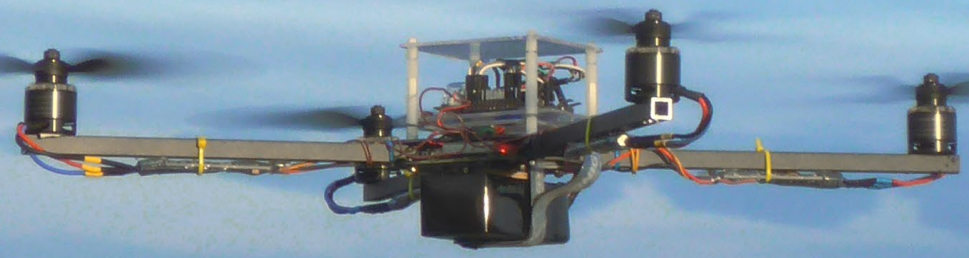
KitKat 4.4.2 on the ODROID-C1 • Android Gaming • Portable Cluster

ODROID Magazine

Year Two
Issue #15
Mar 2015

Age of Flight



Build and fly your own QuadCopter with autopilot using Navio+ software

Docker tips:
Get more from your containers

- Xbox 360 Controllers in Android
- Bench Oscilloscope with ODROID-C1
- Android Dev: Dissect and Modify APK Files
- Power your ODROID-C1 via USB

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 U3 devices 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

Have you ever wanted to fly one of those cool drones, but found them to be too expensive to afford? Our cover article shows you how to take the ODROID-C1 and transform it into an awesome flying machine, using parts readily available from an online electronics store. It even includes a camera, which can be viewed through a standard tablet, with a PS3 controller serving as the flight controls. You can also use Navio+, a popular autopilot software, to assist in navigating the QuadCopter to a specific location.

This issue also features lots of games, with Tobias showing us how to play some very unique NDS games that use innovative user interfaces to solve puzzles, along with several Android game reviews. Our series on Docker continues with some tips on using it with Ubuntu 14.04, and Nanik shows us how to dissect and reconstruct the Android APK. Venkat also presents his project on using the ODROID-C1 as a bench oscilloscope, and we showcase a portable ODROID cluster that includes some pre-built images to help you get started quickly with your own high performance computing projects.

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
Makers of the ODROID family of quad-core development boards and the world's first ARM big.LITTLE architecture based single board computer.
Join the ODROID community with members from over 135 countries, at http://forum.odroid.com, and explore the new technologies offered by Hardkernel at http://www.hardkernel.com.

HARDKERNEL

ODROID Magazine

Rob Roy, Chief Editor

I'm a computer programmer living and working 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 pre-built 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`.

Bo Lechnowsky, Editor

I am President of Respectech, Inc., a technology consultancy in Ukiah, CA, USA that I founded in 2001. From my background in electronics and computer programming, I manage a team of technologists, plus develop custom solutions for companies ranging from small businesses to worldwide corporations. ODROIDs are one of the weapons in my arsenal for tackling these projects. My favorite development languages are Rebol and Red, both of which run fabulously on ARM-based systems like the ODROID-U3. Regarding hobbies, if you need some, I'd be happy to give you some of mine as I have too many. That would help me to have more time to spend with my wonderful wife of 23 years and my four beautiful children.

Bruno Doiche, Senior Art Editor

Is taking Japanese lessons, but so far has only learned to eat a book, drink a newspaper, swim while cooking and a bunch of curses, to the disgust of his teacher. He also got a bunch of Android games that are not in this magazine edition (though the crossy road one made it!)

Nicole Scott, Art Editor

I'm a Digital Strategist and Transmedia Producer specializing in online optimization and inbound marketing strategies, social media directing, and media production for print, web, video, and film. Managing multiple accounts with agencies and filmmakers, from Analytics and Adwords to video editing and DVD authoring. I own an ODROID-U3 which I use to run a sandbox web server, live in the California Bay Area, and enjoy hiking, camping and playing music. Visit my web page at http://www.nicolecscott.com.

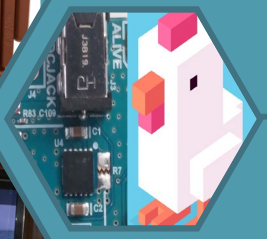
James LeFevour, Art Editor

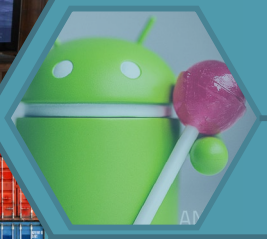
I am a Digital Media Specialist who is also enjoying freelance work in social network marketing and website administration. The more I learn about ODROID capabilities the more excited I am to try new things I'm learning about. Being a transplant to San Diego from the Midwest, I am still quite enamored 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.

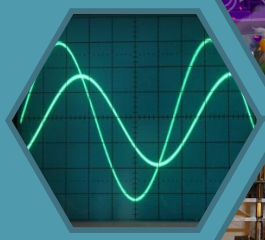
Manuel Adamuz, Spanish Editor

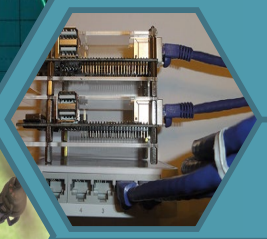
I am 31 years old and live in Seville, Spain, and was born in Granada. I am married to a wonderful woman and have a child. A few years ago I worked as a computer technician and programmer, but my current job is related to quality management and information technology: ISO 9001, ISO 27001, and ISO 20000. 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.

INDEX





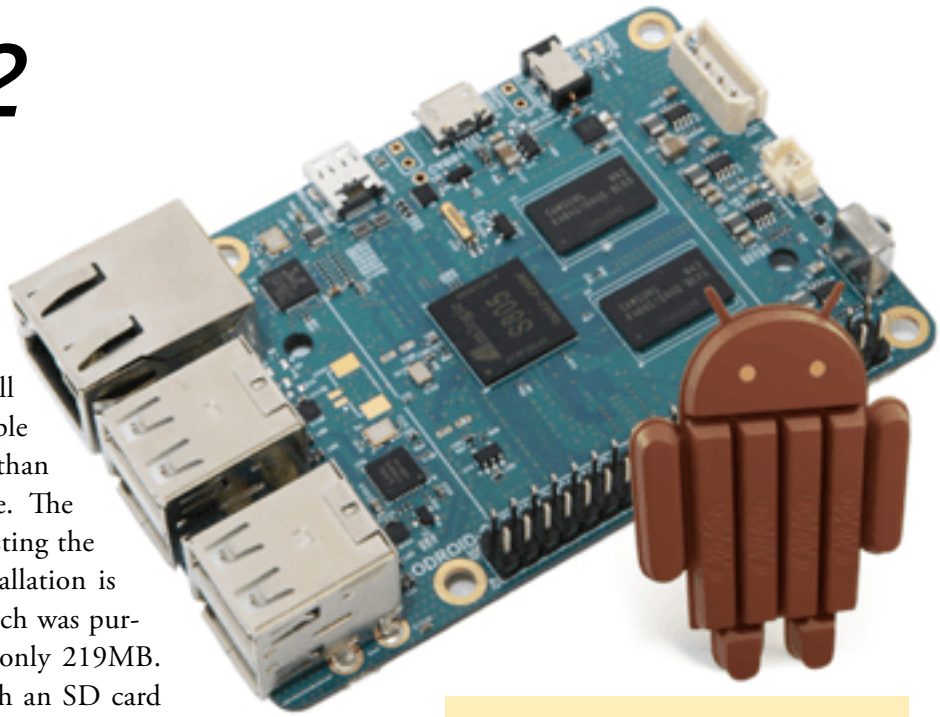




KITKAT 4.4.2

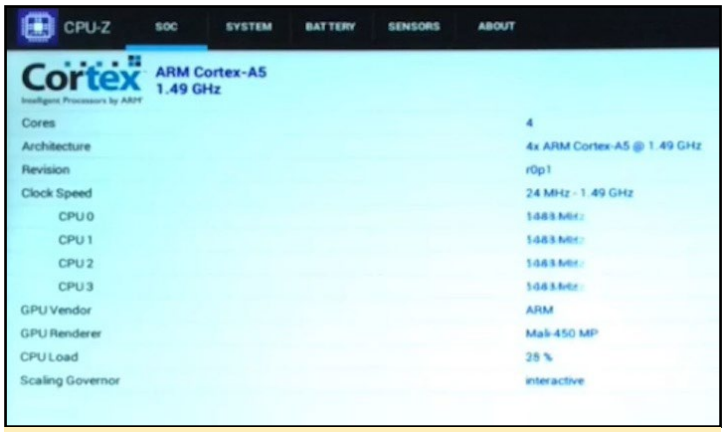
ODROID-C1 REVIEW

by Jose Cerrejon Glez



Want a stable Android KitKat to use as your main computer? Look no further!

In this review of the ODROID-C1, we will discuss whether it runs Android with multiple applications, games and emulators better than other competitor boards in the same price range. The first thing that I should mention is that I'm testing the Android Kit Kat 4.4.2, version 1.1. The installation is on an 8GB UHS-I SD card from Sandisk, which was purchased at the Hardkernel store, and takes up only 219MB. Everything that I had read about Android with an SD card was that it was very slow due to constant input/output access of the operating system, so it is preferable to use an eMMC module instead. However, I have found that the microSD card is perfectly adequate, and will will run very smoothly, as well as you would expect on any tablet. I'm sure with that the eMMC the operating system works faster, but an SD card is enough to have a good experience.



CPU-Z displaying the specifications of the ODROID-C1

Once inserted and booted, the system will resize the available partitions, including one vFAT partition which acts as an external SD card, where we can copy auxiliary files through a PC connected via USB. Another positive aspect is that with an 8GB card, we won't run out of space, because the operating system itself will take care of installing applications on another partition if the base system partition becomes full. I ran my tests at both 1280x800 and 1920x1080 resolutions.

The apps that I installed on the Android image for testing

Preinstalled applications

The following apps that come with the stock Hardkernel image are very useful:

ODROID Utility: The most important use of this tool is to set the screen resolution to fit your particular screen.

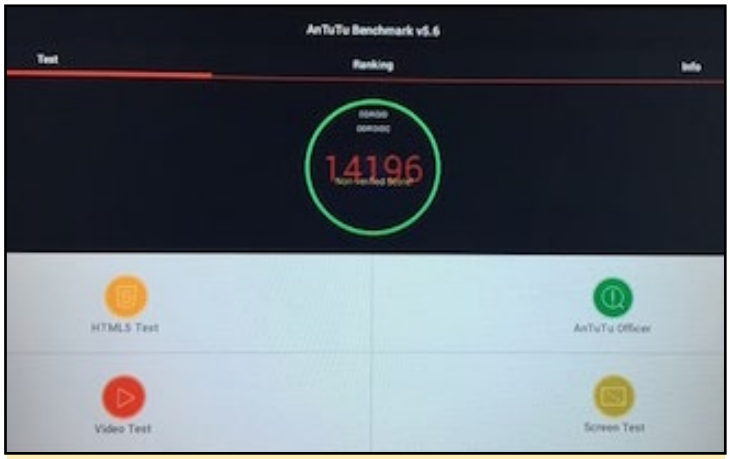
DicePlayer: An unpretentious video player which plays almost any kind of multimedia files without problem, except for 4K videos. I also tested videos with Kodi, and the ODROID-C1 performs the best of all the boards that I tested (Banana Pi and Raspberry Pi), and doesn't drop any frames like the other boards.

Recommended applications

Google Play: The store is not included by default, but it is very easy to install by downloading the installer from `http://bit.ly/1wHG45b`.

SuperSU: The OS comes rooted, but I needed to install the SuperSU application from Google Play in order to grant permissions to programs seeking access to certain files, including Kodi. Without applications like SuperSU, Kodi can't access hardware decoding, so this application is required.

Games: The only limitation that I found is in 3D games - not for performance, but because they are developed for use with touch screens and the mouse does not work. For example, I couldn't press a button to start playing with the mouse. Maybe there is a way, but I've not found a suitable solution yet that works.



AnTuTu Benchmark on the ODROID-CI achieved a score of 14196

RetroArch: I tried the SNES, GENESIS and MAME cores, all of which run flawlessly.

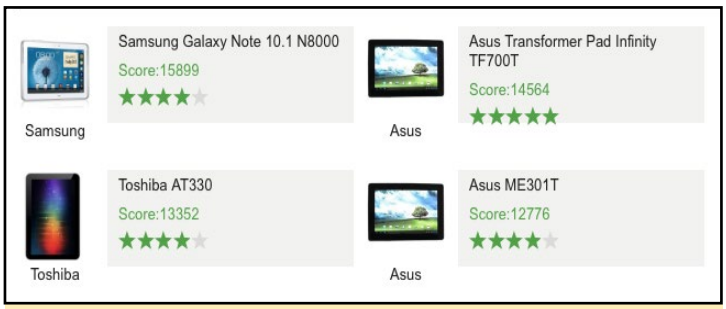
PPSSPP: Except games that use high 3D graphics like God of War, PPSSPP works very well. A particularly nice game is Kingdom Hearts, which runs and looks smooth despite its demanding 3D graphics.

Spectaculator (ZX Spectrum emulator): Runs perfectly.

Mupen64+ running at full speed on an ODROID-CI

Mupen64 + (Nintendo 64): Mario 64 is nearly perfect, both in sound and FPS. There are a few glitches in some shadows and textures, but hardly noticeable.

Kodi playing a 1080p video demo



AnTuTu Benchmark for other devices similar to the ODROID-CI

PPSSPP and the game Kingdom Hearts running on an ODROID-CI

Control Pad: I tried the XBOX 360 wired controller with all of the emulators, and it works out of the box.

Web Browser: I tested the default browser as well as Chrome. The navigation is fluid with no slowdowns. Mobile Youtube via the browser loses picture quality, but using the native Youtube application gives much better performance. Although Flash does not work, who needs it?

Youtube: I installed Youtube because it's a "must have" when my friends come to my house, and of course it works as it should. In addition, it loads videos very quickly.

Karaoke: I have not tried to connect a USB microphone, but by installing a Karoake application from Google Play, I could play any .kar files. It seems incredible that something so simple can not work at all with the Raspberry Pi.

Conclusion

I can say that Android behaves as it should and can work as your main operating system. It's a shame about the problem with 3D games that only support touch screens, but this is understandable. I'm very happy with the combination of ODROID C-1 and Android, and can connect to the TV in my living room, so that I can use it as an entertainment center. I've published a video at `http://bit.ly/18hxfqy` so that you can watch how some of the applications run. For questions and more information, please visit the original post at `http://misapuntesde.com/post.php?id=511`.

POWERING THE ODROID-C1 USING THE MICROUSB PORT

CELLPHONE CHARGER USAGE IS JUST A SOLDER AWAY

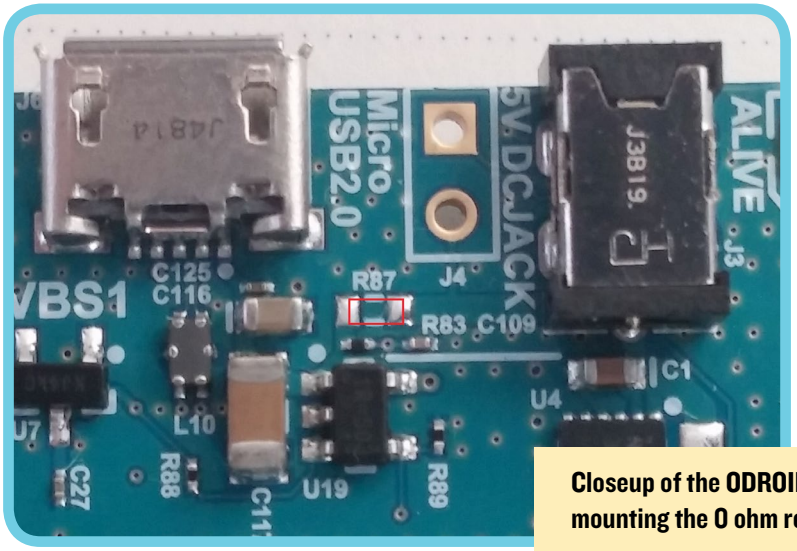
edited by Rob Roy

A common concern of new ODROID-C1 owners, especially those that are migrating from the Raspberry Pi platform, is that the device appears only to be able to be powered via the DC jack. Conversely, many people assume that the microUSB can provide power, and don't order the official power supply, and end up needing to resort to using inferior third-party power supplies, which causes issues when operating at a high CPU load.

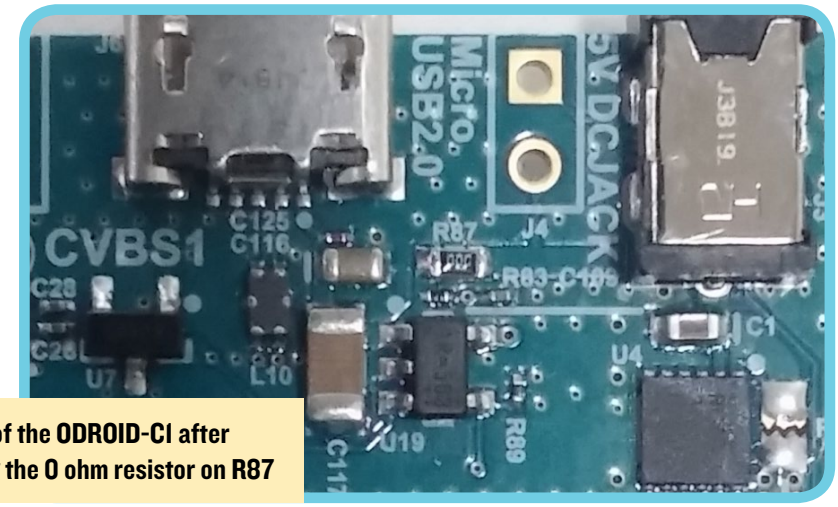
However, the ODROID-C1 is fully capable of being powered by the microUSB port, which just isn't enabled by default. A simple hardware bridge allows this option, which is detailed in the images below.

To perform the modification, you may select one of the following two methods:

- Bridge the two pads of R87 with soldering.
- Mount a 0-ohm resistor on R87 (type 1608).



Closeup of the ODROID-C1 before mounting the 0 ohm resistor on R87

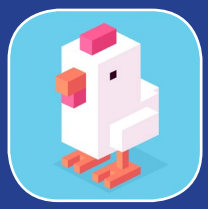


Closeup of the ODROID-C1 after mounting the 0 ohm resistor on R87

CROSSY ROAD

PART CLASSIC, PART REVAMP, ABSOLUTE FUN

by Bruno Doiche



The eldest among us may instantly get flashbacks to the early 80s when it was all the rage to play freeway on the Atari 2600 in order to get a chicken across the road. And without depending on any emulation (not that we don't enjoy that too), there is an incredibly simple and fun game that lets you help a cute polygonal chicken with crossing roads, rivers, and rails, and avoiding pretty much everything else along the way. It's a must have for any ODROID!

https://play.google.com/store/apps/details?id=com.yodo1.crossyroad&hl=en

A simple tapping and swiping game, that is easy to learn and addictive

With a bunch of unlockable other characters as rewards for achievements, you will wonder where the last 2 hours went

HOME THEATER PC

HOW THE ODROID-C1 MEASURES UP

by Douglas Roberts

Back in July of 2013, I put together a couple of fun Raspberry Pi projects: an NFS and Minidlna server (http://bit.ly/18HojM9), and an XBMC home entertainment system component (http://bit.ly/1ASuZl4)

Last month, I purchased a couple of ODROID-C1 units, which looked interesting because, for the same $35 as the Pi, you get a slightly smaller SBC with approximately 6X the power. Both the Pi and the C1 draw approximately 3 - 4 watts when idling. Here is a quick comparison:

Pi CPU: ARM 700MHz vs. C1: Quad-core ARM 1.5GHz
Pi GPU: 24 GFLOPS vs. C1: 54 GFLOPS
Pi 2 USB 2.0 ports vs. C1: 4 USB 2.0 ports
Pi Ethernet: 100MB/s vs. C1 Ethernet: Gigabit

I wanted to see how the ODROID-C1s performed as replacements for the Raspberry Pi units. I was especially looking forward to the greater bandwidth that the C1 Gigabit ethernet would provide, because I have found that the 100Mbps bandwidth of the Pi was a constraint when streaming 1080p Matroska video media, and there was not quite enough bandwidth for the media to stream smoothly.

Building the ARM Ubuntu 14.04 LTS system provided by the ODROID developers was a snap. I simply downloaded the image, uncompressed it, and used the "dd" command to write it to a Class 10 MicroSD card. I then inserted the card into the slot on the underside of the C1, and booted. The instructions for completing the Ubuntu installation were very straight forward.

As I usually do when playing around with an SBC, I installed Tightvncserver for convenient remote access. The ODROID developers did a nice job with their Ubuntu distro, and the LXDE window manager that comes installed by default is perfect.

LXDE is very fast and responsive on the ODROID-C1, and works perfectly

Installing the NFS server software only took a minute, and installing Minidlna 1.1.4 was simply a matter of downloading the source from http://bit.ly/1FQ5SyK and building it.

Once that was done, the C1 became a plug 'n play replacement for the Pi NFS/Minidlna server. There was no muss and no fuss because it simply worked, and is now serving approximately 8TB of media from a stack of USB external drives.

After I replaced the server, I focused my attention on replacing the Pi XBMC unit with the other ODROID-C1. As before, installing the ODROID Ubuntu 14.04 system was quick and easy, since XBMC (now called "Kodi") comes pre-installed on the ODROID-C1 images. All that was needed to get it running was to set it up to automatically mount the media being shared by the file server.

As before, it looked like the ODROID-C1 was going

ODROID-C1 with Gigabit switch

to be a direct replacement that "just worked". Movies and 1080p Matroska Blu Ray files played smoothly. However, I did have some issues with some of my older MPEG-2 movies, which played choppily, like a stop-motion claymation movie. The Raspberry Pi played the movie smoothly, so I then tried using the Gnome MPlayer tool with the ODROID-C1, and the file played well, without any of the choppiness experienced with Kodi.

For more information, please check out the original post at http://bit.ly/1Az3ms9.

ANDROID 5.0 LOLLIPOP

THE NEXT GENERATION OF GOOGLE'S ANDROID CODE

edited by Rob Roy

Android version 5.0 Lollipop, has not yet been officially released for smartphones and tablets, but you can get a sneak preview on your ODROID-C1 for testing purposes. To download the official Android release for 1080p resolution, visit it `http://bit.ly/1DSDs7M`, and for 720p resolution, visit `http://bit.ly/1AG5crp`.

To test out your new apps with Lollipop, it's necessary to use Android Debug Bridge (ADB) to push your app via USB cable. First, connect the microUSB port of the C1 to the USB port of your development machine, then enable developer mode by running the Settings application on the C1, selecting "About", then clicking multiple times on the "Build Number" area.

The WiFi Module 3 from Hardkernel is the only dongle currently supported. The infrared remote controller from Hardkernel is also working. If you'd like to build the image yourself, follow the instructions for the C1 at `http://bit.ly/1wHzuPC`.

For more information, to ask questions or submit improvements, please visit the original post at `http://bit.ly/1B5Ysqh`. You may also check out a video of Lollipop in action on the ODROID at `http://bit.ly/18jgco1`.

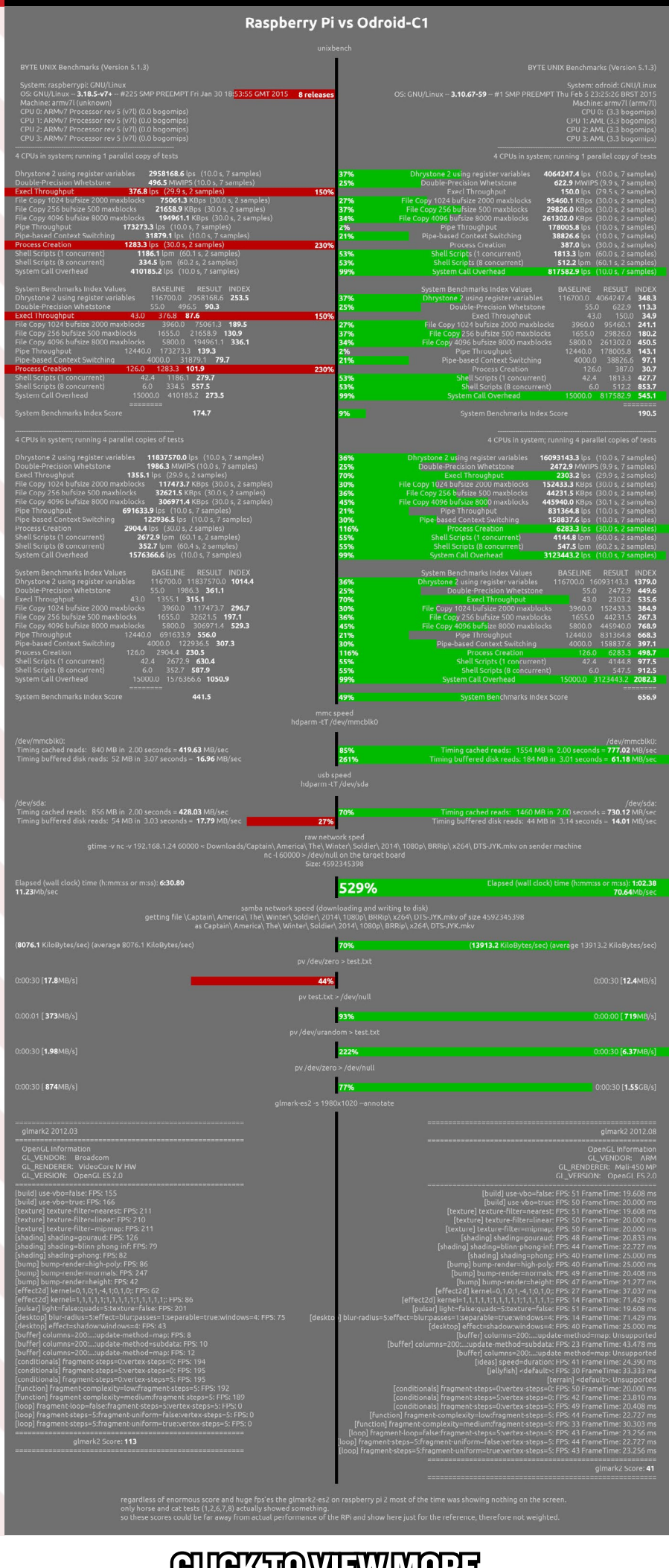
If you prefer using the Cyanogenmod version of Android, you may be interested in the CM 12 Lollipop release at http://bit.ly/1w1okVb, which is available for the XU3 and XU3 Lite models. Its features include:

Kernel 3.10.9 • OpenGL ES 1.1/2.0/3.0 (GPU acceleration) • OpenCL 1.1 EP (GPU acceleration) • Multi-user feature is enabled (Up to 8 users) • On board Ethernet and external USB 3.0 Gigabit Ethernet support • RTL8188CUS , RTL8191SU and Ralink Wireless USB dongle support • exFAT, NTFS(r/w), EXT4 file system support • USB Bluetooth support • USB 3G dongle support • USB GPS dongle support • USB tethering • Portable Wi-Fi hotspot

To add Ralink wifi support, type the following into the Android Debug shell:

```
add ro.hardkernel.wifi2=true to /system/build.prop
```

As development continues, updates may be downloaded from `http://bit.ly/1GilbEi`.



DOCKER ON THE ODROID

USEFUL TIPS

by Uli Middelberg

This article presents several tips for running Docker on ARMv7 based devices. I don't aim to compete with existing tutorials, since they all do a great job. I want to share some of my experiences in setting up and running Docker on an ODROID. You shouldn't expect to run all of the examples mentioned in these tutorials on your ODROID, because they are specifically intended for x86 architecture. However, you should be able to run the examples with some slight modifications.

Before starting

I'm running Ubuntu 14.04.1 on my ARMv7 devices, which makes it easier to install and run Docker. If you prefer to run a different Linux flavor, you might experience small deviations, such as package naming. I created my Ubuntu userland once from scratch and used it with my different ARMv7 devices. Porting Ubuntu to a different device means reusing the userland, then compiling and replacing the vendor-specific kernel and bootloader. For convenience, I regularly push my install images to my Dropbox account at `http://bit.ly/1KoTYCC`.

You shouldn't be afraid to compile a customized Linux kernel, since the most vendor-specific Linux kernels don't include support for the aufs filesystem. Although Docker will run on kernels without aufs, it will work better with the aufs-enabled platforms. Furthermore, some vendors may not include all features needed for Docker to run properly with their default kernel configuration.

Install Docker

Ubuntu 14.04.1 includes a docker.io package (which is actually version 1.0.1), which may be installed from a Terminal window:

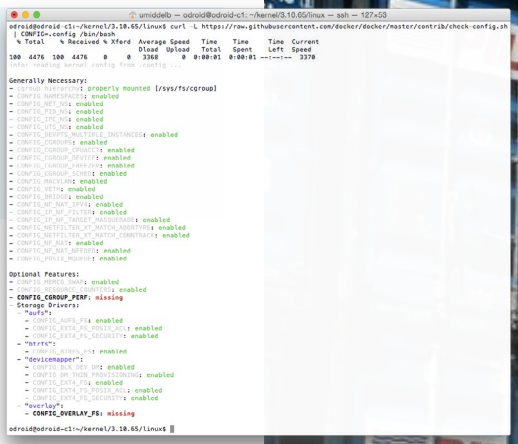
```
$ sudo apt-get install lxc aufs-tools cgroup-lite \
apparmor docker.io
```

This command will install the binary and start-up scripts. But, it is very likely that the Docker service won't start successfully after installing. Usually there is a specific kernel feature that is not immediately available, which Docker relies on.

See what's missing

Running "/usr/bin/docker -D -d" will give you a clue about the actual reason why the Docker daemon may refuse to start. The Docker developers provide a bash shell script which queries whether all Docker-related requirements are met by a particular kernel config. The check-config.sh script (http://bit.ly/1DUaYIh) will either check the config of the running kernel, which is accessible via/proc/config.gz, or will try to read the kernel source's config file specified on the command line. Use the following command to execute the shell script:

```
$ curl -L https://raw.\
githubusercontent.com/docker/\
docker/master/contrib/\
check-config.sh | /bin/bash \
/dev/stdin /path/to/.config
```



Compile the kernel

Many vendors publish their modified kernel sources for their ARM devices along with instructions on how to install the compiled kernel image. Before compiling, you need to install a compiler toolchain (make and gcc) as well as some third-party tools (bc, lzop, u-boot-tools). Be careful, as this will overwrite the default kernel image. You may consider first making a backup of your actual kernel. The "make menuconfig" command gives you a much more user-friendly interface than editing the configuration manually with a text editor.

ODROID-C1

The latest ODROID kernel sources for the ODROID-C1 have integrated Docker and aufs support now, so there is no need to alter any of the kernel options via make menuconfig.

```
$ git clone https://github.com/hardkernel/linux \
--single-branch -b odroidc-3.10.y
$ cd linux
$ make odroidc_defconfig
$ make clean
$ make -j 4 uImage modules
$ make dtbs
$ sudo make modules_install
$ sudo cp arch/arm/boot/uImage /media/boot
$ sudo cp arch/arm/boot/dts/meson8b_odroidc.dtb /
media/boot
```

ODROID-U3

The ODROID kernel sources for the ODROID-U3 have already integrated aufs support.

```
$ git clone --depth 1 \
https://github.com/hardkernel/linux.git \
-b odroid-3.8.y
$ cd linux
$ make odroidu_defconfig && make menuconfig
$ make clean
$ make -j4
$ sudo make modules_install
$ sudo cp arch/arm/boot/zImage /media/boot
```

ODROID-XU3

```
$ git clone --depth 1 \
https://github.com/hardkernel/linux.git \
-b odroidxu3-3.10.y
$ cd linux
$ make odroidxu3_defconfig && make menuconfig
$ make clean
$ make -j4
$ sudo make modules_install
$ sudo cp arch/arm/boot/zImage arch/arm/boot/dts/\
exynos5422-odroidxu3.dtb /media/boot
```

AUFS integration

As previously mentioned, Docker can be sped up up significantly if the kernel includes support for the AUFS filesystem. I've only used the standalone version (kernel module only) so far.

```
$ cd <kernel source directory>
$ git clone git://git.code.sf.net/p/aufs/\
aufs3-standalone aufs3-standalone.git
$ cd aufs3-standalone.git
$ git checkout origin/aufs3.10   # 3.10 ..  3.10.25
$ git checkout origin/aufs3.10.x # 3.10.26 and above
$ git checkout origin/aufs3.14   # 3.14
$ rm include/uapi/linux/Kbuild   # this will keep
your kernel sources config management from being
damaged
$ cp -rp *.patch fs include Documentation ../
$ cd ..
$ cat aufs3-kbuild.patch aufs3-base.patch \
aufs3-mmap.patch aufs3-standalone.patch | patch -p1
```

The aufs release numbering corresponds to the kernel version, so you may reference origin/aufs3.14 for a Linux 3.14.x kernel source code. The 3.10 aufs comes with two branches, 3.10 and

3.10.x. Unfortunately, the aufs developers decided to discontinue support for kernels below 3.14 since the beginning of 2015.

```
$ make oldconfig
...
  Aufs (Advanced multi layered unification filesystem)
support (AUFS_FS) [N/m/y/?] (NEW) m
    Maximum number of branches
    > 1.  127 (AUFS_BRANCH_MAX_127) (NEW)
      2.  511 (AUFS_BRANCH_MAX_511) (NEW)
      3.  1023 (AUFS_BRANCH_MAX_1023) (NEW)
      4.  32767 (AUFS_BRANCH_MAX_32767) (NEW)
    choice[1-4?]: 1
    Detect direct branch access (bypassing aufs)
(AUFS_HNOTIFY) [N/y/?] (NEW) y
      method
      > 1.  fsnotify (AUFS_HFSNOTIFY) (NEW)
      choice[1]: 1
    NFS-exportable aufs (AUFS_EXPORT) [N/y/?] (NEW) y
    support for XATTR/EA (including Security Labels)
(AUFS_XATTR) [N/y/?] (NEW) y
    File-based Hierarchical Storage Management (AUFS_
FHSM) [N/y/?] (NEW) y
    Readdir in userspace (AUFS_RDU) [N/y/?] (NEW) y
    Show whiteouts (AUFS_SHWH) [N/y/?] (NEW) y
    Ramfs (initramfs/rootfs) as an aufs branch (AUFS_
BR_RAMFS) [N/y/?] (NEW) y
    Fuse fs as an aufs branch (AUFS_BR_FUSE) [N/y/?]
(NEW) y
    Hfsplus as an aufs branch (AUFS_BR_HFSPLUS)
[Y/n/?] (NEW) y
    Debug aufs (AUFS_DEBUG) [N/y/?] (NEW) n

$ configuration written to .config
```

This will add the aufs related kernel configuration items to your existing config. Make sure to choose "m" for aufs support.

OverlayFS

Docker also supports OverlayFS, which was introduced with the 3.18 linux kernel. If you've managed to run Linux 3.18 on your arm device OverlayFS, it could be a replacement for aufs.

Test Docker

Now, it's time to recompile and install the new kernel. If everything went well, the Docker service will run on your device and listens for service requests.

```
$ sudo docker info
Containers: 0
Images: 0
```

```
Storage Driver: aufs
 Root Dir: /var/lib/docker/aufs
 Backing Filesystem: extfs
 Dirs: 0
Execution Driver: native-0.2
Kernel Version: 3.10.66-aufs
Operating System: Ubuntu 14.04.1 LTS
CPUs: 4
Total Memory: 983.4 MiB
Name: odroid-c1
ID: 324D:YXY2:2XQP:CATB:KIQD:AFXA:UZBQ:IEPO:WSB5:3Y2R:
O5QU:FRDU
```

Choosing the image

Most of the Docker images are intended for the x86 platform, and Docker itself isn't platform dependent, although the Docker images contain a record of the architecture on which they have been created:

```
$ sudo docker images
REPOSITORY                TAG                 IMAGE ID
CREATED             VIRTUAL SIZE
<none>                    <none>
d8115ff9b785        22 hours ago        301.5 MB
armv7/armhf-ubuntu_core   14.04               c11f-
1521cacf        2 weeks ago         159 MB
$ sudo docker inspect d8115ff9b785 | \
jq '.[] | .Architecture'
"arm"
```

When running a command using an x86-built image on an ARMv7 device, the following error will occur:

```
$ sudo docker run ubuntu /bin/echo 'Hello world'
FATA[0205] Error response from daemon: Cannot start
container 9b55520a44ad4c069cc577afa51983713afb8e96ebe-
55a736e0819706b94f10b: exec format error
```

Most of the Docker images for ARMv7 devices in the Docker registry have a name starting with "armhf-" You may search for them using the following command:

```
$ sudo docker search armhf-
NAME                          DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
mazzolino/armhf-debian        Debian Wheezy base image for armhf devices      4
mazzolino/armhf-ubuntu        Ubuntu-Core images for armhf (ARMv7) devices    4
armv7/armhf-archlinux         archlinux arm docker image for the ARMv7(a...   2
armbuild/ubuntu-debootstrap   ARMHF port of ubuntu-debootstrap                1                      [OK]
armv7/armhf-ubuntu_core       ubuntu core docker images for the ARMv7(ar...   1
hominidae/armhf-ubuntu        ubuntu trusty/14.04 image (minbase) for ar...   1
dehy/armhf-couchdb            ARMHF port of klaemo/couchdb                    0
mazzolino/armhf-tiddlywiki    Tiddywiki5 on NodeJS for armhf (ARMv7) dev...   0
```

```
hominidae/armhf-wheezy        armhf image of Debian Wheezy, made with de...   0
dpniel/dekko-armhf            armhf utopic image to build dekko click pa...   0
hominidae/armhf-supervisord   ubuntu trusty/14.04 for armhf architecure ...   0
chanwit/fedora-armhf          Fedora for the armhf architecture               0
mazzolino/armhf-twister       Twister for armhf / armv7 devices               0
jalessio/armhf-ubuntu         Cloned from mazzolino/armhf-ubuntu              0
hominidae/armhf-archlinux     ArchLinux base image for armhf architectur...   0
pshouse/armhf-guacamole       armhf-ubuntu version of hall/guacamole          0
hominidae/armhf-cupsd         armhf image Wheezy, with sshd and cups pri...   0
onlinelabs/armhf-ubuntu                                                       0
mazzolino/armhf-butterfly     Butterfly for armhf (ARMv7) devices             0
moul/armhf-busybox                                                            0                    [OK]
armv7/armhf-ubuntu            'official' ubuntu docker images for the AR...   0
armv7/armhf-baseimage         ubuntu docker images for the ARMv7(armhf) ...   0
zsoltm/ubuntu-armhf           Ubuntu 14.04.1 minimal install, latest upd...   0
armv7/armhf-fedora            minimal fedora docker images for the ARMv7...   0
mazzolino/armhf-prosody       Secured Prosody XMPP server for armhf (ARM...   0
```

I publish my Docker images using the armv7 profile on Dockerhub. So, let's try the same command using the armv7/armhf-ubuntu_core image:

```
$ sudo docker run armv7/armhf-ubuntu_core /bin/echo \
'Hello world'
Unable to find image 'armv7/armhf-ubuntu_core:latest'
locally
Pulling repository armv7/armhf-ubuntu_core
c3802ac1b0ad: Download complete
Status: Downloaded newer image for armv7/armhf-ubuntu_
core:latest
Hello world
```

This time, I've used a different device which hasn't downloaded the image before. After downloading, it will be part of the local image cache:

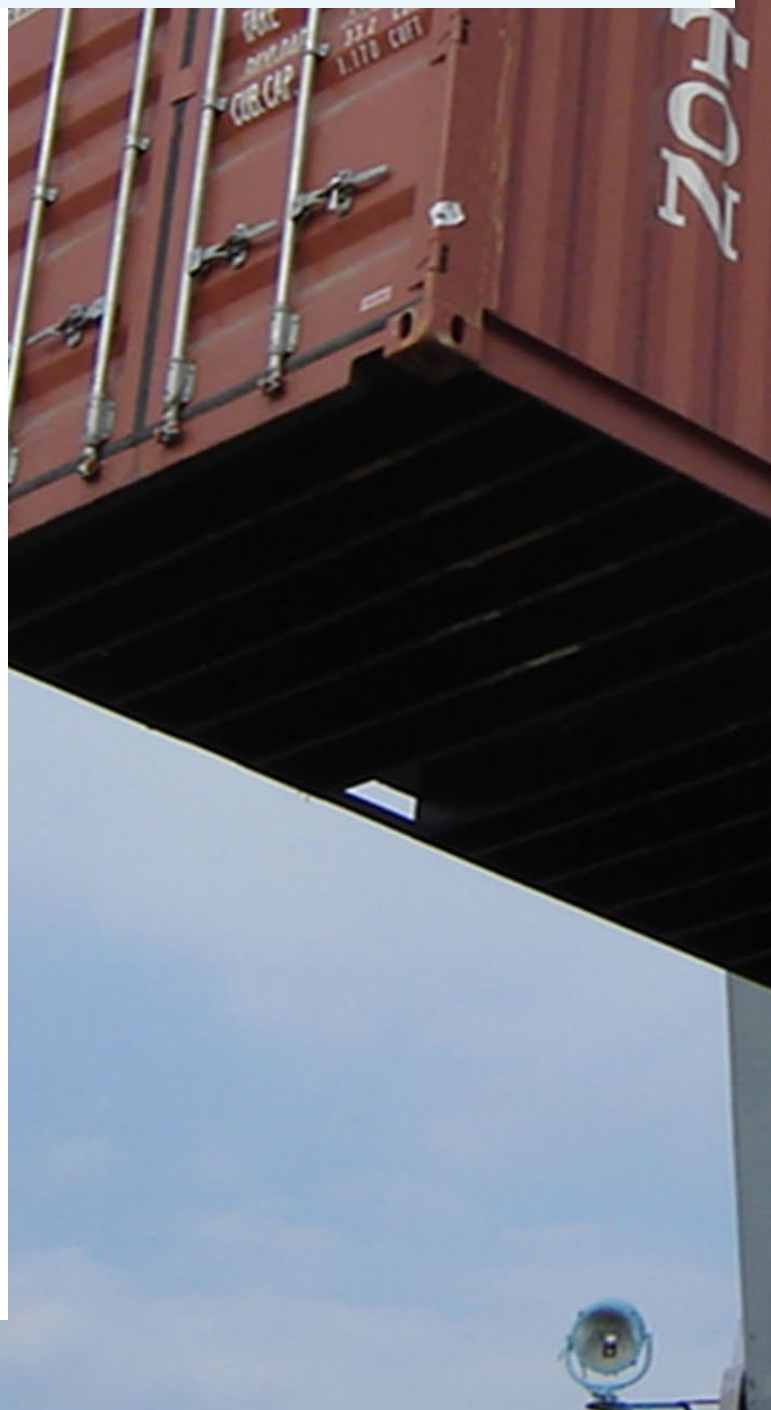
```
$ sudo docker images
REPOSITORY                TAG                 IMAGE ID
CREATED             VIRTUAL SIZE
armv7/armhf-ubuntu_core   latest
c3802ac1b0ad        About an hour ago   163.5 MB
```

I have found lots of excellent online resources to help you get a better understanding of Docker, including `http://bit.ly/1MgEBtz`, `http://bit.ly/1FnLGUY`, and `http://bit.ly/1A5PpTQ`.

Updating from version 1.0.1

You may have experienced a tedious bug in Docker 1.0.1 which will, from time to time, prevent containers from starting:

```
$ sudo docker run armv7/armhf-ubuntu_core /bin/echo \
```



```
 'Hello world'
2015/01/15 17:57:10 finalize namespace drop
capabilities operation not permitted
```

This bug seems to have been fixed in Docker version 1.4.0. You can build Docker 1.4.0 from source with Docker 1.0.1, but you will need patched sources from `http://bit.ly/1BR3mJL`, otherwise the build-in fuse will prevent docker from starting:

```
FATA[0000] The Docker runtime currently only supports
amd64 (not arm).  This will change in the future.
Aborting.
```

The Docker wiki pages at `http://bit.ly/1NseXDB` will guide you in building the Docker binary from source. Starting with release 1.5.0, the Docker developers removed the "fuse" which explicitly requires the amd64 platform, and integrated most of the patches, making Docker 32-bit safe. You can build Docker for ARMv7 with the latest original sources now. The only thing you still need is a slightly modified Dockerfile for the armhf/ARMv7 platform:

```
$ git clone -b 'v1.5.0' --single-branch \
https://github.com/docker/docker.git
$ cd docker
$ curl -L https://github.com/umiddelb/armhf/\
raw/master/Dockerfile.armv7 > Dockerfile
$ make build
$ make binary
$ sudo service docker.io stop
$ sudo cp bundles/1.5.0/binary/docker-1.5.0 /usr/bin
$ (cd /usr/bin; sudo mv docker _docker; sudo ln -s
docker-1.5.0 docker)
$ sudo service docker.io start
```

It is very likely that the bug mentioned above will interrupt the build process. In this case, you have to issue the "sudo make build" command more than once. Alternatively, you may download the final Docker binary from `http://bit.ly/1aRZu0P` for convenience. The binary itself is linked statically, and will run on other Linux flavors as well:

```
$ file /usr/bin/docker-1.5.0
/usr/bin/docker-1.5.0: ELF 32-bit LSB  executable,
ARM, EABI5 version 1 (SYSV), statically linked, for
GNU/Linux 2.6.32, BuildID[sha1]=eef157201c4e1d888d0977
0a8187edf956605176, not stripped
```

Just replace the existing Docker binary in /usr/bin with the new one:

```
$ sudo docker version
[sudo] password for umiddelb:
Client version: 1.5.0
Client API version: 1.17
Go version (client): go1.4.1
Git commit (client): a8a31ef-dirty
OS/Arch (client): linux/arm
Server version: 1.5.0
Server API version: 1.17
Go version (server): go1.4.1
Git commit (server): a8a31ef-dirty
```

Installing docker on Fedora 21

Unfortunately, there is no rpm package for Docker available on armhfp. So, the installation procedure is a little bit longer, and most of the steps can be derived from the source package docker-io-1.4.1-7.fc22.src.rpm:

```
$ sudo yum install rpm-build
$ sudo yum install glibc-static
$ sudo rpmbuild --rebuild \
http://copr-be.cloud.fedoraproject.org/results/\
gipawu/kernel-aufs/fedora-21-x86_64/\
aufs-util-3.9-1.fc20/aufs-util-3.9-1.fc21.src.rpm
$ sudo rpm -i /root/rpmbuild/RPMS/armv7hl\
/aufs-util-3.9-1.fc21.armv7hl.rpm
$ sudo yum install lxc bridge-utils device-mapper \
device-mapper-libs libsqlite3x docker-registry \
docker-storage-setup
$ mkdir docker
$ cd docker
$ wget ftp://fr2.rpmfind.net/linux/fedora/linux/\
development/rawhide/source/SRPMS/d/\
docker-io-1.4.1-7.fc22.src.rpm
$ rpm2cpio docker-io-1.4.1-7.fc22.src.rpm | cpio -idmv
$ tar -xzf v1.4.1.tar.gz
$ curl -L https://github.com/umiddelb/armhf/raw/\
master/bin/docker-1.5.0 > docker
```

This installation procedure is derived from docker-io-1.4.1-7.fc22.src.rpm::docker-io.spec:

```
# install the docker binary
$ sudo install -p -m 755 docker /usr/bin/docker

# install bash completion
$ sudo install -p -m 644 docker-1.4.1/contrib/\
completion/bash/docker /usr/share/bash-completion/\
completions

# install container logrotate cron script
```

```
$ sudo install -p -m 755 docker-logrotate.sh \
/etc/cron.daily/docker-logrotate

# install vim syntax highlighting
$ sudo install -p -m 644 docker-1.4.1/contrib/syntax/\
vim/doc/dockerfile.txt /usr/share/vim/vimfiles/doc
$ sudo install -p -m 644 docker-1.4.1/contrib/syntax/\
vim/ftdetect/dockerfile.vim /usr/share/vim/vimfiles/\
ftdetect
$ sudo install -p -m 644 docker-1.4.1/contrib/syntax/\
vim/syntax/dockerfile.vim /usr/share/vim/vimfiles/syntax

# install udev rules
$ sudo install -p docker-1.4.1/contrib/udev/\
80-docker.rules /etc/udev/rules.d

# install systemd/init scripts
$ sudo install -p -m 644 docker.service \
/usr/lib/systemd/system

# for additional args
$ sudo install -p -m 644 docker.sysconfig \
/etc/sysconfig/docker
$ sudo install -p -m 644 docker-network.sysconfig \
/etc/sysconfig/docker-network
$ sudo install -p -m 644 docker-storage.sysconfig \
/etc/sysconfig/docker-storage

# install docker config directory
$ sudo install -dp /etc/docker

$ getent passwd dockerroot > /dev/null || sudo \
/usr/sbin/useradd -r -d /var/lib/docker -s \
/sbin/nologin -c "Docker User" dockerroot
$ sudo /bin/systemctl enable docker.service
```

For more information, or to post questions or comments, please visit the original post at `http://bit.ly/1zZxyxP`.

FLYING ODROID

GET YOURSELF AIRBORNE

by Gregory Dymarek

You will be the envy of your neighborhood when everyone sees you flying this around!

Drones have become becoming more and more popular recently. We hear about them in the media because they are very versatile machines. On one side, armies have been using them for some time, and Amazon is looking into using them for parcel delivery. On the other side, they are great hobby tools, since people use them for aerial photography. You can also use them to take part in quadcopter racing competitions and other hobbyist purposes.

Quadcopters may be purchased off the shelf, or you build one yourself like I did. Ready-built quadcopters can cost as little as an average children's toy, but the more professional ones are priced closer to a small car. The principal behind them is the same, but the quality and functionality will vary a lot.

Fortunately, quadcopters are known to be relatively simple to build, and with the help of some DIY skills, you can save a lot of money. To build one at home, you will need some basic soldering skills, or learn them as you go. With some time on your hands, you can program them in the way that you want.

My first quadcopter was a Hubsan H107. Flying it was so enjoyable that it made me dive straight into the world of quadcopters. I would recommend this particular model of quadcopter to anyone who is new to the hobby. It is cheap and robust so that you can practice your flying skills. It also delivers very good performance for its price.

Having spent endless hours reading about different quadcopters, it became apparent that I could try to build one myself from scratch. That is how the AvrMiniCopter project was born. My objective was to create a quadcopter controller that runs on a Linux system. In this way, I could make the controller extensible and re-use standard Linux drivers. Also, programming on a fully featured Linux system is far quicker and easier than creating programs for embedded boards. The total cost for all of the components was approximately USD$150, not including the ODROID, PS3 controller and tablet.

Key components

- Warthox Frame set (25cm arm)
- 4 motors SunnySky X2212 KV980
- 4 ESCs Afro Slim 20A
- ODROID-W
- Arduino Pro Mini 16MHz
- MPU9150 sensor board
- BMP180 sensor board
- Bluetooth or WIFI usb dongle
- Raspberry Pi camera, or any h264 Linux compatible model
- Turnigy 2200mAh, 3S 25C LiPo battery, battery monitor and battery charger

Project history

On my first go, I gathered all the basic parts that I needed, then wired everything up. At that time I was using a Raspberry Pi as the only board to control it. The result was satisfactory, since I wrote my own controller code for the RPi, and immediately had a flying quadcopter. However, the frame and the motors chosen were not a perfect fit, and I had to do some additional work in order to mount them firmly. On top of it, I was getting some random glitches and delays in the execution,

Birds-eye view from the QuadCopter while flying

which caused the quadcopter to crash on occasion. Later on, I figured out this was due to the SD card socket on the RPi that allowed the card to slip out, resulting in a kernel panic. It also became apparent that the project could benefit from a real-time system. The Raspberry Pi is not a real-time system, and as such, cannot guarantee a response within a strict timeframe, which is required for smooth and reliable flight. I tried to address this issue by using the well known real-time framework called Xenomai. However, the amount of additional work needed, especially around driver implementations, made me look for a different approach.

On my second attempt, I decided to use a dedicated controller board, for which I chose the Arduino Pro Mini along with an ODROID-W. The Arduino is a proven and well-known real-time board that can deliver smooth stabilization of the quadcopter, while the ODROID-W was used for other, less time-sensitive computations like barometer readings, PS3 controller support and video capture. Also, the move to the ODROID-W brought a significant physical size reduction of the construction. The RPi controller software that I originally wrote was then ported to the Arduino platform, and an image for the ODROID-W was created for which a buildroot environment was set up to streamline the image building process. This approach proved to work flawlessly and has not changed since I first implemented it.

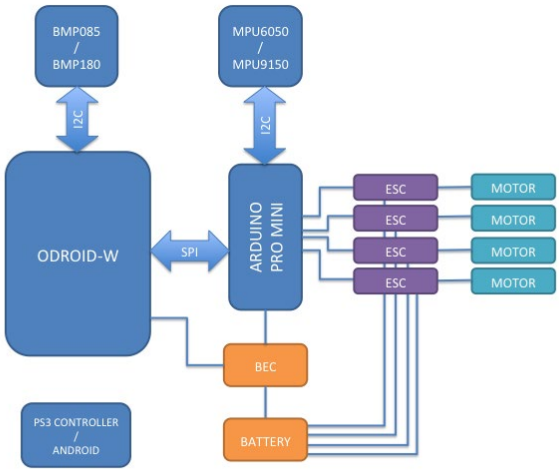
Parts

To build your AvrMiniCopter, you will need to source some parts. In my case, I bought most of them from HobbyKing and eBay. There are number of websites and forums where you can find a lot of information about what types and sizes of motors, Electronic Speed Controls (ESCs) and batteries to use. Some of them will also estimate the duration of flight that you will be getting. My advice is not to make the flight time the priority for your first quadcopter, otherwise the build will get more difficult.

- Frame: Choose a frame that will have enough space for all your gear. It is easier to build a bigger quadcopter. However, it will be more expensive and more dangerous to fly.

- Brushless Motors: There are too many types of motors to consider all of them. Choose ones that fit the frame and have more than enough power. The rule of thumb is to find ones that cumulatively generate enough thrust to lift twice the weight of your quadcopter. Bigger quadcopters will accommodate bigger propellers, so smaller KV motors (revolutions per volt) will be acceptable, making the quadcopter more efficient.

- Electronic Speed Control (ESC): For the AvrMiniCopter project, you will need ESCs that take Pulse Width Modulation (PWM) as an input signal which can deliver more than enough power for your motors. Some ESCs have a built-in Battery Eliminator Circuit (BEC) so that you will not need to purchase a dedicated one.



Block diagram of the hardware design of the QuadCopter

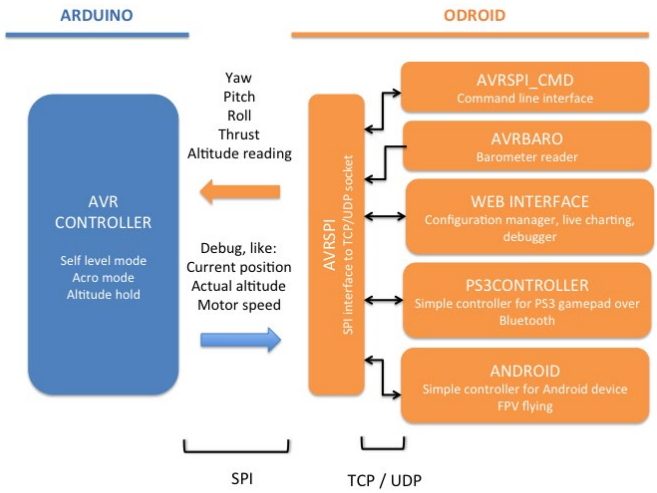
- LiPo Battery: The battery is one of the single heaviest components of your quadcopter and will greatly affect the flight time. Make sure you have a battery with the correct voltage and discharge rating for your motors.

Software

The AvrMiniCopter software consists of two parts: an Arduino controller and the host (ODROID) management tools. The Arduino code is an 8-bit implementation of a flight controller, and its core functionality is reading and computing the current position of the quadcopter, keeping it stable at all times. On the other side, the host's prime activity is to read the pilot's PS3 controller and forward any requests to the Arduino for execution in terms of yaw, pitch, roll, and thrust.

The actual implementation is a bit more fragmented, and is divided into such modules as the AVRSPI component that translates SPI interface into TCP socket, the AVRBARO function that reads barometer data in a loop and passes back the

Block diagram of the software design of the QuadCopter



data to Arduino through AVRSPI, the AVRCONTROLLER module which handles PS3 gamepad inputs, configuration management and flight log data. There is also a web interface for communicating with the controller and setting it up, which is a very convenient way of adjusting and debugging while running it outside while using a mobile device.

All of the software that is needed also comes in a pre-compiled version which may be downloaded as a disk image that can be simply flashed onto an SD card, which allows you to get started without the hassle of compiling anything. The image is a custom-built, minimalist Linux distribution that is created and maintained using a set of buildroot scripts. All the scripts are available on the project GitHub page at http://bit.ly/1NingC0, and the pre-built Arduino images may be downloaded from http://bit.ly/1EOKc8e.

Functionality

Currently, the AvrMiniCopter software is a full-featured controller capable of controlling quadcopters of any size in an X configuration. It delivers two flight modes: stabilized mode, where the quadcopter auto-levels itself, and rate mode for more agile flying. With the help of a barometer, the quadcopter will also be able to hold a target altitude.

As for the flight controller, a PS3 gamepad that works to a distance of around 50 meters is supported, which communicates via bluetooth. A Wi-fi solution is in development that will provide a longer rang, as well as live camera streaming to your mobile device. Soon, you will be able to use your smartphone or tablet to control the flight without the need for a dedicated gamepad, saving the cost of buying a PS3 controller. I am also looking to support a GPS module in order to do pre-programmed flights, but this is currently in a planning stage.

Notes

Consider making or buying a rigid frame for your first prototype from a strong but light material such as aluminum. I have broken more than 10 wooden frames while testing and learning before moving to an aluminum frame. Make sure to keep your quadcopter away from anything that can break, and do not install the propellers while testing.

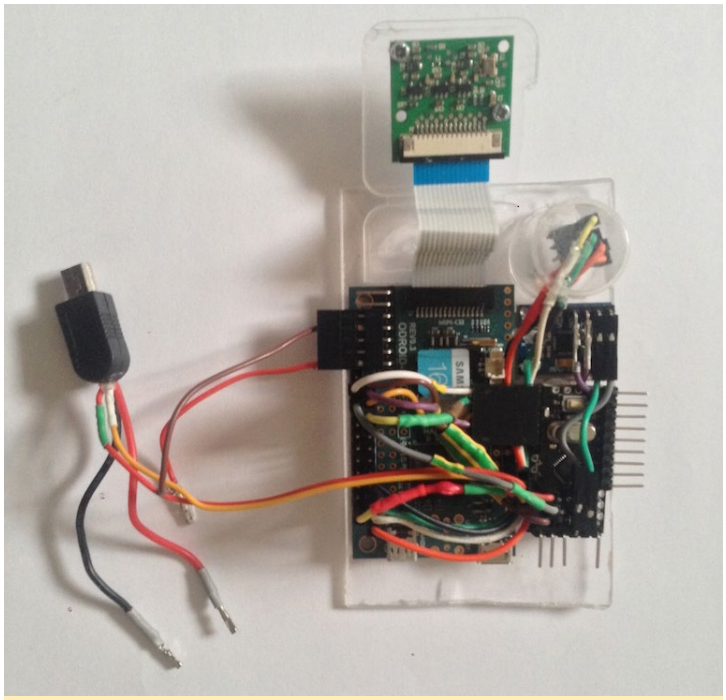
Keep in mind that quadcopters are great toys to learn and fly, and as hobbyists, we do not want to provoke more regulations and restrictions because of misuse or lack of common sense. For example, never fly your quadcopter next to an airport or other people. Propellers spin with an incredible speed and can easily cause and injury to bystanders.

There are many other aspects of quadcopter flying that have not been addressed in this article. Nevertheless, I really hope that this has provided you with some insight into the topic and gotten your attention. You may post any questions on the ODROID forums at http://forum.odroid.com. For more information about the AvrMiniCopter project, please visit the project wiki at http://bit.ly/1DX3OWb. You may also check out a YouTube video of the quadcopter project in action at http://bit.ly/1w5gvhv.

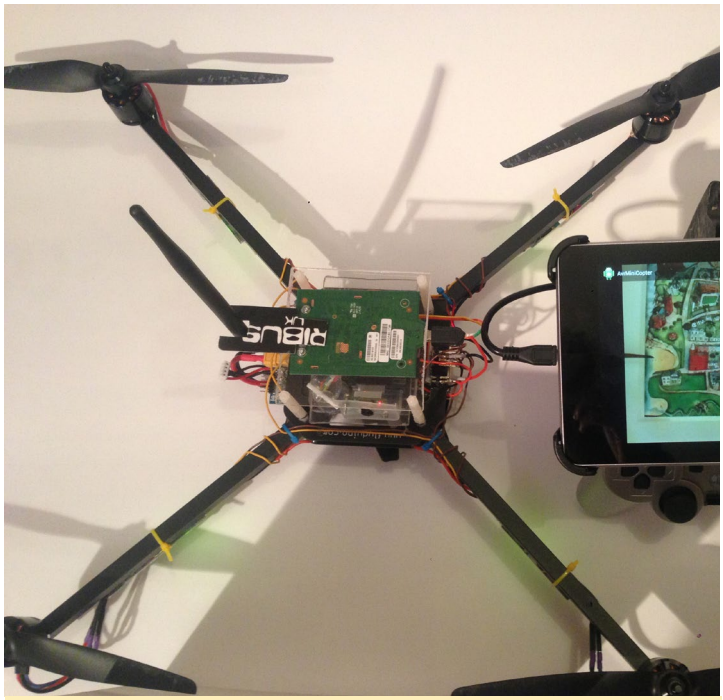
The QuadCopter components are a tablet and PS3 controller

The tablet showing the camera view from the QuadCopter

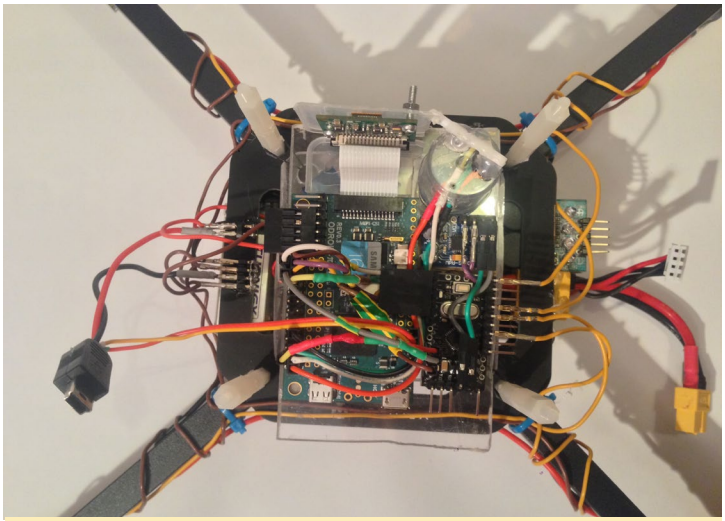
Closeup of the PS3 controller attached to the tablet



ODROID-W components used in the Quadcopter



The protective case of the QuadCopter prevents damage



Closeup of QuadCopter controller mounted on the propellers

QuadCopter fully assembled and ready for its first flight

The QuadCopter can levitate in one place without wavering

The QuadCopter in flight sailing smoothly over top of the trees

ANDROID DEVELOPMENT

DISSECTING AND MODIFYING THE APK FILE

by Nanik Tolaram

In my previous article, we discussed the internals of an Android Package Kit (APK) file, including how it is structured, as well as the tools that you can use to generate it. In this article, we will look at the different tools that are available to inspect and make changes to an APK file. As an example, we are going to dissect the GitHub Android app, which can be downloaded from http://bit.ly/1Ecc0Tp.

Basic Structure

Before dissecting the APK file, let's take a look at the contents of the file in Figure 1.

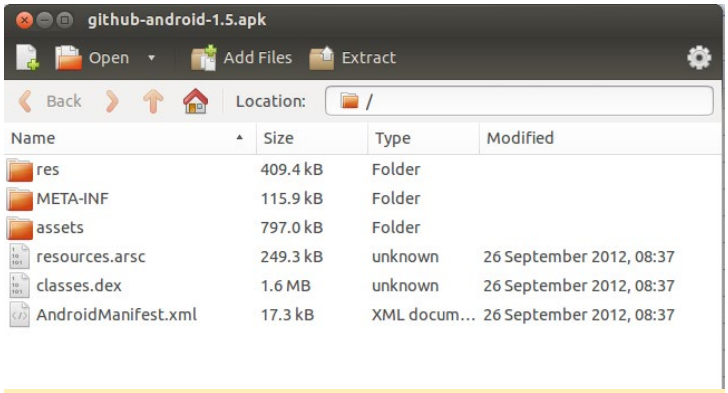


Figure 1 - Inside github-android-1.5.APK

As you can see, there are 3 main folders: res/ contains the resource file (such as layout and strings), assets/ which normally contains big files (such as videos and images) and META-INF/ containing informational files such as the SHA1 hash coding of the files inside the res/ and assets/ folders. The classes.dex is the main file containing the source code of the application, which is also the file that is read and executed by the Android Dalvik virtual machine.

Unpacking

Now that we have seen the basic structure of APK files, we are going to take a look at how to unpack them. One thing that you must remember is that everything inside the APK is not in text format. For example, if you unzip the APK file and open the AndroidManifest.xml file, you will see illegible characters, as shown in Figure 2.

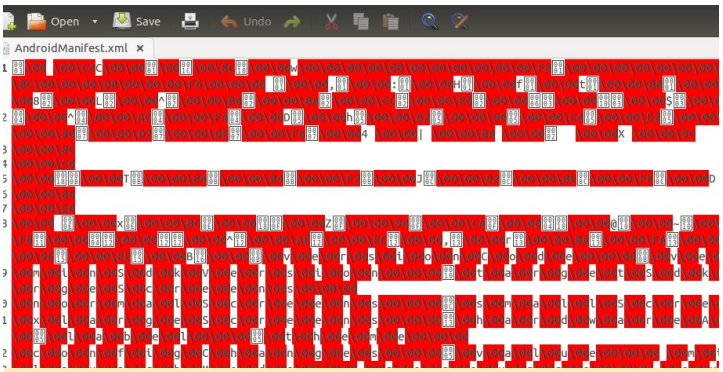


Figure 2 - AndroidManifest.xml gibberish view

When you want to see the content of the APK in its original text format, you need to unpack it using a tool called APKtool that you can download from http://bit.ly/1FPDVHo/. For this article, we will be using version 1.5.2. Follow the installation steps outlined on the wiki page at http://bit.ly/1EHQCpy to install it on your development machine.

First, make sure that you have your Android SDK directory in your path. As an example, here is the content of my PATH variable on my computer:

/home/nanik/Downloads/android-sdk-linux/build-tools/20.0.0/:/home/nanik/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/home/nanik

Execute the following script to decode the APK file:

```
#!/bin/sh
java -jar <APKtool_directory>/APKtool.jar d ./github-android-1.5.APK
```

Make sure to replace the <APKtool_directory> with the location of your local directory that you have unzipped using the APKtool into. Once you run the command, you will see the following output:

```
I: Baksmaling...
I: Loading resource table...
I: Loaded.
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file:
/home/nanik/APKtool/framework/1.APK
I: Loaded.
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Done.
I: Copying assets and libs...
```

One thing to note is this part of the output:

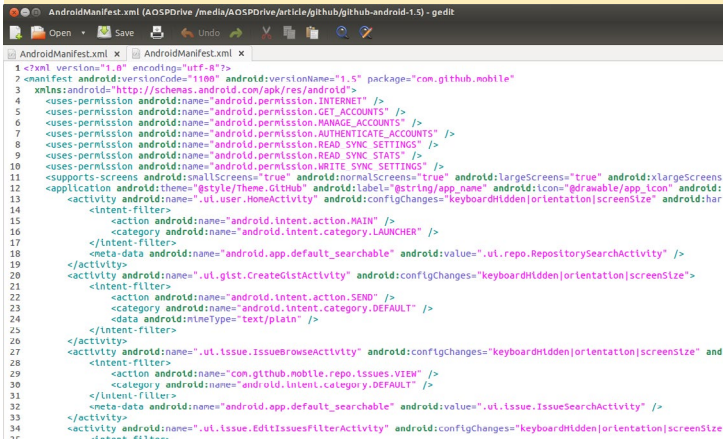
```
I: Loading resource table from file:
/home/nanik/APKtool/framework/1.APK
```

This line is important, as it shows which framework APK file that Android is using to decode the file. If you are decoding Android 5.0, it will use a different framework APK as compared to Android 4.0 or 2.3. Once you complete the decoding step, you will see a new folder named after the APK file, and the content will look like Figure 3. Notice that it is the same structure as in Figure 1. If you open the AndroidManifest.xml, you will see something similar to Figure 4.

Figure 3 - Inside ./github-android-1.5

```
drwxrwxr-x  6 nanik nanik  4096 Feb 17 21:36 ./
drwxrwxr-x  3 nanik nanik  4096 Feb 17 21:36 ../
-rw-rw-r--  1 nanik nanik 12129 Feb 16 22:15 AndroidManifest.xml
-rw-rw-r--  1 nanik nanik   259 Feb 16 22:15 apktool.yml
drwxrwxr-x  4 nanik nanik  4096 Feb 16 22:15 assets/
drwxrwxr-x  3 nanik nanik  4096 Feb 17 21:36 build/
drwxrwxr-x 47 nanik nanik  4096 Feb 16 22:15 res/
drwxrwxr-x  7 nanik nanik  4096 Feb 16 22:15 smali/
```

Figure 4 - Decoded AndroidManifest.xml



Making Changes

Let us try to make some simple changes to the decoded APK file, and then pack it up and deploy to the ODROID, or to your own Android device. The original screen before modification can be seen in Figures 5 and 6.

Figure 5 - Original login screen

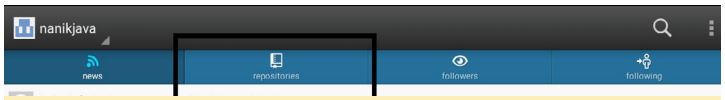


Figure 6 - Original main screen

We are going to make several text changes by replacing the words "Login or Email" and "Password", and replace the word "repositories". The codes that needs to be changed are found inside the res/values/strings.xml file. Modify the following line:

```
before: <string name="login_or_email">Enter or
Email</string>

after: <string name="login_or_email">Please enter
your login or email</string>
```

Then, change the Password text:

```
before: <string name="password">Password</string>

after: <string name="password">Enter your password</
string>
```

Finally, change the word repositories:

```
before: <string name="tab_
repositories">repositories</string>
after: <string name="tab_repositories">repo</string>
```

Packing

Now that we have completed the changes, we need to pack or encode the files back to an APK, then sign it and deploy to the device. To pack the files, use the following script:

```
#!/bin/sh
```

```
java -jar <APKtool_directory>/\
APKtool.jar b \
./github-android-1.5/ \
./github.APK
#the following command is to gen-
erate .keystore
#-------------------------------
---------------------------
#keytool -genkey -keystore
./<yourpersonal>.keystore \
-validity 10000 \
-alias <yourkeystorename>

#the following command is to
build and sign the .APK
<your_jdk_directory>/bin/\
jarsigner -keystore ./<yourper-
sonal>.keystore -verbose \
./github.APK <yourkeystorename>
```

The first line that runs the APK tool is used to encode/build the APK file, and once that's done, you'll need to use the jarsigner tool to sign the APK with your own keystore. After running the jarsigner tool, you will see the following output:

```
I: Checking whether sources has
changed...
I: Checking whether resources has
changed...
I: Building APK file...
Enter Passphrase for keystore:
   adding: META-INF/MANIFEST.MF
   adding: META-INF/NANIK.SF
   adding: META-INF/NANIK.DSA
  signing: assets/lib/util/\
loadmode.js
  signing: assets/lib/util/\
multiplex.js
  signing: assets/lib/\
codemirror.js
  signing: assets/mode/clike/\
clike.js
  signing: assets/mode/clike/\
index.html
  signing: assets/mode/clike/\
scala.html
  signing: assets/mode/clojure/\
clojure.js
  signing: assets/mode/clojure/\
index.html
  ..
  ..
  ..
  signing:
res/xml/sync_adapter.xml
  signing: AndroidManifest.xml
  signing: classes.dex
  signing: resources.arsc
     pkg: /data/local/tmp/
github.APK
```

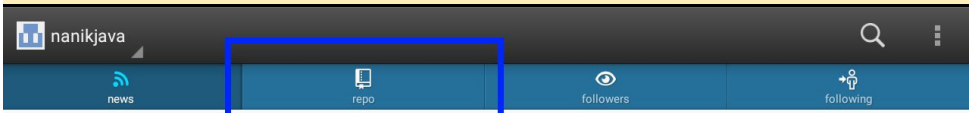
Finally, you will get a file called github.APK which you can install using the following command:

```
$ adb install /path/to/app.apk
```

After the app is installed, you can run it to verify that the new text appears as shown in Figures 7 and 8:

Figure 7 - Modified login screen

Figure 8 - Modified main screen



ANGRY BIRDS TRANSFORMERS

A GREAT MIX OF OLD AND NEW HEROES

by Jeremy Leesmann

What do you get when you mix Transformers and Angry Birds…AUTOBIRDS! It's a fast-paced action side-scrolling shooter. You'll enjoying destroying all the pigs in new ways using different methods such as lasers and dynamite, and it runs great on the ODROID-U3.

https://play.google.com/store/apps/details?id=com.rovio.angrybirdstransformers

The main screen shows the island where the pigs have set up their home base, and you need to clear them out fast!

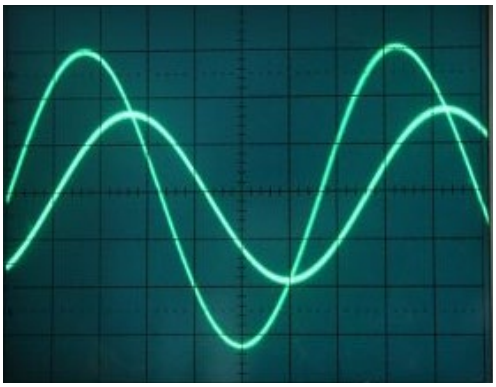
This version of Angry Birds is very fast-paced, and keeps you moving

The Transformers will always win, even against the devious pigs

OSCILLOSCOPE

USING THE ODROID-CI AS A BENCH OSCILLOSCOPE

by Venkat Bommakanti



The integration of General Purpose Input/Output (GPIO) capabilities with System-On-a-Chip (SoC) based complex computation devices has resulted in low-cost, powerful, maker-friendly development platforms like the ODROID-C1. The C-Tinkering Kit along with the wiringPii library for C1 running Lubuntu, has all of the open-source essentials to start prototyping some intelligent sensory circuits.

Now, what if we took that combination and added to that mix, an open-source oscilloscope and a logic analyzer with a USB interface? We would get a moddable, complete and yet portable electronics lab which is capable of signal test, measurement, data acquisition, storage and analysis. It is a matter of time before every maker, interfacing with the sensory world, realizes the need for such a setup.

The oscilloscopes and logic-analyzers selected for this article are mere examples of the kind of measurement devices that one could use, with an eye towards full open-source software ownership, good price-to-performance ratio, and robust feature set. Keep in mind that Hardkernel does not endorse any specific device. The reader is expected to perform adequate research to select a measurement device appropriate for their need, realizing that these products may themselves be early-stage devices requiring further refinement. Be aware that pricing and capabilities are wide-ranging..

Requirements

1. An ODROID-C1. While this article targets a C1, it can apply to a U3 or an XU3/XU3-Lite device.
2. C1 accessories such as, HDMI cable, CAT 5E+ ethernet cable or WIFI 3 dongle, recommended PSU, RTC battery, or ODROID-VU.
3. A 16GB+ eMMC 5.0 card with latest C1 specific Lubuntu desktop image and/or a 16GB+ Class 10 MicroSD with an SDCard reader/writer.
4. Mali OpenGL-ES SDK v2.4.4
5. Mono runtime 3.2.8
6. A network where the device has access to the internet and the ODROID forums.
7. Networked access to the C1 via utilities like PuTTY, FileZilla, TightVNC Viewer (MS Windows 7+), or Terminal (Mac, Linux) from a testing desktop.
8. A C-Tinkering kit
9. An oscilloscope and logic analyzer such as the DSLogic (DreamSource Lab) or SmartScope (LabNation), or a logic analyzer such as the BeagleLogic. It is beneficial to use a sigrok-compliant device, to avail all the benefits of an open-source signal analysis library.

Install Lubuntu

Install the latest C1 image on to the eMMC card and attach the eMMC card to the C1. With the VU display attached, boot up the system. Run the ODROID Utility and set the display resolution to say 800p and reboot. Then, expand the installation partition to use all of the eMMC by selecting the "Resize your root partition" option.

Reboot and re-run the ODROID Utility again, configuring and updating all remaining aspects of the system, which may require a final reboot. For the most recent images, you would have to run the following commands in order, to update the system:

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
```

Prepare the system

Install the required software components using the following command:

```
$ sudo apt-get install \
git-core gcc g++ \
autoconf automake make \
cmake libtool \
pkg-config libglib2.0-dev \
libglib2.0 \
libzip-dev libudev-dev \
libasound2-dev \
libasound2 libusb-1.0 \
python3-dev python3 check \
libqt4-dev libboost-dev \
libboost-all-dev \
libboost-test-dev \
libboost-thread-dev \
libboost-system-dev \
mono-runtime
libmono-system-core4.0-cil \
libmono-system-drawing4.0-cil \
libmono-cairo4.0-cil \
libsdl-mixer1.2 \
libsdl1.2debian \
libmono-system-xml-linq4.0-cil \
libmono-system-windows-\
forms4.0-cil
```

Build DSLogic

Next you need to prepare a placeholder to receive the latest DSLogic (Ver. 0.4) software. Create a directory by typing the following set of commands in a Terminal window:

```
$ cd ~ && mkdir dslogic && \
cd dslogic
```

Download the DSLogic software (DSLogic-v0.4.tar.gz) from http://bit.ly/1Fo4Gmk and move it to the directory created above. Expand the source tarball using the command:

```
$ tar xvzf DSLogic-v0.4.tar.gz
```

Next, build libusbx:

```
$ cd DSLogic-v0.4/libusbx-1.0.18/
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

Build the libsigrok4DSLogic library, which is a sigrok-compliant plugin library for the DSLogic oscilloscope/logic-analyzer device that provides the basic API for DSLogic hardware, using the commands:

```
$ cd ../libsigrok4DSLogic
$ ./autogen.sh
$ ./configure
     ...
     libsigrok configuration sum-
mary:

      - Package version (major.
minor.micro): 0.2.0
      - Library version
(current:revision:age): 1:2:0
      - Prefix: /usr/local
      - Building on: armv7l-un-
known-linux-gnueabihf
      - Building for: armv7l-un-
known-linux-gnueabihf

    Detected libraries:

      - glib-2.0 >= 2.32.0: yes
(2.40.2)
      - libzip >= 0.10: yes
(0.10.1)
      - libserialport >= 0.1.0:
no
      - libusb-1.0 >= 1.0.9: yes
(1.0.18)
      - libftdi >= 0.16: no
      - libudev >= 151: yes (204)
      - alsa >= 1.0: yes
(1.0.27.2)
      - check >= 0.9.4: yes
(0.9.10)

    Enabled hardware drivers:

      - demo.....................
.......  yes
      - DSLog-
ic.........................  yes

$ make
$ sudo make install
```

Build libsigrokdecode library:

```
$ cd ../
$ git clone git://sigrok.org/\
libsigrokdecode
$ cd libsigrokdecode
$ ./autogen.sh
$ ./configure
    ...
    libsigrokdecode configuration
summary:

      - Package version (major.
minor.micro): 0.3.0
      - Library version
(current:revision:age): 2:0:0
      - Prefix: /usr/local
      - Building on: armv7l-un-
known-linux-gnueabihf
      - Building for: armv7l-un-
known-linux-gnueabihf

    Detected libraries:

      - (REQUIRED) python >= 3.2:
yes (3.4)
      - (REQUIRED) glib-2.0 >=
2.24.0: yes (2.40.2)
      - (OPTIONAL) check >=
0.9.4: yes (0.9.10)

    Enabled features:
      - (OPTIONAL) Library unit
test framework support: yes

$ make
$ sudo make install

    Installing 45 protocol decod-
ers:

    swd pan1321 tca6408a jtag_
stm32 jtag i2c i2cdemux midi pwm
    ir_nec rgb_led_spi usb_sig-
nalling sdcard_spi dcf77 uart
    mx25lxx05d i2s rfm12 ds1307
lm75 spdif am230x onewire_link
    usb_packet ir_rc5 nunchuk mx-
c6225xu can nrf24l01
    guess_bitrate edid onewire_
network avr_isp z80 xfp parallel
    jitter tlc5620 maxim_ds28ea00
eeprom24xx rtc8564 mlx90614
    i2cfilter lpc spi
```

Build the DSLogic-gui application using the commands:

```
$ cd ../DSLogic-gui/
$ .  export BOOST_LIBRARYDIR=\
"/usr/lib/arm-linux-gnueabihf/"
$ cmake .
$ make
$ sudo make install
```

You may encounter some code errors such as:

```
...  no matching function for
call to 'min(double, qreal)'...
```

In such a case, change the invocation:

```
double delta = min((double)
max(pos - UpMargin, 0), \
get_view_rect().height());
```

to match the following line:

```
double delta = min((double)
max(pos - UpMargin, 0), \ (dou-
ble)get_view_rect().height());
```

Repeat the build process until it is successful, making the necessary changes in-between.

Electronics lab setup

Obtain the C-Tinkering kit and set it up using the instructions at http://bit.ly/1NsrlU9. Prepare a directory to receive the wiringPi source and build it:

```
$ cd ~ && mkdir tkit && cd tkit
$ git clone https://github.com/hardkernel/wiringPi
$ cd wiringPi
$ sudo ./build
```

Create a placeholder directory to receive the example code:

```
$ cd ~ && mkdir tkit-example/ && cd tkit-example/
```

Fetch example-lcd.c from above wiki and place it in the new working directory, then create a copy:

```
$ cp myexample-lcd myexample-lcd.c
```

The original file deals with numerous LEDs, but for the sake of the simple example in this article, it's advisable to update the logic to deal with just one LED. Apply the following patch to the copy (myexample-lcd.c) file to do so:

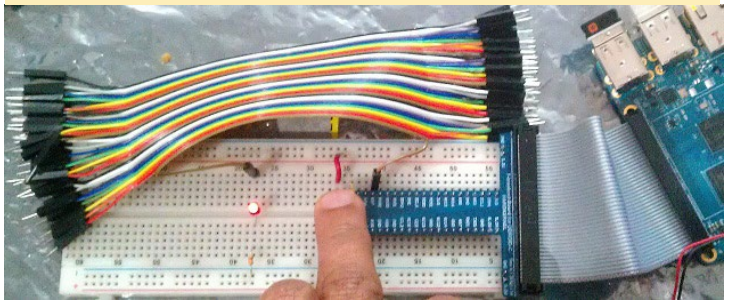
```
94a94,96
> #define OFF 0
> #define ON  1
> static unsigned long int ctr = 1;
97,108c100,123
<
<     // adc value read
<     if((adcValue = analogRead (PORT_ADC1)))       {
<             ledPos = (adcValue * MAX_LED_CNT * 1000)
/ 1024;
<             ledPos = (MAX_LED_CNT - (ledPos /
1000));
<     }
<     else
<             ledPos = 0;
<
<     //  LED Control
<     for(i = 0; i < MAX_LED_CNT; i++)      digi-
talWrite (ledPorts[i], 0); // LED All Clear
<     for(i = 0; i < ledPos;         i++)    digi-
talWrite (ledPorts[i], 1); // LED On
---
>     int tmp;
>
>     // adc value read
>     adcValue = analogRead (PORT_ADC1);
>     if(adcValue)    {
>             ledPos = (adcValue * MAX_LED_CNT *
1000) / 1024;
>     tmp = ledPos;
>             ledPos = (MAX_LED_CNT - (ledPos /
1000));
>             printf("%10lu: adc-value:%10d tmp:%3d
ledPos:%3d\n", ctr++, adcValue, tmp, ledPos);
>     }
>     else
>     {
>             ledPos = 0;
>             printf("%10lu: adc-value:%10d
ledPos:%3d\n", ctr++, adcValue, ledPos);
>     }
>
>     //  LED Control
>     for(i = 0; i < MAX_LED_CNT; i++)      digi-
talWrite (ledPorts[i], 0); // LED All Clear
>     if (adcValue < 15)
>     tmp=ON;
>     else
>     tmp=OFF;
>     for(i = 0; i < ledPos;         i++)    digi-
talWrite (ledPorts[i], tmp); // LED status depends on
light
>     usleep(10000);
141a157
>
```

In a new Terminal session, build the example and run it:

```
$ cd ~/tkit-example/
$ gcc -o myexample-led myexample-led.c \
-lwiringPi -lwiringPiDev -lpthread
$ sudo ./myexample-led
```

The modified tinkering-kit setup (hardware/software) using the GPIOX.BIT0 pin (marked #97) would look like Figure 1, reacting to the blocking of the light sensor:

Figure 1: Modified Tinkering kit setup



After exiting the example application, you are now set to use the oscilloscope or logic analyzer to study the electrical behavior on the pins (GPIO).

Test the setup

Using its user manual and wiki notes, attach the DSLogic oscilloscope to the LED like Figure 2.

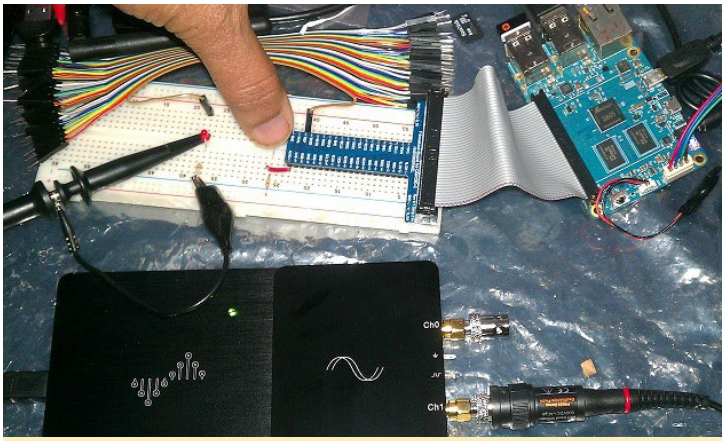


Figure 2: DSLogic oscilloscope setup

In a new Terminal session, launch the DSLogic gui application using the following commands:

```
$ cd ~/dslogic/DSLogic-v0.4/DSLogic-gui/
$ sudo ./DSLogic
```

In another Terminal session, launch the example application:

```
$ cd ~/tkit-example/
$ sudo ./myexample-led
```

With the LED OFF/LOW (0 volts), the baseline oscilloscope gui should look Figure 3.

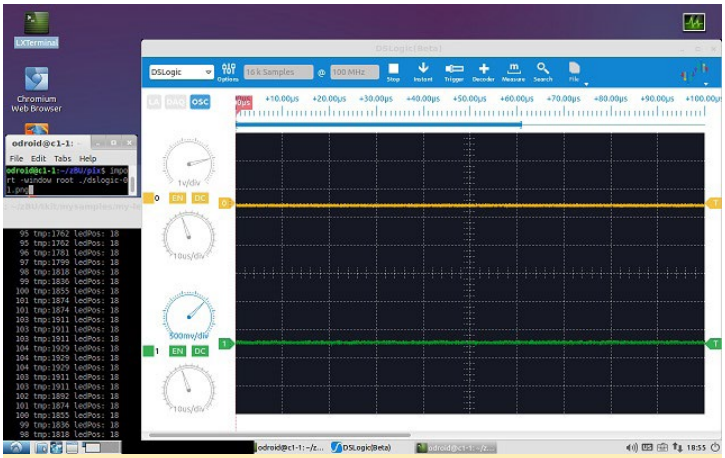


Figure 3: DSLogic baseline oscilloscope view

Refer to the user manual and wiki notes for the proper configuration. Next, slowly move a finger to block light to the sensor. The LED should light up. With the LED ON/HIGH (3.3 volts), the oscilloscope gui should look like Figure 4.

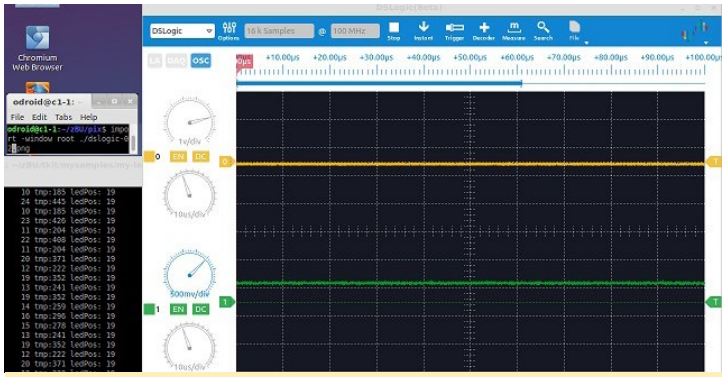


Figure 4: DSLogic 3.3 V oscilloscope view

Note the solid green graph-line at 3.3V and the dotted-green graph-line at 0V.

Exit the DSLogic-gui application. Referring again to the user-manual and wiki notes, setup the DSLogic device in the logic analyzer mode. It should look like Figure 5.

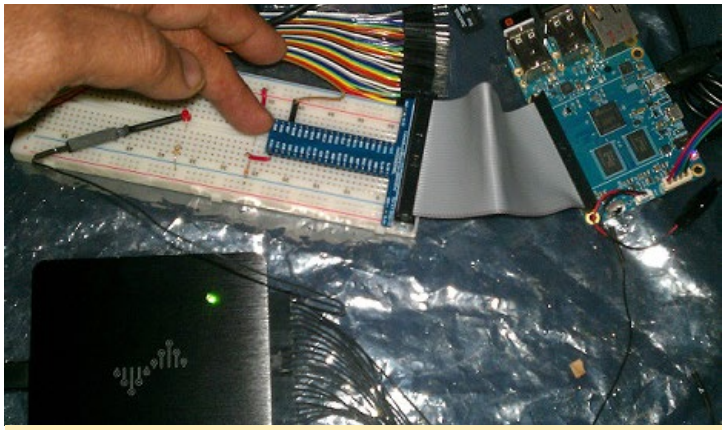
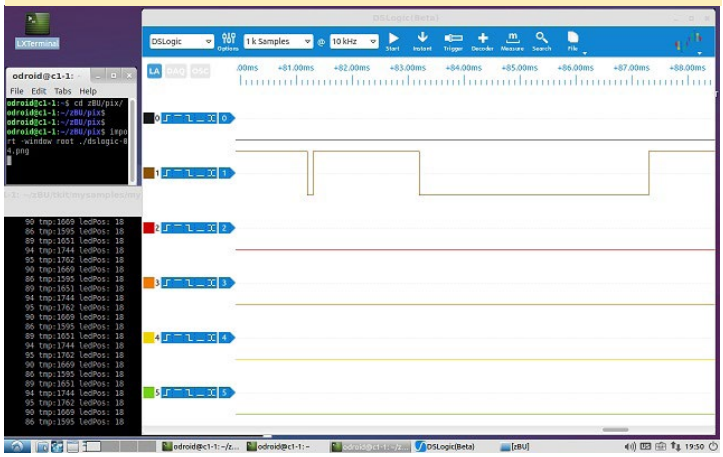


Figure 5: DSLogic logic analyzer setup

Re-launch the DSLogic-gui application. Once again, move a finger to block and unblock the light sensor a few times. You should see the logic levels change as shown in Figure 6.

Figure 6 - DSLogic logic analyzer view



Build Smartscope

Prepare a placeholder folder to receive the latest Smartscope software (version 0.0.7.0):

```
$ cd ~ && mkdir smartscope && cd smartscope
```

Download the Smartscope software from http://bit.ly/1BgUwTM and move it to the directory created above. Install the debian package:

```
$ sudo dpkg -i SmartScope-Linux-0-0-7-0.deb
```

Launch SmartScope-gui

Using its user manual and wiki notes, attach the Smartscope oscilloscope to the LED as shown in Figure 7.

Figure 7: Smartscope oscilloscope setup

In a new terminal session, launch the Smartscope gui application using the following command:

```
$ LIBGL_DEBUG=verbose sudo mono \
/opt/smartscope/SmartScope.exe
```

Note the use of the Mono library here. In another terminal session, launch the example application:

```
$ cd ~/tkit-example/
$ sudo ./myexample-led
```

With the LED OFF/LOW (0 volts), the baseline oscilloscope GUI should look like Figure 8.

Next, slowly move a finger to block light to the sensor. The LED should again light up. With the LED ON/HIGH (3.3V), the oscilloscope gui should look like Figure 9. Note the coincidence where the screenshot image capture coincided with the drop of voltage from 3.3V to 0V.

I will leave the use of the Smartscope logic analyzer as an exercise for the reader. I hope the reader sees the power of all the devices used here in creating a truly portable and powerful electronics lab!

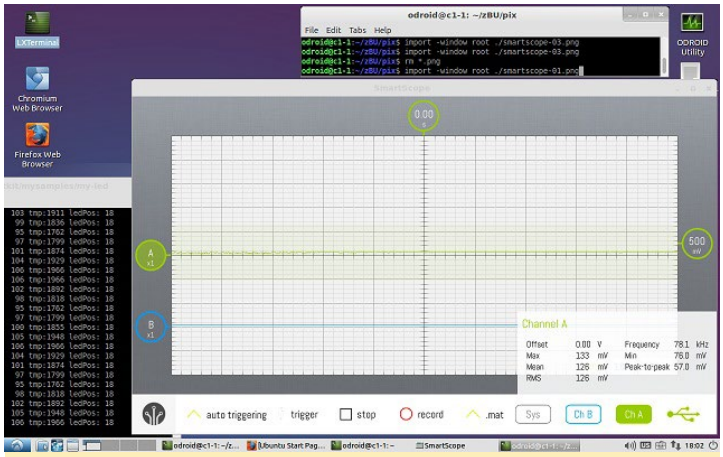


Figure 8: Smartscope baseline oscilloscope view

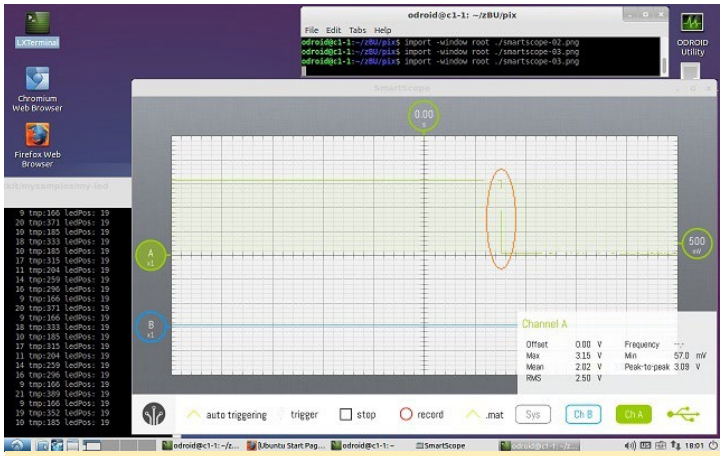


Figure 9: Smartscope 3.3 V oscilloscope view

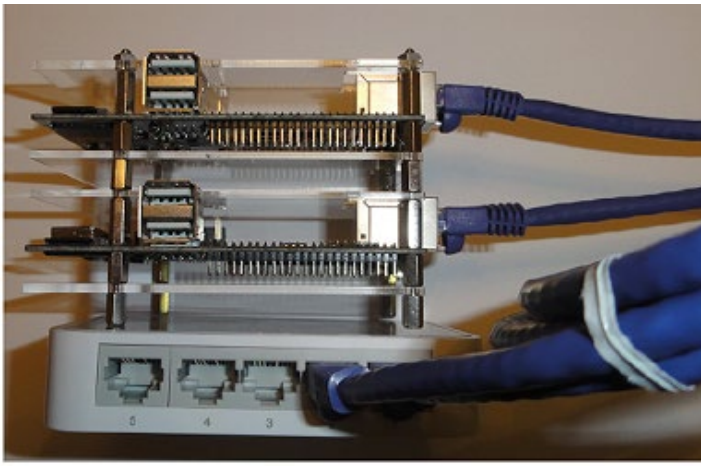
In a future article, I will explore the use of these oscilloscopes and logic analyzers under the Android operating system using an ODROID-VU, with its more intuitive multi-point touch control. For additional information or questions, please visit the original information sources at:

- http://bit.ly/1NsrlU9
- http://bit.ly/1BcMRqW
- http://bit.ly/1HapizJ
- http://bit.ly/1zNxyES
- http://bit.ly/1zYhM57
- http://bit.ly/1BXoiR3
- http://bit.ly/1BcN11m

HIGH PERFORMANCE COMPUTING

BUILDING AN AFFORDABLE AND PORTABLE CI OR U3 CLUSTER

by Dave Toth



As a professor, my goal was to create a high performance computing cluster that each of my students could purchase instead of a textbook for my parallel computing course, which means that the cluster needed to be inexpensive. This setup allows each student to do their work where and when they want, without messing up each other's efforts like they can on shared equipment. It also allows colleges with smaller budgets to teach a class on parallel computing.

I built my first educational cluster which contains 2 dual-core nodes (for a total of 4 cores) for about USD$200 in late 2013. If you shop carefully, you should be able to get the price down closer to USD$175. By using dual-core nodes, we can efficiently test both Message Passing Interface (MPI) and OpenMP code. For reference, a paper about this project called "A Portable Cluster for Each Student" was published in the Fourth NSF/TCPP Workshop on Parallel and Distributed Computing Education (EduPar-14) in May 2014. My next educational cluster cost just a little more, but had 2 quad-core nodes! I have just completed the third version, which brought the price down to about USD$150 while still using 2 quad-core nodes. I call these "half-shoebox clusters", since they fit in a box half the size of a typical shoebox.

One of the nicest things about these clusters is that I provide disk images for them, so you can just download the images and flash them onto microSD cards, pop them into the boards, power them on, and have your cluster completely configured! You don't have to install and configure MPI, set hostnames or IP addresses, create a machine file, or anything else.

Auxiliary equipment

- Two 1-foot Ethernet cables (USD$0.49 each). I spent a few more cents per cable to get a different color for fun. You might be able to get away with 6-inch cables instead, but I was afraid that was cutting it too close.

- One network switch (USD$9.99). If you want to take advantage of the Gigabit Ethernet on the ODROID-C1 cluster, you'll need a different switch and Ethernet cables, and a tiny Gigabit Ethernet switch costs around USD$20.

- One Sterilite plastic box (USD$0.97)

You may want some other items to go with your cluster to make it a bit more convenient to use. I found that it's nice to have a USB keyboard in case you don't want to SSH into the nodes, and an extra Ethernet cable so you can hook the cluster up to your router at home and SSH into that way. Please note that I have not configured any security settings for the cluster, so do all of this at your own risk!

Hardware

- Two ODROID boards with power adapters, either the U3 or C1 model, depending on your budget. These boards both have quad-core ARMv7 processors. I skipped getting the cases for these and instead stack them on top of each other with standoffs. I like the metal standoffs better, but the nylon ones are so cheap that I use them instead.

- Two 16GB microSD cards. I used class 10 cards since they recently came down in price so much. I recommend getting ones that come with the adapter so that you can plug them into a standard SD card slot in order to flash them. It takes about 20 minutes per card to flash the images to the cards.

- One micro-HDMI to HDMI cable. The nice thing about this connection is that you can hook the cluster up to any HDMI-capable TV or monitor that displays 720p or 1080p resolution.

Software

- A program to write the image onto the microSD cards. I use the free software called Win32DiskImager for Windows. If you are proficient with using the dd command on Linux, MacOS, or through Cygwin on Windows, you can use that, too.

- The images for the top node and the bottom node of the cluster, downloaded to the same machine being used to write them to the microSD cards. If you are using ODROID-U3 devices for the cluster, download the top node image from http://bit.ly/1wWZL6Z, and

the bottom node image from http://bit.ly/1BJPktk. If using ODROID-C1 devices, download the top node image from http://bit.ly/18S8X7K, and the bottom node image from http://bit.ly/1BJQ8hS.

Directions

1. Unzip the image files and write (don't drag and drop) the images onto your microSD cards using your preferred method mentioned above.
2. Insert the microSD cards into your ODROID boards.
3. Connect the ODROID boards to a power strip.
4. Power on the strip.
5. I typically connect a keyboard and monitor to each ODROID node, but you do not have to do so.
6. Log on to each board with username "odroid" and password "odroid".
7. In the directory in which the nodes start are 3 files: machinefile, hellompi.c, and a precompiled version of hellompi.c called hellompi. You can test the cluster by running the following command:

```
$ mpirun -n 8 -f machinefile \
./hellompi
```

If the output is 8 lines, each saying Hi. I'm processor x, rank y of 8, where x is one of two different names (odroidtop and odroidbottom for the U3, or c1top and c1bottom for the C1) and y is 0, 1, 2, 3, 4, 5, 6, or 7, then the system should be set up correctly. Note that the y values are not likely to be in order, which is fine, as long as they are all there.

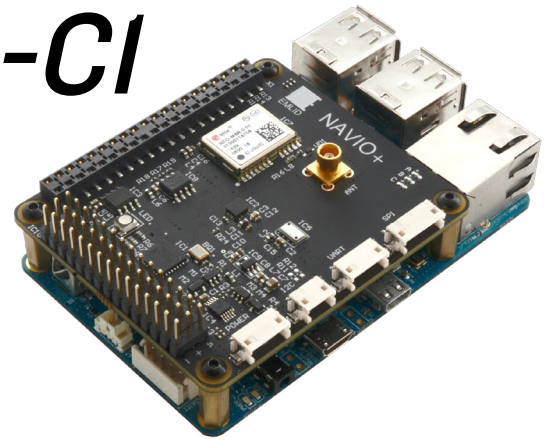
If you make a new program that will use MPI (or recompile the existing one), make sure to transfer the new binary/executable to the other node with a flash drive or by Secure Copy (SCP) before running it, or it won't work correctly!

For questions, comments and more information, please refer to the original post at http://bit.ly/1EOOZXi.

NAVIO+ FOR THE ODROID-C1

AUTOPILOTING YOUR DRONE

by Igor Vereninov



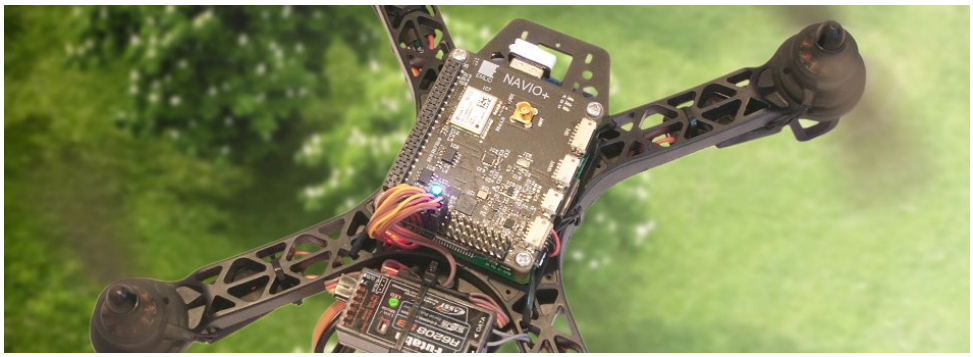
Navio+ is a HAT-compliant autopilot extension. The main goal of the project is to build a truly next-generation autopilot system that will run under Linux. With Navio+, you can make any flying or ground vehicle autonomous. All of the required sensors are located on the board, including 9DOF IMU, barometric pressure sensor, GPS, ADC and a PWM generator. It is fully compatible with the ODROID-C1, and may be purchased at http://www.emlid.com.

Creating the right hardware is not enough to make an autopilot. Software is the key. We have added support for APM, which is the most advanced open-source autopilot software available, which allows you to control copters, planes and rovers. The ground control station is very feature-rich and runs virtually on any device.

When the ODROID-C1 came out, we were very excited about its computational power and made porting software to it our priority. We have already developed support for the Navio+ and Odroid-C1 combination using APM. The porting was mostly straightforward, thanks to APM's HAL, and because we already had the drivers for Navio+. Initially, a couple of system components did not work as expected, but we are glad that the ODROID-C1 is maturing, and that the Hardkernel team is quickly adding new features and fixes.

Another important step towards a reliably flying autopilot is the RT_PREEMPT kernel. We are currently working on building a real-time kernel, which it will be made available to all ODROID-C1 users after testing. We look forward to the possibilities that are afforded by the incredible computational power of the C1. It makes running APM a breeze, and leaves a lot of power for other tasks, such as improved positioning algorithms, computer vision and others. There are several things left to implement and after that, our ODROID-C1 project will take flight!

To download the open-source Navio+ software, please visit our GitHub repository at http://bit.ly/18hK1oP.



LINUX GAMING

DISCOVERING THE WORLD OF NINTENDO DS(I) EMULATION

by Tobias Schaaf

ODROID devices can emulate many different retro systems. One of the more unique systems available for emulation is the Nintendo DS(i) through DeSmuME, which is a predecessor of the current Nintendo 3DS. It has a huge variety of games available for it, and some have very unique ways of game interaction via stylus, Dpad, camera and microphone.

The question is, how well do games with such input options work on the ODROID? What games do work on ODROIDs at all? What games are having issues? Can you play these games with a gamepad controller, or do you need a keyboard and mouse? I want to look at these things and see how well ODROIDs perform when emulating an NDS system.

General Overview

The first versions of DeSmuME on the ODROID were very slow, and were only able to play a few games that were strictly using 2D graphics. Even then, it depended on the game if it was working in any acceptable speed, and 3D games were unplayable.

Since the JIT compiler for ARM is now working, the speed went up, and the emulator runs very stable and fast on the ODROID-U3 and higher-end ODROID models. The speed of videos is perfect in all movies that I've seen. 2D games run at mostly full speed, and many 3D games run in good speed as well, but not all of them work. Also, the faster the ODROID, the faster the emulator. I noticed around 10-15 FPS more on the ODROID-XU3 while playing heavy games as compared to the U3.

Direct control through DPad

Some games can be directly controlled through gamepad interactions. One game that I enjoy playing is "Bleach – The 3rd Phantom", which is a RPG strategy game which can only be controlled by gamepad. The game uses the D-pad and action keys, and you can't control it with the stylus, which makes it perfect to control with a gamepad. The XBox360 controller, for example, feels very natural for this game, and since it's all 2D, the game runs really nicely

Bleach – The 3rd Phantom (below and right) is an RPG strategy game which can only be controlled by gamepad

on the ODROID, although there are a few elements with scrolling backgrounds that show degraded performance. However, since these scenes are rare, the game is really fun to play.

Basically, every NDS game that I tried that can be controlled with the gamepad is really fun to play on the ODROID. There are plenty of more NDS games that use the same input method, including Dragon Ball Z.

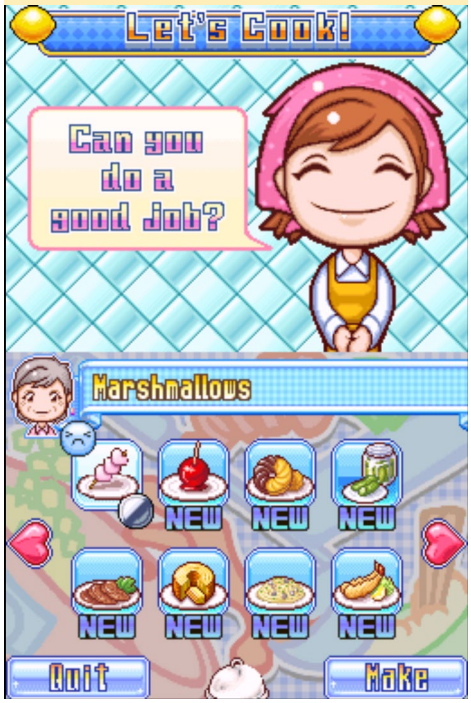
Dragon Ball Z – Attack of the Saiyans is another nice action RPG that runs very well on an NDS emulator

fine. It also uses just the Dpad as an input method.

Touch control

As said earlier, all games that use the Dpad as an input run very well on the ODROID, but what about the games that use the touchscreen itself? Is there a way to control the touchscreen with the gamepad? And if so, is the touchscreen working well enough to play with? What other options do you have? Well, let's start with some examples again involving precision and circle drawings or quick movement from one site to another. Can this be done with a gamepad? Technically, yes it can. You can activate the Pointer emulation in the core settings of Retroarch, and map it either to the left or right analog stick. This moves a small cursor over the screen, and you can use the R2 button to simulate a touch on the touchscreen. While this is feasible, the movements will quickly test your patience, since you can only react so fast with a gamepad by moving to different directions. Complicated movements, like swirling in circles or dragging and dropping with precision, are rather hard. In some games, this might actually be

Cooking Mama 3 (below and right) and Plants vs. Zombies (right) are two NDS games that use the touchscreen



acceptable when you only interact with the touchpad to make some selections for example, but in games like Plants vs. Zombies or Cooking Mama 3, you will really push the limits of the gamepad.

So, is there any other way to interact with the touchscreen? There is! You can use the mouse to move around the little white pointer and use the left mouse button to "touch". This works very nicely, but is still not as ideal as using the real touchscreen of the NDS. Playing games like Plants vs. Zombies with relatively easy interaction works fine with mouse, but trying to play Cooking

Mama 3 shows you that there are limitations. The game does generally work with a mouse and is playable, but you soon find out that it's really hard to do fast and precise movements, even with the mouse. Most of you probably have tried to draw a picture in Paint, or a similar drawing program, with the mouse. Have you ever tried drawing a circle with your mouse without using the circle tool? It will look very different from a real circle, and that's also what you can see in Cooking Mama 3 when using the mouse to control it.

Still, with some practice you probably can make do. All in all, touchscreen interaction is possible, but limited. In most cases you will be forced to use the mouse to control a little white cross cursor in order to interact with the touchscreen. This works acceptably, but has its flaws. Input through the mouse is limited, especially when it comes to precision and speed. Also, the little white cursor sometimes is really hard to see. On a white and very bright screen it's particularly hard to determine where to click.

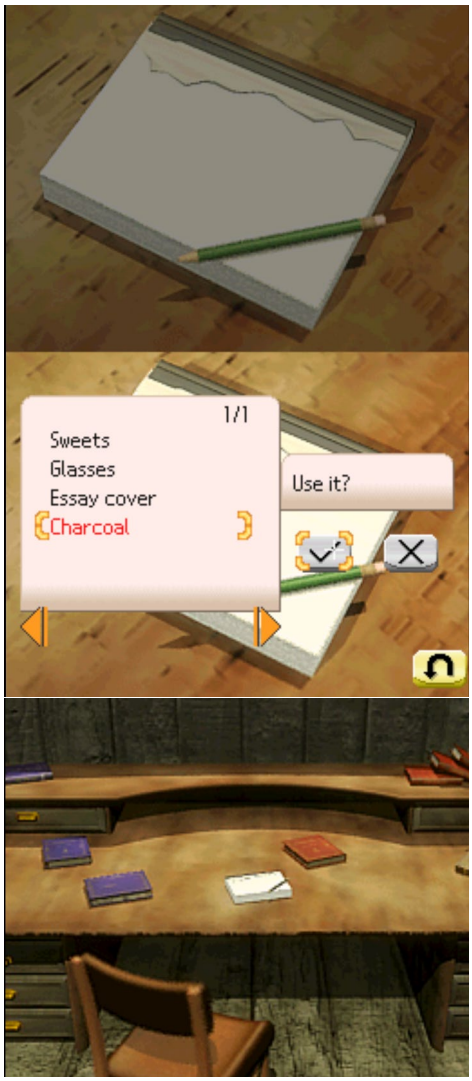
I'd like to mention some final words as to the two games mentioned above. Cooking Mama 3 and Plants vs. Zombies use only 2D graphics and because of this, are mostly playable in full speed. However, in Plants vs. Zombies, you have to deal with slow downs if there are many items and/or enemies on the screen.

Other inputs

The Dpad and touchscreen are not the only methods of input on the NDS. In fact, it's packed full with little gimmicks like a microphone and a camera, and some games even use the fact that you can fold and close your NDS. These are rather unique ways of inputs, and I want to see if it's possible to do this with the emulator as well.

A very interesting game which uses many different input methods is Another Code: Two Memories (Trace Memory

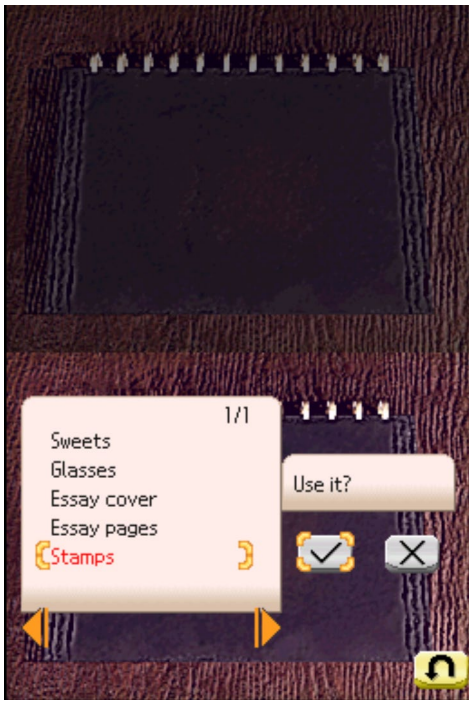
In Another Code - Two Memories (above, below and upper right), you have to solve many riddles the different input methods of the NDS



in NA) to solve many riddles using different items. You use the different input methods of the NDS such as blowing into the microphone to clear away dust on a picture, and closing and opening the NDS a couple of times in order to "stamp" a picture. You have to interact with the touchscreen by scratching, turning and pointing at things. It's very unique, and there aren't many games that made much use of these alternative input methods.

The question is, can the ODROID do this as well? Well, there are two scenes in this game that can give an answer to that: Stamping in a picture book and blowing at a dusty painting, which are two of the special input methods of Another Code – Two Memories. In order to stamp the picture, you need to

Stamping in a picture book and blowing at a dusty painting, two of the special input methods of Another Code – Two Memories (above, below and next page)



close and open your NDS, which is apparently somewhat hard to mimic on an emulator. Still, there were some other games that also required you to close and open the lid a couple of times in order to progress in certain games. It was rare, but still used, and Another Code – Two Memories was really good at using all of the options that the NDS had to offer.

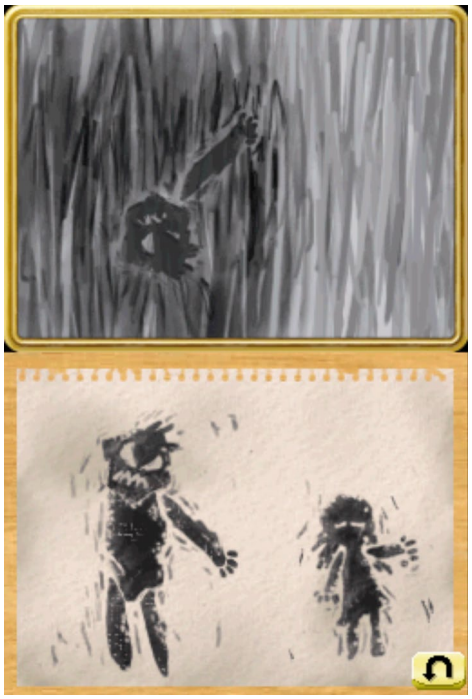
The NDS had a built-in microphone, and there were quite a few games that could make use of it. But most of the games did not require a special input, but rather reacted to any kind of "noise", which means that blowing or scratching on the microphone mostly worked on all games that required microphone input. Some emulators actually used that fact, and included a "white noise" emulation to simulate microphone input.

In Retroarch and Libretro, the DeSmuME core has mapped the L2 button to opening and closing the lid, so by simply pressing the L2 button, you closed the lid, and pressing it again reopened the lid. That way, you can click the L2 button a couple of times and pass the stamping puzzle.

Although the stamping action was easy to resolve, what about the blowing puzzle? Well, it turned out that this was not as easy to address. The version of DeSmuME libretro core that I was using did not offer such a feature. I looked up the DeSmuME Libretro project, and found a relevant bug report which apparently was solved in early January. Since my core was from late December, this fix was not included. I checked out the newest version of the core, then compiled it along with the newest version of Retroarch, and tried again. While the microphone was still not working out of the box, the gaming experience improved a lot. I checked the code and found that the white noise simulator was mapped to the L3 button. So I tried this as well.

This was finally working, and with that I was able to use all of the features that the NDS had to offer, except for the camera, but I couldn't remember any game that actually used this. Another

Solving the stamp puzzle (below and right), normally done by closing and opening the lid, can be performed by pressing the L2 button on the gamepad



Code – Two Memories is a very unique game that uses many features of the NDS to solve all kinds of puzzles. The story is very interesting and is fun to follow. The game uses a mixture of 3D and 2D elements. While most puzzles and talking to people are in 2D, walking on the map is done in 3D, so the game varies in the smoothness of its rendering. While 2D elements are running full speed, the 3D elements sometimes drop to 40 FPS and below, which can be a little annoying since the sound starts to stutter a little. But, since you have all the time that you want in the game, and there are no you want in the game, and there are no action scenes that requires fast walking or anything similar, it doesn't interfere with the overall gaming experience.

Conclusion

NDS emulation on the ODROID works well because there's a mapping for everything, although I wasn't able to to directly use the microphone that I connected to it. Still, this is an amazing piece of work that allows you to play all of the games available for the NDS and DSi. Although all games are working, not all of them are performing at 100% full speed, and it really depends on the game whether you have good experience.

However, since there are a couple

CLASH OF CLANS

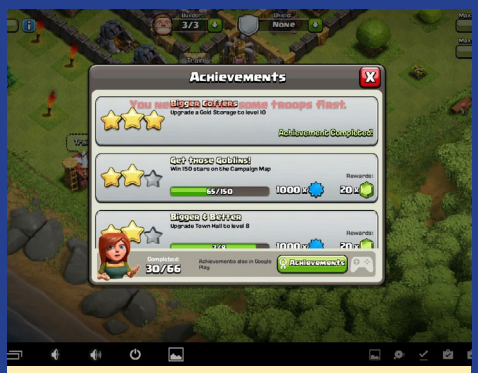
EPIC BATTLES ON A BIG SCREEN

by Jeremy Leesmann

Clash of Clans brings friends together in epic battles. This game takes a lot of thought and strategy to build a well-defended base, while also developing a strong offense to win the Clan Wars. If you want to lose lots of free time (in a good way), then you'll enjoy Clash of Clans.

https://play.google.com/store/apps/details?id=com.supercell.clashofclans&hl=en

Build up your base, then fend off the invaders who want to destroy it



Clash of Clans has achievements that you can unlock as the game progresses, giving you rewards like experience and gems

More views of the stamp puzzle, which uses a unique lid action that was only implemented on the NDS system. The authors of the emulator had to come up with a way to simulate a non-traditional way of interacting with the device.

thousand games for the NDS and DSi, it should offer everyone some that are worth playing. Also, the better your ODROID is, the better your gaming experience you will have. This means that the ODROID-XU3 outperforms the U3, which outperforms the C1 when it comes to fluidity and performance. NDS on the ODROID can be quite fun, and there are really some lovely games for the NDS, and since my DSi XXL has some issues by now, I really enjoy playing my DSi games on the ODROID.

The original Nintendo DSi came in a variety of colors, including rainbow!

ODAMEX

PLAY DOOM IN HIGH DEFINITION WIDESCREEN WITH MULTIPLAYER

by Jeremy Kenney

Doom has been with us for many years now, and has amassed a very large audience. The game is still being played today. Even the prize-winning Wolfenstein 3D has had a recent revival in a form of Total Conversion, which uses the Doom engine.

In my last article, I presented a tutorial on how to compile SDL Doom. But that's not the ultimate experience of Doom: you want more damage, more monster slaying and more mazes with friends! Odamex is now available for the ODROID, which features full-speed gameplay, fully customizable video modes including widescreen with 720 to 1080P, the ability to easily setup and find servers, MIDI playback, and more. This high-quality source port of Doom will give you non-stop action and millions of compatible wads.

Wads made for other source ports may require additional files and/or wads, or may not even work at all. However, this is to be expected, since some other source ports have changed the original source code to something more "modern". The Odamex port gives you that original game feel, but also delivers rich advanced options. In a few steps you can be up and running Multiplayer Doom quickly.

Requirements

What you will require is the basic libsdl 1.2 or greater in order to make this program work. If you happen to know a library that enables you to have a "MIDI port" open for Midi playback, such as freepats, install that as well. The MIDI library is optional, but allows you to have in-game music if you wish.

Now that you have all of your libraries downloaded and installed, download the Odamex package from http://bit.ly/1DXz2MX, as well as the server package at http://bit.ly/18SdfMl. Install it as you would any other package in Linux by opening up a Terminal window and typing:

```
$ sudo dpkg -i \
packagenamehere.deb
```

After installing both the client and server parts of this software, you now have "odamex" and "odasrv" available as runnable commands in Terminal, and if you chose to download the GUI, you also have "odalaunch". Odamex is the client, odasrv is the server start command, and odalaunch is the launcher. Now we can proceed to configure the game.

Installing wads

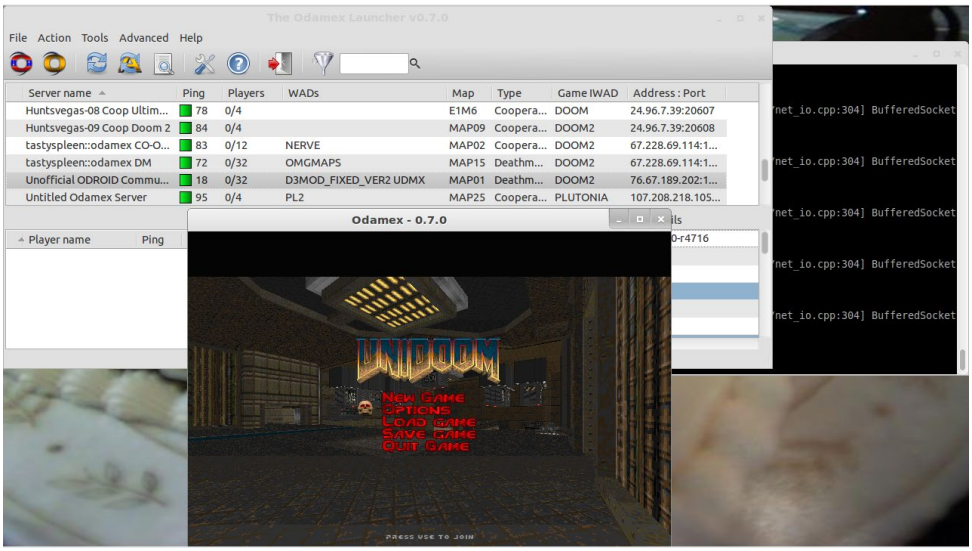
To save some money, download the shareware wad from http://bit.ly/17TeidM, since original copies of the game are USD$5 or more. However, if you have purchased original floppies or CDs of the game, you can copy the DOOM.WAD and DOOM2.WAD over to "/usr/local/share/odamex" so that the game can find the wads. If you wish to change the directory for your wads, when you run odamex, type in the following to run the game:

```
-waddir /path/to/directory/here
```

After running the game, you may want to optimize it for HD video mode, and enable a couple of other options. Go into the "Options" menu and select "Set Video Modes", "Fullscreen on", and "1280 x 720". Note that when using any resolution higher than 720p, you will need to enable horizontal and/or vertical Detail Modes in order to smooth the rendering. You can cap or uncap the frames if you want, but in 720P or higher, you won't get uncapped frames on an ODROID-U3.

Multiplayer action

If you want to play in multiplayer mode, pressing F8 is necessary order to see other players' messages, and to enable the message notification sound. Messages also can seen in the console by pressing the ~ key. Next, you will need to configure your server if you wish to host a game. There's a folder of samples



for multiple game types located in the folder "/usr/local/share/odamex/" called "config-samples", which should get you started on some basic servers.

Grab one of the server samples and copy it to the .odamex directory in your home folder. Rename the copied sample file to "odasrv.cfg", then edit it using your favorite text editing software. Using the text editor, you can configure the email, hostname, message of the day (MOTD), which appears on every first connection to the server for any player, the website if you have one, and your wad directory path. If you scroll down some more, you can configure the Game Mode that you have chosen by specifying the time limit, frag limit, gravity, compatibility options and much more.

You have now successfully configured your server to play online with your friends. If you want to see a master server list, you can type "odalaunch" in Terminal, which will give you an error at first. Just untick the box that says "show dialog next time" and press continue. If the program doesn't display any servers after a minute or two, click on "Action", then "Get List". Now, you can configure odalaunch by clicking "File", then "Settings" under the tab labelled File Locations. Click on "Odamex Path" and choose "Other". Find your executable located in /usr/local/bin, then click "Open", and select the folder icon in order to add the folders where you store your wads, such as /home/odroid/Downloads/Wads/.

After you choose a server, the client will automatically download and install the wads if necessary. The wads will install one-by-one, so you need to type "Reconnect" after each wad has been downloaded and installed. You may also want to go to Player Setup to change your nickname and DoomGuy Color. There is an available Shareware Server at 74.207.250.98:10668 if you don't own a copy of Doom, and I've opened an unofficial public server for the ODROID Community for Doom 2 copy owners at 76.67.189.202:10666. Note that when you type connect and enter the IP, you can leave the port blank because it will default to 10666. My server is not guaranteed to be open 24/7 due to normal shutdowns and maintenance, but if you receive an unavailable message, the server should be up in less than 2 hours. Dooming online has never been better than with Odamex teamed up with ODROID. The 1990s 3D fun experience never stops!

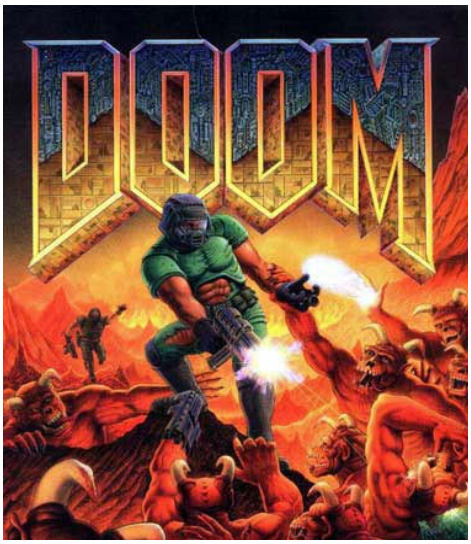
Magazine bonus

As a bonus, I natively ported the game "Commander Keen Dreams and The Catacombs Series" to the ODROID for the benefit of people reading this article. You can have access to exclusive content as a way to thank everyone who enjoys reading ODROID Magazine!

First, download the blobs for the game from http://bit.ly/1DXC2Zr. If you don't own your own copy of "Commander Keen Dreams and The Catacombs Series", then there are executables included in order to run the shareware version of the game. After downloading the zipped file, extract it to a fresh folder, then take any of the copies that you own (or use the shareware version) and associate all of the files with the right executable. The legend is as follows :

```
refcat3d-100 = Version 1.0 Catacombs 3D
refcat3d-122 = Version 1.22 Catacombs 3D
refcatabyss-113 = Version 1.13 Catacombs Abyss
refcatabyss-124 = Version 1.24 Catacombs Abyss
refcatapoc-101 = Version 1.01 Catacombs Apocalypse
refcatarm-102 = Version 1.02 Catacombs Armageddon
refkdreams-cga105 = Keen Dreams CGA Version 1.05
refkdreams-reg193 = Registered Version Keen Dreams 1.93
refkdreams-shar113 = Shareware Keen Dreams Version 1.13
refkdreams-shar120 = Shareware Keen Dreams Version 1.20
```

You can find more information about shareware Catacombs in the included readme file, along with other great tips.



MAP DPAD ON XBOX 360 CONTROLLERS IN ANDROID

GET FULL USE OF YOUR GAMEPAD

by @seismograf

Although the Xbox 360 wired and wireless controllers work out of the box with all of Hardkernel's Android releases, the DPad is not mapped by default in the controller files, which makes some emulator experiences difficult. By following the instructions in this article, you can enabled the DPad controls and enabled you to map them in PPSSPP and many other popular Android emulator applications.

To begin, download the pre-built .kl files from http://bit.ly/1aKX4Rq, and put the files somewhere convenient, such as the /sdcard0/download folder. The file Vendor_045e_Product_0719.kl is meant for use with the Xbox 360 wireless controller, and the Vendor_045e_Product_0291.kl file is intended for the wireless version.

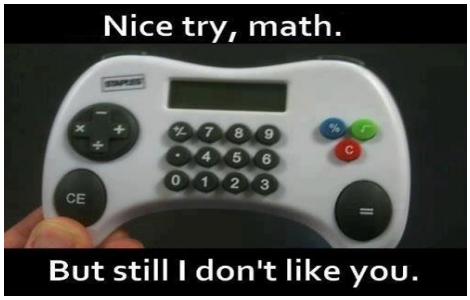
Next, open a terminal session using the Terminal app, and type the following set of commands:

```
$ su
$ mount -o remount,rw /system
$ cp /storage/sdcard0/download/\
Vendor_045e_Product_0719.kl \
 /system/usr/keylayout/
$ cp /storage/sdcard0/download/\
Vendor_045e_Product_0291.kl \
 /system/usr/keylayout/
$ cd /system/usr/keylayout/
$ chmod 644 \
Vendor_045e_Product_0719.kl
$ chmod 644 \
Vendor_045e_Product_0291.kl
$ exit
```

Alternatively, you can add the following lines to either /system/usr/keylayouts/Vendor_045e_Product_0719.kl or /system/usr/keylayouts/Vendor_045e_Product_0291.kl

```
key 704 DPAD_LEFT
key 705 DPAD_RIGHT
key 706 DPAD_UP
key 707 DPAD_DOWN
```

Thanks to the forum members at xda-dev, who originally posted the solution at http://bit.ly/1BFRH0v. For questions and comments, or to suggest improvements, please visit the original post at http://bit.ly/1BFRPwI.



BOOM! TANKS

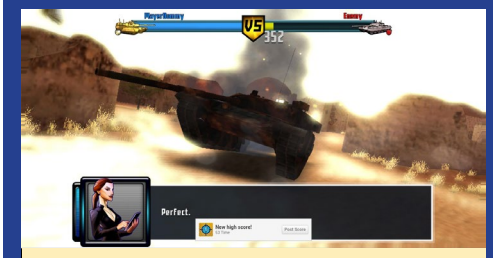
SIMPLE KILL OR BE KILLED TANK BATTLE

by Jeremy Leesmann

In Boom! Tanks you only have to worry about being the gunner, no driving the tank. It's a great battle game that isn't difficult to master. Build up your tank and blow the other guys away before they blow you away.

https://play.google.com/store/apps/details?id=com.reliancegames.android.boomtanks&hl=en

Boom! Tanks lets you achieve your dream of blowing up everything you see



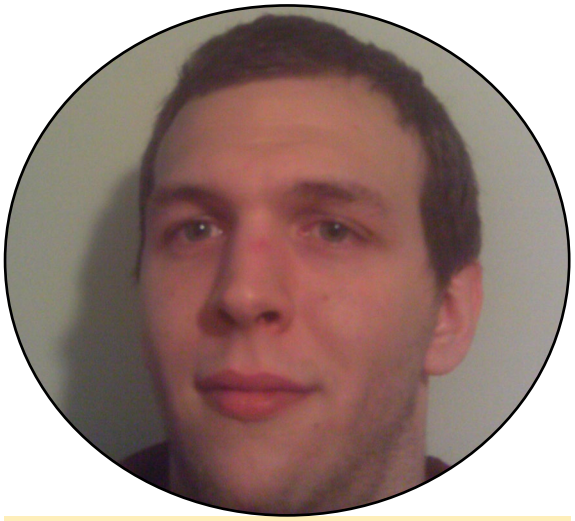
The fast action is simple to learn, yet difficult to master

Android gaming has come a long way, and Boom! Tanks exemplifies the best of the shoot-em-up genre

MEET AN ODROIDIAN

JEREMY KENNEY (@CARTRIDGE) OUR RETRO GAMING WIZARD

edited by Rob Roy



Jeremy Kinney produces lots of ODROID software

Please tell us a little about yourself.

I'm 23 years old, and just bought my own house in Canada. My native language is French and I'm also a Sega fan. Not that Nintendo did not make great games, because there are great games in the Nintendo Library, but I'm just not a fan of the Nintendo console. I'm the kind of guy who likes the traditional way of playing multi-player games, with friends sitting next to each other enjoying a good game cartridge. That's why I call myself Cartridge on the forums!

How did you get started with computers?

I was about 3 when my uncle gave a computer to my parents, which contained a copy of Wolfenstein that I never stopped playing. I then received a demo copy of some Internet BBS software. The name of the BBS on the CD slips my mind, but it also contained a Doom shareware copy and a game called Jetpack. These two games were the gateway to my computer experience today. I was so amazed that I couldn't stop getting more and more demos to see the art style being used, and what the gameplay consisted of. I tried multiple times to create my own games, animations and music. Everything I know now is self taught. I could never get someone to mentor me, since the town I lived it was almost Amish. I was a huge demo player and played everything that I could find. I then stumbled upon some Windows books, as I knew nothing about Windows at the time. I read it all the way through and immediately taught myself bash scripting, which is very basic and easy to learn. Bash scripts were the gateway to Linux, as I never viewed Windows as "something good". With the endless drivers, Blue Screens of Death, along with the many errors for simple problems, Windows was a nightmare.

What drew you to the ODROID platform?

I was on a search for something that could run Linux that I could build myself. I had never even heard the words "ARM processors" at the time. I kept spending money to build something better each year. This led me to financial problems and brought my builds to a stop. I then stumbled upon the Raspberry Pi boards, but heard bad things about them, which led me in a different direction. The speed of the processor and the RAM capacity, along with the yellow AV/OUT (since HDMI was not on the RPI back then) kept me searching for more information. Then, someone in a certain forum mentioned the ODROID family of computers. I immediately found what I now own today: an ODROID-U2 which is about to reach 3 years old now, and runs stable as a rock. The quad-core processor, the Mali GPU, and the RAM capacity options were all valuable assets, and the U2 finally made computers cheaper to buy a new board every now and then, instead of always buying parts every year to keep up with all of the newer games and files that come out.

Which ODROID is your favorite?

My favorite ODROID has to be the U2. I like to say that ODROIDs have Blast Processing as a fun 1990s joke. I've also had the chance to try a U3, but I'm more impressed by the form factor of the U2. It sits well, and the heatsink makes the job even more easier as I don't need to have a case for it.

Your ODROID software ports are very popular! How did you become so proficient in Linux?

For this, I have to thank @meveric at the Hardkernel forums. I had a basic idea because of Bash Scripting (DOS) but I learned how to port and modify code thanks to him. He showed me in detail how it's done, and what to do when an error has occurred while compiling. Of course, Google as well gave me some answers, but the great community at the Hardkernel forums has enhanced my capabilities. Compiling different programs means bringing in different libraries, which always means exerting more effort and time into making them. You get to learn a lot from other peoples' source code.

What hobbies and interests do you have apart from computers?

I like to play golf every now and then, and bicycling is a passion of mine. I like to walk around parks. Art is also something I love as much as I love art-

work in games. It's not about how graphic it is, but what it represents and what it means.

Are you involved with any other computer projects unrelated to the ODROID?

Not necessarily, just numerous attempts at having a working website for storage and web-based fun things. I'm also trying to find beta software for a certain preservation website. I have a small project using Windows 98 over at TheIsoZone forums (www.theisozone.com). The project is about optimizing your current Windows 98SE to run most programs of today, which includes abilities for HTML5, Flash, MSVCRT, kernel extensions, and much more. You can find me in TheIsoZone under the same name of Cartridge.

Jeremy's equipment show us that he is a pure breed from the era of 90s computing

What type of hardware innovations would you like to see for future Hardkernel boards?

It's hard to think of something better when you already have it. But then there's dreams. Why is the Raspberry Pi popular despite its speed and RAM capacity? If you use it right, you can achieve anything. Because of this reason, too fast or too slow is not a factor for me. It's more a question of what it's capable of, and ODROID is already capable of a lot more than what some people may think. So for this, it would be to optimize what we already have and maybe someone with a bit of creativity can put a chip alongside the ARM CPU to process x86 code. This may be impossible of course. But I can dream about it.

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

Doom is a great game to start programming with. But it's a game!? Yes, indeed it's a game, but also the game is highly modifiable and it uses assembly language for its coding. If you get a Doom Editor, you can scroll through the wads and see some coding. The coding used in Doom mimics the same structure as scripting with the C language. It's a good gateway to learn coding, and also there's a lot of terms used in Doom that you see in almost all programming languages.

I can get you started in programming Doom by getting you to download "XWE Doom Editor". With this program you can code, input textures, patches, sprites, sound and music. If you download a modified wad (because mods contain more scripting than the original wads) you can see how the guns are scripted and how they work. Now you can go to Zdoom Wiki (http://bit.ly/1EiBSgt) and find a cheatsheet of commands for use in scripting. There are many great mods only using ASM coding. For example, if you download the wad from http://bit.ly/1LsvDK8, there's a lot to learn in just this mod. It uses what is already in the game to create a new version with scripting. Of course you can get around this to make it easier, make new sprites, new monsters, or new guns, but that only changes the appearance. The above wad I mentioned shows just what you can do by scripting only. Not only are you learning to script with Doom but you're also modding it which is a fun way to learn! Remember that the Zdoom wiki is always available for help.