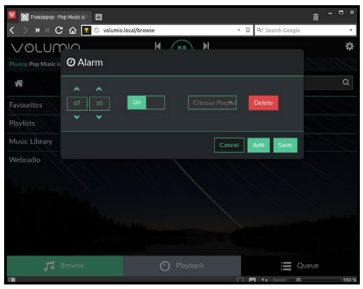
The new interface is easy to use and also includes an Android app

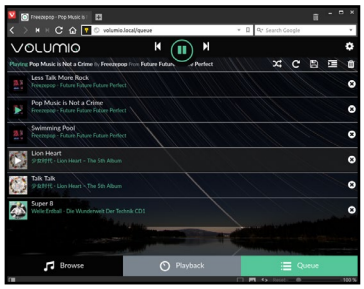
vestigation, I could have gotten it to mount properly. Beyond that, Volumio was able to play every format I have. Various type of bit rate MP3, M4As, FLAC, and OGG files all played wonderfully. Some of the larger FLAC files would need a second or so to buffer, but this was mitigated by changing some of the file buffer settings. From the main page, you can easily dig through your music queue, browse for more songs, or just look at what's playing. It's important that you have your music arranged in a logical layout, since Volumio simply uses your directory structure. The web interface scales very well if you're using a large screen or phone, and remains very fast even on older phones. There is a Volumio Android app available too, although the web interface is more than sufficient.

Overall, Volumio is a very feature packed jukebox that is extremely easy to setup. After using Volumio for about a month, I could see a huge potential. I'm eagerly awaiting the final version 2 release, and would recommend anyone looking for a network media player to check it out. For more information, or to download Volumio and try it for yourself, visit the following links:

References

Volumio 2.0: http://bit.ly/2a5TxxN
Image download page: http://bit.ly/2aciw17
Github page: http://bit.ly/2aoJUfq





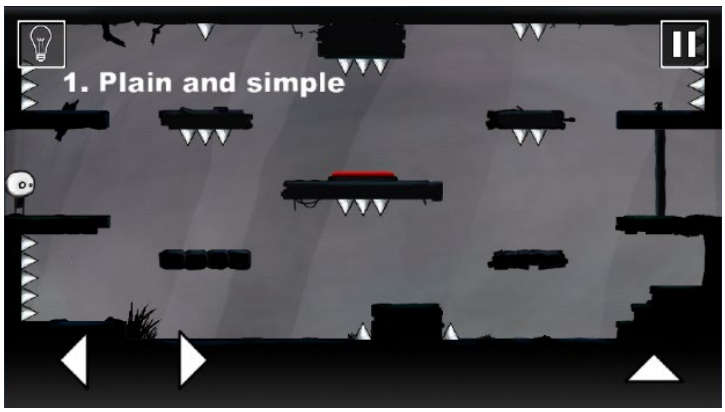
THAT LEVEL AGAIN

WHERE ALL LEVELS ARE THE SAME EXCEPT THE WAY TO WIN

by Bruno Doiche

Puzzle games often ask you to discover a pattern in order to solve them, and once you do so, you will never have a problem completing that game again the same way, right?

Well, That Level Again turns that assumption upside down, with 64 levels that each one have 64 different ways to solve it! This presents a challenge never before seen in a puzzle game.



Those who love to think outside the box are in for a treat, and for those that were just expecting a simple puzzle game, welcome to your hardest challenge.

https://play.google.com/store/apps/details?id=ru.iamtagir.game.android

MEET AN ODROIDIAN

JOSHUA SHERMAN, ASSISTANT EDITOR OF ODROID MAGAZINE

edited by Rob Roy

Josh enjoys photography as well as computing

In London, trying out Google Glass with a colleague

Please tell us a little about yourself.

I'm one of the volunteer editors with ODROID Magazine and a tinkerer at heart, always searching for a new project to give a run for its money. I'm a former writer for a tech publication called Digital Trends, and these days I work in the San Francisco Bay Area, though I'm originally from New York on the east coast of the United States. Although I love tearing apart gadgets, my formal education is in English from a liberal arts college back in New York, a far cry from some of the impressive doctoral candidates and engineers I see contribute to this awesome magazine. That doesn't stop me from breaking and (most of the time) eventually fixing computers of all shapes and sizes in all kinds of pet projects, including ODROIDs.

How did you get started with computers?

My dad gave me a computer when I was very young, though "very young" for me is a Pentium-era PC running Windows 98. Right away, I found a way to dual-boot it with Windows XP because I wanted the new features of XP but also wanted the DOS game support of Windows 98. Ever since then, I've slowly but surely become an avid fan of the computing industry, working as a journalist for mobile phones and staying interested in anything with a CPU inside, whether it sat on my desk, inside a home-made arcade emulator, or in my pocket. I've always been searching for ways to get the most out of the hardware I owned, and along the way often joined all sorts of forum communities, like ODROID's community, XDA-Developers, PPCGeeks, and others, to share my knowledge and learn from others too. This is what led to my writing guides on how to set up an RTL-SDR on an Android device, turn a computer into an arcade emulator, and of course contribute to ODROID magazine.

What attracted you to the ODROID platform?

It's so versatile as a hardware platform, and it attracts such an active, dedicated community of creators and developers. I found my start with the Raspberry Pi B+, but found its hardware limiting in what I truly wanted to do. At the time, it was arcade emulation, and the ODROID-XU4 opened far more doors to build out better emulator options and bootstrap more features that would otherwise be impossible on the latest Pi hardware, though greatly dependent on the hard work of developers across the Single Board Computing world. Overall, I enjoy the access to high-end hardware like the XU4, the dedicated community we have, and the direct support we see from Hardkernel on new products and operating systems.

How do you use your ODROIDs?

Right now I've got a handful ODROIDs (a C2, a C1, and two XU4s) sitting in boxes since I just moved. Once I get them out, I have some interesting plans to experiment with building a more advanced media center device, testing out a DVR on an ODROID, and perhaps even home automation. I also tried out some really cool x86 emulation with an XU4 recently, and I'm even considering a home-made firewall or server solution built from a few XU4s, if I ever get around to getting a few more! I also started thinking about a self-sustaining weather station project featuring an ODROID-C0, but have yet to acquire a lot of the parts necessary for such an endeavor.

Which ODROID is your favorite and why?

The ODROID-XU4 blew me away when I first bought it. It was just amazing to see how it could handle the Android OS like a hot knife through butter, with plenty of processing power to experiment with it as a full-blown desktop device. I'm always for using the biggest and most powerful hardware the Single Board Computing world has to offer.

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

I'm a big fan of integrated WiFi and Bluetooth for the next XU4 to help cut down on the number of USB ports spent on peripherals, especially when I like getting started with a keyboard and mouse for some experimental projects, or for my potential media server, but prefer to minimize the need for other cables or external USB hubs. Most importantly, I'm hoping that Hardkernel figures out a way to improve GPU and VPU performance, whether through kernels, hardware, or support for Tegra or other potential ARM-based graphics processing needs. I also hope the WeatherBoard gets an update to support external sensors, or some other way to separate the electronics from the environment for more robust outdoor monitoring.

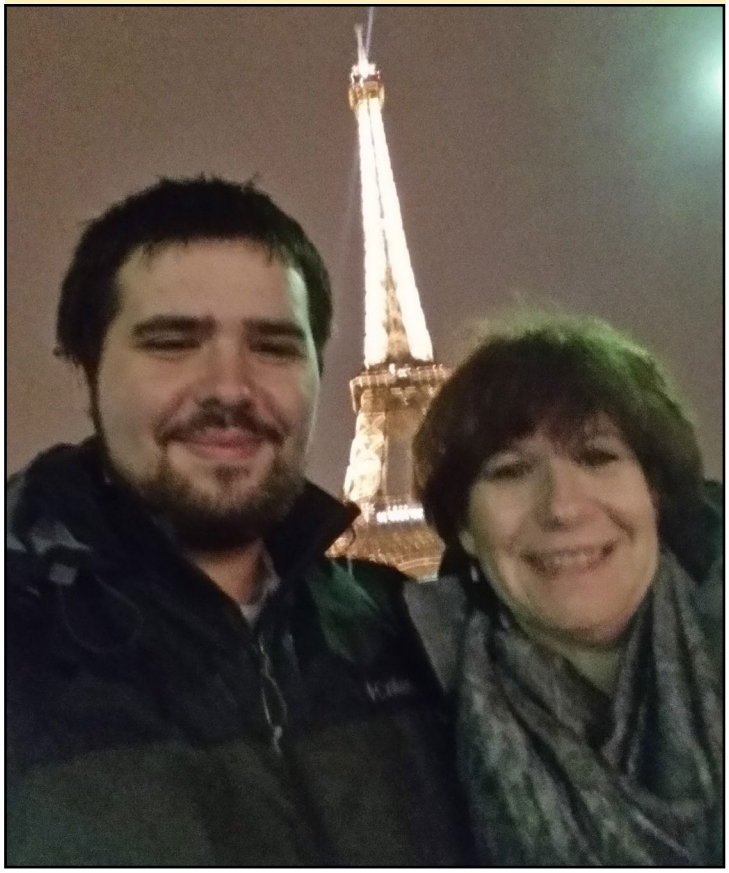
What hobbies and interests do you have apart from computers?

Computers small and large, including smartphones, SBCs, and desktops take up the bulk of my hobbies, but I enjoy photography with my Sony a6000, cycling, and writing as well. I also own a Linux VPS that I love to test new web apps with and just experiment with new private cloud software.

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

I'm still a total greenhorn on the programming side, but I definitely recommend trying some turn-key projects or experiments in the ODROID forums and elsewhere. They offer a taste into the world of working with the Linux Operating System, which includes editing shell scripts and creating your own shell scripts for whatever you may need. It's just a start, but I think that learning from others is always the best place to begin.

Josh and his mother in Paris, visiting the Eiffel Tower



With his family dog, a Miniature Pinscher rescue named Trixie