Install Linux on an External Disk • eMMC Recovery • GPIO Shutdown

ODROID Magazine

Year Two
Issue #16
Apr 2015

Kali Linux

ODROID's advanced penetration testing distribution

Build your own Ambilight

- Automate your house with OpenHAB
- ODROID-C1 triple boot
- Android Dev: Low Memory Killer Daemon
- Play the original Warcraft series on Linux

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

Kali Linux is one of our favorite distributions, as it includes a wide variety of security penetration testing software. Offensive Security, the developers of Kali, are ODROID fans, and have released a version for every model so far. Kali on the ODROID can be the basis for a lucrative business, offering something that every company needs: peace of mind. Learn how to install Kali Linux on the ODROID-C1,and you may be able to offer affordable security testing with minimum startup costs.

Also featured is another amazing technical guide by Venkat showing us how to make our homes smarter using OpenHAB, as well as a tutorial by Marian on Ambilight, which makes it more fun to watch videos and listen to music. Tobias brings us a special on running the original Warcraft series on modern hardware, Nanik looks at the Android Low Memory Killer Daemon, we learn how to triple-boot on an ODROID-C1 with Android, Linux and OpenELEC, and much more!

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

HARDKERNEL

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

Dusted off all his Dungeons and Dragons books to get inspired for the Kali Linux article and cover. After that, got to watch a marathon of Star Trek:TNG for the OpenHAB article. And at last, he came to be "THE ONLY ONE!"

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

BUILDING KALI LINUX

TAKE YOUR ODROID TO THE EDGE WITH THE MOST ADVANCED PENETRATION TESTING DISTRIBUTION EVER CREATED

by @tmpuser and @yhfudev

According to its developers, Kali Linux is the most advanced and versatile penetration testing distribution available. It contains a whole set of amazing features bundled as an all-in-one security distribution, geared at streamlining the penetration testing experience. The ODROID family of single-board computers is a favorite of Offensive Security, the publishers of Kali Linux, and they have released their distribution for every ODROID model so far, including the U3 and XU3.

Several volunteers on the ODROID forums have come up with their own method of bringing Kali to the ODROID-C1 by creating a step-by-step guide to installing Kali on a minimal Debian, ARCHLinux or Ubuntu distribution. The first section presents a manual method for building Kali, and the second section describes a single-click script which automatically installs it.

Manual method

To begin, boot up a copy of Ubuntu or Debian, and type the following set of commands into a Terminal window or SSH session. Note that this is not a shell script, so everything should be entered by hand.

```
$ sudo su -
# sudo apt-get install qemu-user-static binfmt-support lzop libncurses5-
dev dosfstools

# export architecture="armhf"
# export distro="kali"
# export path_rootfs="/mnt"
# export INSTALL_MIRROR=http.kali.org
# export INSTALL_SECURITY=security.kali.org
# PACKAGES_ARMHF="ssh,vim-tiny,nano,locales,dialog,xz-utils"
# PACKAGES_BASE="binutils console-common initramfs-tools sudo parted
e2fsprogs usbutils uboot-mkimage u-boot-tools ntfs-3g lsb-release module-
init-tools fbset ntpdate"
# PACKAGES_DESKTOP="kali-defaults gnome-core kali-root-login desktop-
base"
# PACKAGES_TOOLS="curl make bc lsof less git nmap ethtool unrar-free ca-
bextract unace unrar zip unzip p7zip p7zip-full p7zip-rar uudeview mpack
arj ncompress apt-file python-software-properties"
# PACKAGES_SERVICES="ntp"
# PACKAGES_EXTRAS="iceweasel iceweasel-l10n-en-en myspell-en wpasuppli-
cant gparted gnome-tweak-tool xserver-xorg-video-fbdev xserver-xorg-vid-
eo-mali mali-fbdev aml-libs file-roller totem totem-plugins totem-mozilla
rhythmbox deb-multimedia-keyring gedit"
# export PACKAGES="${PACKAGES_BASE} ${PACKAGES_DESKTOP} ${PACKAGES_TOOLS}
${PACKAGES_SERVICES} ${PACKAGES_EXTRAS}"

# mkdir -p ..path_rootfs/rootfs
```

```
# export pathlocal="..path_rootfs/rootfs"
# cd $pathlocal
# dd if=/dev/zero of=${pathlocal}/rootfs.img bs=1M count=4096
# mkfs.ext4 -F -O ^has_journal -E stride=2,stripe-width=1024 -b 4096 -L
rootfs -U e139ce78-9841-40fe-8823-96a304a09859 ${pathlocal}/rootfs.img
# mount -o loop ${pathlocal}/rootfs.img ${path_rootfs}

# debootstrap --verbose --foreign --arch ${architecture}
--include=${PACKAGES_ARMHF} ${distro} ${path_rootfs} "http://${INSTALL_
MIRROR}/${distro}"
# cp /usr/bin/qemu-arm-static ${path_rootfs}/usr/bin/
# DEBIAN_FRONTEND=noninteractive DEBCONF_NONINTERACTIVE_SEEN=true LC_
ALL=C LANGUAGE=C LANG=C chroot ${path_rootfs} /debootstrap/debootstrap
--second-stage -cpu cortex-a9
# DEBIAN_FRONTEND=noninteractive DEBCONF_NONINTERACTIVE_SEEN=true LC_
ALL=C LANGUAGE=C LANG=C chroot ${path_rootfs} dpkg --configure -a
# mount -t sysfs none ${path_rootfs}/sys && mount -t proc proc ${path_
rootfs}/proc && mount -B /dev ${path_rootfs}/dev && mount -B /dev/pts
${path_rootfs}/dev/pts && sync

# cat << EOF > ${path_rootfs}/etc/apt/sources.list
deb http://${ INSTALL_MIRROR }/${distro} ${distro} main contrib non-free
deb http://${ INSTALL_SECURITY }/${distro}-security ${distro}/updates
main contrib non-free
deb http://${ INSTALL_MIRROR }/${distro} ${distro}-proposed-updates main
non-free contrib
deb http://www.deb-multimedia.org wheezy main non-free
deb-src http://www.deb-multimedia.org wheezy main non-free
deb http://deb.odroid.in/c1/ trusty main
deb http://deb.odroid.in/ trusty main
EOF
# sed -i "s/^#\ en_EN/en_EN/g" ${path_rootfs}/etc/locale.gen
# echo "linux" > ${path_rootfs}/etc/hostname
# echo -e "nameserver\t8.8.8.8"> ${path_rootfs}/etc/resolv.conf
# echo -e '127.0.0.1\tlocalhost' > ${path_rootfs}/etc/hosts && echo -e
'127.0.1.1\tlinux' >> ${path_rootfs}/etc/hosts
# cat << EOF > ${path_rootfs}/etc/network/interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
EOF


# echo "KERNEL==\"mali\",SUBSYSTEM==\"misc\",MODE=\"0777\",GROUP=\"vid
eo\"" > ${path_rootfs}/etc/udev/rules.d/10-odroid_mali.rules && echo "KER
NEL==\"ump\",SUBSYSTEM==\"ump\",MODE=\"0777\",GROUP=\"video\"" >> ${path_
rootfs}/etc/udev/rules.d/10-odroid_mali.rules
# echo "KERNEL==\"ttySAC0\", SYMLINK+=\"ttyACM99\"" > ${path_rootfs}/etc/
udev/rules.d/10-odroid-shield.rules
# echo "KERNEL==\"fb1\", SYMLINK+=\"fb6\"" > ${path_rootfs}/etc/udev/
rules.d/50-odroid-hdmi.rules
# echo "KERNEL==\"CEC\", MODE=\"0777\"" > ${path_rootfs}/etc/udev/
rules.d/60-odroid-cec.rules
# echo "KERNEL==\"amstream*\",SUBSYSTEM==\"amstream-
dev\",MODE=\"0666\",GROUP=\"video\"" > ${path_rootfs}/etc/udev/
rules.d/10-odroid_am.rules && echo "KERNEL==\"amvideo*\",SUBSYSTEM==
\"video\",MODE=\"0666\",GROUP=\"video\"" >> ${path_rootfs}/etc/udev/
rules.d/10-odroid_am.rules
```

```
# cat << EOF > ${path_rootfs}/etc/fstab
LABEL=rootfs / ext4 errors=remount-ro,noatime,nodiratime 0 1
LABEL=BOOT /boot vfat defaults,ro,owner,flush,umask=000 0 0
tmpfs /tmp tmpfs nodev,nosuid,mode=1777 0 0
EOF

# cp -r ..path..modules_firmware_odroid/firmware/* ${path_rootfs}/lib/firmware/
# cp -r ..path..modules_firmware_odroid/modules ${path_rootfs}/lib/
# sed -i "/exit 0/i\echo 0 > /sys/devices/platform/mesonfb/graphics/fb1/blank" ${path_rootfs}/etc/rc.local
# echo "S0:23:respawn:/sbin/getty -L ttyS0 115200 vt102" >> ${path_rootfs}/etc/inittab

# export MALLOC_CHECK_=0
# LC_ALL=C LANGUAGE=C LANG=C chroot $path_rootfs
# export PATH=/bin:/sbin:/usr/bin:/usr/sbin:$PATH
# apt-key adv --keyserver keyserver.ubuntu.com --recv-keys AB19BAC9
# dpkg-divert --add --local --divert /usr/sbin/invoke-rc.d.chroot --rename /usr/sbin/invoke-rc.d && cp /bin/true /usr/sbin/invoke-rc.d
# cat << EOF > /usr/sbin/policy-rc.d
#!/bin/sh
exit 101
EOF

# chmod +x /usr/sbin/policy-rc.d
# apt-get update && apt-get -y upgrade
# sed -i -e 's/KERNEL\!=\"eth\*|/KERNEL\!=\"/' /lib/udev/rules.d/75-persistent-net-generator.rules
# cat << EOF > /tmp/debconf.set
console-common console-data/keymap/policy select Select keymap from full list
console-common console-data/keymap/full select en-latin1-nodeadkeys
EOF
# debconf-set-selections /tmp/debconf.set && rm -f /tmp/debconf.set
# locale-gen en_EN en_EN.UTF-8 en_EN ISO-8859-1 && update-locale LANG="en_EN.UTF-8" LANGUAGE="en_EN" LC_ALL="en_EN.UTF-8"
# dpkg-reconfigure tzdata && dpkg-reconfigure locales
# echo "root:root" | chpasswd && useradd -m -s /bin/bash -G adm,sudo,plugdev,audio,video,cdrom,floppy,dip odroid && echo "odroid:odroid" | chpasswd

# apt-get --yes --force-yes install $PACKAGES

# mv /etc/X11/xorg.conf /etc/X11/xorg.conf_save
# cat << EOF > /etc/X11/xorg.conf
Section "Screen"
Identifier "Default Screen"
Monitor "Configured Monitor"
Device "Configured Video Device"
DefaultDepth 16
```

```
EndSection
EOF
# cat > /etc/asound.conf << _EOF_
pcm.!default {
type plug
slave {
pcm "hw:0,1"
}
}
ctl.!default {
type hw
card 0
}
_EOF_

# cat /etc/pulse/default.pa | sed s/"#load-module module-alsa-
sink"/"load-module module-alsa-sink"/g > /tmp/default.pa
# mv /tmp/default.pa /etc/pulse/default.pa
# cat /etc/pulse/default.pa | sed s/"#load-module module-alsa-source
device=hw:1,0"/"load-module module-alsa-source device=hw:0,1"/g > /tmp/
default.pa
# mv /tmp/default.pa /etc/pulse/default.pa
# update-rc.d ssh enable && rm -f /usr/sbin/policy-rc.d && rm -f /usr/
sbin/invoke-rc.d && dpkg-divert --remove --rename /usr/sbin/invoke-rc.d

# lsof |grep mnt
# /etc/init.d/dbus stop
# /etc/init.d/ssh stop

# rm -rf /root/.bash_history && exit && rm -f $path_rootfs/usr/bin/qemu*
# umount /mnt/dev/pts && umount /mnt/dev && umount /mnt/proc && umount /
mnt/sys && umount $path_rootfs && sync
```

For the final step, substitute `/dev/sdX` with the device name of the SD card or eMMC module to which the image should be installed. The media should already have the boot partition installed, with <offset> reflecting the size of the boot partition:

```
# dd if=rootfs.img of=/dev/sdX bs=1M skip=<offset>
```

Single command method

A script file is also available which creates the entire Kali image using one command, which may be downloaded from `http://bit.ly/1HdZgyn`.

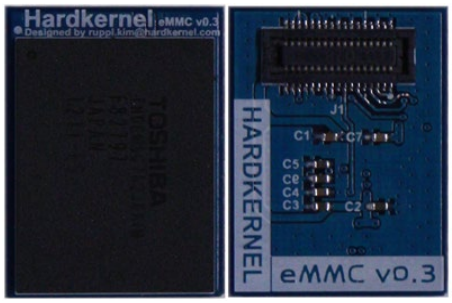
```
$ git clone https://github.com/yhfudev/arch-kali-odroidc1.git
$ cd arch-kali-odroidc1
$ sudo ./runme.sh
```

The script automatically downloads and installs the prerequisites and source/tool trees. It supports multiple Linux distributions, such as Debian and ARCHLinux. For questions, comments and suggestions, please visit the original posts at `http://bit.ly/1CEYLe8` and `http://bit.ly/180xfig`.

EMMC RECOVERY

TAKE ADVANTAGE OF FLASH BLOCK UTILIZATION TO RECOVER YOUR DATA

by Bo Lechnowsky



I recently lost about six months of work on an important script through an unfortunate series of events involving my backup and the -f flag on the command line, when I should've been more cautious (oops!).

Did you know that whenever you save a file, most of the time, the new copy of that file is located in a different block on an eMMC or microSD module? This fact saved me a lot of headaches (literally). To recover a text file that was lost, as long as the media that it was saved on is still intact and has some free space on it, the following procedure on Ubuntu will likely get you your file back. Insert an external USB flash drive if you want to be extra cautious about not overwriting any remnants of your file.

Type the following command in a Terminal window:

```
$ df -h
```

Make a note of the partition where your file was kept, which should be similar to `/dev/mmcblk0p2`. If you inserted a flash drive, also make a note of where that is mounted, like `/dev/sda`.

Try to remember a bit of text that was unique to the file that you were editing. Also, try to estimate how many lines the file was. It's best to estimate on the high side, for instance, if you think it might have been 500 lines, use 1000 instead.

Run the following command in the terminal:

```
$ sudo grep -a -A1000 -B1000 'bit of unique text here' /dev/mmcblk0p2 |
strings > recovered_file
```

Replace the two "1000" numbers with your guess on the file size, the "bit of unique text here" with your unique text that you think was in your file, "`/dev/mmcblk0p2`" with the name of the device where your file was located, and "`recovered_file`"with the path to where you want the recovered file to be saved, like the path to your USB flash drive.

Be aware that this will also output the blocks around where your text was found on the media, so you may see a lot of gibberish, but you should also see fragments of your file, or in some cases, the file in full, bordered by random characters. Also note that sometimes, fragments will be in reverse line order (i.e. line3 -> line2 -> line1). When I used this method, I was so happy to see a recent copy of my file that I didn't care in what order the lines were in, as long as I could reconstruct my work.

One final note is that the most recent fragments seem to be at the end of the recovered file, so I worked from the end of the file backwards to rebuild it. My recovered file was 239 lines long, but a lot of work went into those 239 lines!

ONLY ONE

A TRIBUTE TO THE ACTION GENRE LIKE YOU ALWAYS WANTED

by Bruno Doiche

This is it my friends! Only One is one of those games that pushes all of your buttons: you are an 8-bit hero trapped in the top of a gigantic tower where you take possession of a sword. You raise it and shout "I'll become the Only One!". I couldn't find a single person that wasn't instantly hooked in this adventure against incredible odds. You will find some amazing hilarious easter egg in a game that, like the best games, won't hold your hand, but will bring out the best in of you, making you think both quick and strategically. Want to push your enemies to their doom instead of getting their power? Suit yourself! Want to finish off that other one and steal his shield? No? Fine! Just be sure to fight all the enemies in the best way that you can in the 90 levels of pure fun. Oh, and the soundtrack is legendary!

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

Don't get yourself surrounded, unless you are packing enough power to make these guys run for their lives!

BUILD YOUR OWN AMBILIGHT

ADD AN EXTRA DIMENSION TO YOUR VIDEOS

by Marian Mihailescu

Ambilight is capable of producing some beautiful lighting effects, and can be synchronized to the music or video that is playing on your amazing ODROID media center

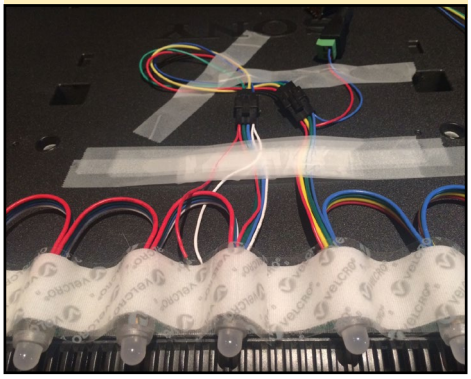
Ambilight is an ambient lighting system for television developed by Philips, where light effects are created around the TV that correspond to the video content. You can achieve a similar effect by using a strip of RGB LEDs and software that samples the image on the screen, then colors each individual LED in a different shade accordingly. In this article, we shall see how to use an ODROID to achieve this.

This article is loosely based on the Adalight Project (`http://bit.ly/1EnG6zZ`), which uses an Arduino for controlling the LEDs. Instead of the Arduino, we are using an ODROID running XBMC as the interface for the video content. The other components are:

- **A strand of LEDs (just like the Adalight, you can use 12mm WS2801 LEDs)**
- **5V/2A power adapter - sufficient for 50 LEDs; for 100 LEDs you need 5V/4A power adapter**
- **A 2.1mm female power jack**

These components are sufficient for the ODROID-C1, but if you are using an ODROID model based on a Samsung Exynos SoC, such as the ODROID-XU, you will also need a level shifter, like the Freetronics Logic Level Converter (`http://bit.ly/1wWkMEb`). LEDs can be purchased either from Adafruit or from eBay (`http://bit.ly/1GYRjgr`). You can use one strip, or combine several strips. Figure 1 shows how I connected the output end to the input end of two 25-LEDs strips.

Figure 1 - Connecting two LED strips, which may be purchased online from either Adafruit or Ebay



Setting up the hardware is straightforward: you connect the input of the first LED strip to the ODROID using 3 wires:

- **Ground (GND: pin 6 on Odroid-C1, pin 2 on Odroid-XU),**
- **SPI clock (SCLK: pin 23 on Odroid-C1, pin 10 on Odroid-XU)**
- **SPI data (MOSI: pin 19 on Odroid-C1, pin 7 on Odroid-XU)**

In the case of Exynos SOC, these 3 wires are connected to the GND, Low I/O 1 and Low I/O 2 of the logic level shifter, with two additional wires:

5V (pin 1 on Odroid-XU), connected to High V IN
1.8V (pin 29 on Odroid-XU) connected to Low V IN

From the logic level shifter, you connect to the LED strip with three wires: GND, High V I/O 1 (clock) and High V I/O 2 (data). The LED strip power is then connected on the red and blue wires at the output end of the strip. Insert the wires in the power jack, and connect the power by inserting the adapter in the jack. Figure 2 shows the connection diagram.

The next step, and the one that I found to be the most difficult, is mounting the LEDs on the TV. If you have

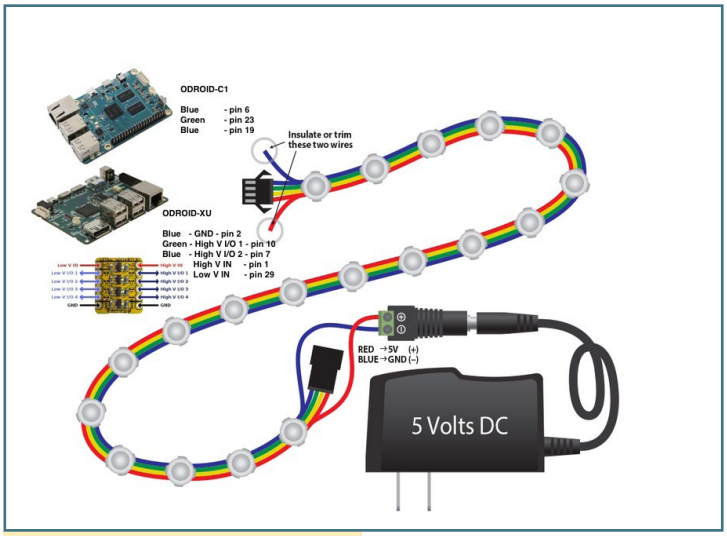


Figure 2 - Connection diagram

a proper toolkit, you can make a metal, wooden or 3D-printed frame with holes for each LED that can be mounted on the TV using the VESA holes. I used double-sided velcro tape: one side pasted on the TV with the LEDs placed on top of the velcro, with the other velcro side keeping the LEDs in position, as shown in Figure 3.

For the software, I used the boblight daemon, which is available from `http://bit.ly/1bdW7S9`, and installed the corresponding boblight XBMC add-on. I then configured, compiled and installed boblight by typing the following commands:

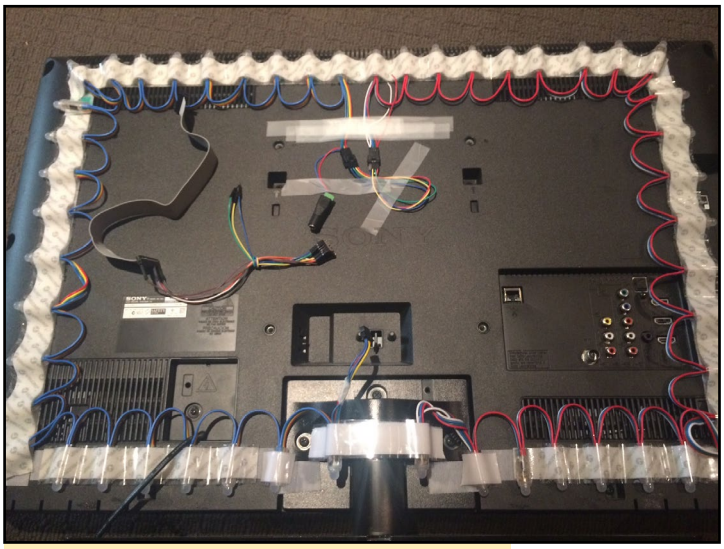


Figure 3 - Mount the LEDs with velcro tape

```
./configure –without-portaudio --without-x11 --with-
out-libusb
make; sudo make install
```

Boblight needs a configuration file, which can easily be generated by Boblight Config Maker, a program where you specify how many LEDs you use and their position on the TV. For the Odroid-C1, the output interface is `/dev/spidev0.0`, while for the Odroid-XU the output interface is `/dev/spidev1.0`:

```
# config file created with BobLight Config Maker v.1.0
# (C) Hans Luijten - http://www.tweaking4all.com
# Orientation naming as seen from the REAR of the TV
# Date: 16-8-14 01:10:39

[global]
interface 127.0.0.1
port      19333

[device]
name         ODroidLight
type         ws2801
channels     150
output       /dev/spidev1.0
interval     20000
rate         48000
```

To start boblight daemon on boot, add an upstart (Ubuntu 14.04) script in /etc/init/boblight.conf:

```
#!upstart
description “Boblight daemon”

env USER=root
```

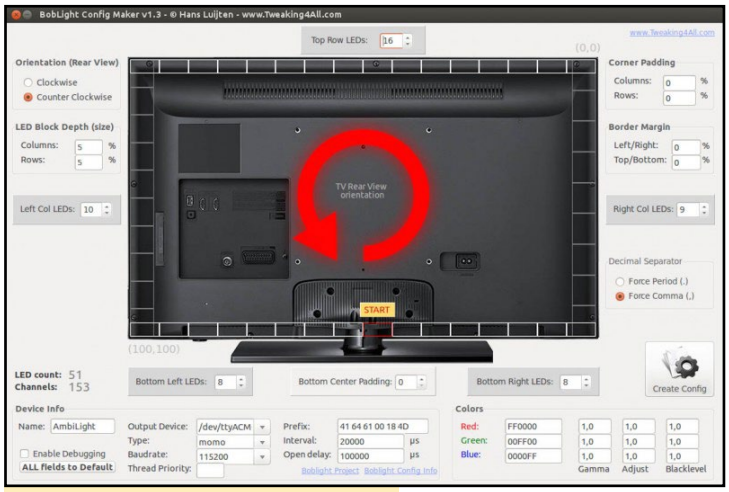


Figure 4 - Boblight Config Maker

```
env GROUP=root
env DAEMON="/usr/local/bin/boblightd"
env DAEMON_ARGS=""

start on runlevel [2345]
stop on runlevel [06]

respawn
respawn limit 15 5

expect daemon

exec start-stop-daemon --start --quiet -b --make-
pidfile --pidfile /run/boblightd.pid --chuid $USER -g
$GROUP --exec $DAEMON -- $DAEMON_ARGS

pre-stop exec start-stop-daemon --stop --signal TERM
--pidfile /run/boblightd.pid --quiet --exec $DAEMON
```

To enable the SPI interface on Odroid-XU, the following kernel patch is required:

```
Author: memeka <mihailescu2m@gmail.com>
Date:   Sun Aug 31 09:01:52 2014 +0930

    enable spi pins on spidev

diff --git a/arch/arm/mach-exynos/board-odroidxu-
ioboard.c b/arch/arm/mach-exynos/board-odroidxu-
ioboard.c
index 6157e07..f643003 100644
--- a/arch/arm/mach-exynos/board-odroidxu-ioboard.c
+++ b/arch/arm/mach-exynos/board-odroidxu-ioboard.c
@@ -84,24 +84,24 @@ static struct   platform_device
odroidxu_ioboard_adc = {
```

```
 static struct s3c64xx_spi_csinfo spi1_csi[] = {
        [0] = {
+               .fb_delay = 0x2,
                .line       = EXYNOS5410_GPA2(5),
                .set_level  = gpio_set_value,
        },
 };
 static struct spi_board_info spi1_board_info[] __
initdata = {
        {
-               .modalias                       =
"ioboard-spi",
-               .platform_data          = NULL,
-               .max_speed_hz           = 20 * 1000 *
1000,     // 20 Mhz
+               .modalias                       =
"spidev",
+               .max_speed_hz           = 40 * 1000 *
1000,     // 20 Mhz
                .bus_num                        = 1,
                .chip_select            = 0,
-               .mode                           =
SPI_MODE_0,
+               .mode                           =
SPI_MODE_3,
                .controller_data    = &spi1_csi[0],
        }
 };

 static struct platform_device odroidxu_ioboard_spi
= {
-       .name                   = "ioboard-spi",
+       .name                   = "spidev",
        .id                     = -1,
 };
```

We may have been overly enthusiastic in setting up our own Ambilight

INSTALLING LINUX ON AN EXTERNAL USB DRIVE

GET MORE SPACE ON YOUR ROOT PARTITION

by @Guso

The March 2014 issue of ODROID Magazine presented a method for running an operating system from external USB drive, which was compatible with earlier ODROID models such as the X, X2 and U2. However, there is an easier way to do so now, since Hardkernel has updated its bootloader software to use a human-legible boot.ini script instead of a compiled boot.scr file, which can now be used with the modern XU3, U3 and C1 devices. This technique allows you to use any size of external drive for the main Linux partition, permitting it to be expanded beyond the 64GB limit of the internal microSD card or eMMC module.

Since ODROIDs are designed to boot from internal storage, the boot partition is required to reside on a microSD card or eMMC module. However, the root partition can be moved to a USB drive, allowing the ODROID to use the internal media only for the initial boot process.

To begin, burn the image to the microSD card or eMMC module, then burn it to the the external USB drive as well, using any method, such as Win32DiskImager or the Linux dd utility. Insert the microSD card or eMMC module into the ODROID, plug in the external USB drive, and boot the device.

Once the boot has completed, make a note of the the device name of the external USB drive by typing the following command into a Terminal window:

```
$ sudo fdisk -l

[...]

Disk /dev/sda: 1000.2 GB, 1000204885504 bytes
255 heads, 63 sectors/track, 121601 cylinders, total
1953525167 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00074666
```

In this example, the device identifier is /dev/sda. Next, determine the UUID of the external USB drive by matching for the device identifier to the output of the blkid command:

```
$ sudo blkid
/dev/mmcblk0p1: SEC_TYPE="msdos" LABEL="BOOT"
UUID="6E35-5356" TYPE="vfat"
/dev/mmcblk0p2: LABEL="trusty" UUID="e139ce78-9841-
40fe-8823-96a304a09859" TYPE="ext4"
/dev/sda1: LABEL="MyLabel" UUID="213C-1DF1"
TYPE="ext4"
```

In this case, the UUID of /dev/sda1 is 213C-1DF1. Edit the file /media/boot/boot.ini, which resides on the microSD card or eMMC module that was initially booted, and look for the following line:

```
setenv bootrootfs "console=tty1
console=ttySAC2,115200n8 root=UUID=e139ce78-9841-
40fe-8823-96a304a09859 rootwait ro"
```

Change the "root" variable to match the UUID identifier noted in the previous step, and save the file:

```
setenv bootrootfs "console=tty1
console=ttySAC2,115200n8 root=UUID=213C-1DF1 rootwait
ro"
```

Finally, edit the file /etc/fstab entry for the "/" partition to match the new drive as well. This file must be edited on the external drive, not the microSD card or eMMC module. Make sure to substitute the actual device identifier for /dev/sda1:

```
$ mkdir ~/temp
$ sudo mount /dev/sda1 ~/temp
$ cd temp/etc
$ sudo nano fstab
```

Here is the first line of the original contents of the fstab file:

```
UUID=e139ce78-9841-40fe-8823-96a304a09859 / ext4
```

```
errors=remount-ro,noatime,nodiratime 0 1
```

Change the first line of the fstab file to use the UUID of the external USB drive:

```
UUID=213C-1DF1 / ext4 errors=remount-ro,noatime,nodiratime 0 1
```

After saving the file, reboot the ODROID and verify that the external USB drive is being used as the root partition:

```
$ blkid -o list
device       fs_type label    mount point   UUID
------------------------------------------------------------------------
/dev/mmcblk0p1
             vfat    BOOT     /media/boot   6E35-5356
/dev/sda1
             ext4    MyLabel  /             213C-1DF1
```

For more information, or to post comments, question and suggestions, please visit the original post at http://bit.ly/18DTdo7.

GPIO SHUTDOWN

ADD AN EXTERNAL SHUTDOWN TO STOP YOUR DEVICE WITH A SINGLE TOUCH

by Mauro Ribeiro and @joerg

If you have an ODROID running as a kiosk, laptop, car computer, or arcade cabinet, it can be useful to add an external shutdown button to power down the device to prevent damage to the storage media, which can be difficult to repair in an embedded system.

Linux version

To enable the Linux version of the script, use a momentary two-pin push button switch, which can be ordered from any electronics store. Connect the button between GPIO pins 15 and 17. Then, create a BASH file with the following contents, and add a reference to the script inside the /etc/rc.local file so that it runs at startup:

```
#!/bin/bash

# PIN 15 -> GPIO 115
# CONNECTION is PIN17(3.3V) to PUSH-BUTTON to PIN15
GPIO_N=115

# Setup GPIO for INPUT!
echo $GPIO_N > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio$GPIO_N/direction

while true; do
        GPIO_VALUE=`cat /sys/class/gpio/gpio$GPIO_N/
value`

        if [ $GPIO_VALUE -eq 1 ]; then
                # Do something here if the button is
pressed
                # like reboot, poweroff...
        fi

        sleep 1

done
```

After adding your own modifications where indicated in the script comments, you can run any Linux command when the button is pushed, such as a custom shutdown script which ensures that all resident programs safely exit before powering down.

Android version

The Android APK version, which may be downloaded from `http://bit.ly/1xgUHAl`, allows a shutdown or a reboot using Pin 35 (GPIO97) and Pin 36 (GPIO98). The app monitors the two inputs, which are activated by shifting down to 0V using a 2K resistor. Pin 35 causes a shutdown, and Pin 36 initiates a reboot. Note that the app does not have a UI, so a desktop icon will not be visible.

Reboot after installing the APK, and grant root permission to the app when requested. The service will start automatically approximately 30 seconds after Android is started. For more information, or to post comments, question and suggestions, please visit the original posts at `http://bit.ly/1FxyMWQ` and `http://bit.ly/1Ex9Wo8`.

OPENHAB

A GUIDE TO OPEN SOURCE HOME AUTOMATION

by Venkat Bommakanti

In the September 2014 issue of ODROID Magazine, we introduced the Freedomotic software platform, which utilizes the Internet of Things (IoT) architecture, which is useful for building and managing smart spaces. In this article, we present another interesting open source platform called openHAB, which is a Java-based, OSGi-compliant, rules-driven, comprehensive home automation software framework. Since it is a vendor and technology-agnostic solution, it is a perfect integration application for the ODROID platform.

In order to be as up to date as possible, this article provides the steps necessary to deploy a recent build of the next release candidate of openHab (v1.7.0), using publicly available prebuilt packages. However, it does not cover:

- **The use of the prebuilt packages of latest stable release (1.6.2)**
- **Building one's own set from source code (GitHub). This method was attempted with latest 1.7.0 daily snapshots, but the build failed on the C1, U3 and x86 Linux systems, despite following the openHAB recommended procedure, due to issues with fetching appropriate openHAB-specific build-related components such as Maven and Eclipse, which were not related to the hardware platform**
- *** The setup of the next-generation 2.0.0 alpha version**

Requirements

1. An ODROID-C1, although these steps may also be used with a higher end ODROID system
2. C1 accessories such as, HDMI cable, CAT 5E+ ethernet cable or WIFI 3 dongle, recommended PSU, RTC battery, and ODROID-VU or HD monitor
3. A 16GB+ eMMC 5.0 module with latest C1-specific Lubuntu desktop image or a 16GB or larger Class 10 Micro SD, with a compatible SD card adaptor
4. C-Tinkering kit and Wiringpi software which will be used to simulate sensors and lights in a home
5. Sonos WiFi sound system to show integration of multimedia control
6. A network where the device has access to the Internet
7. Oracle Java 8
8. openHAB software (Release Candidate: 1.7.0 Build# 879)
9. Intranet access to the C1 via utilities like PuTTY, FileZilla, TightVNC Viewer, or Terminal that can be run on a host computer

Install Lubuntu

Copy the latest ODROID-C1 image on to the eMMC module or SD card, attach the boot media to the device, and boot up the system with the monitor attached. Run the ODROID Utility and set the display resolution to match the monitor, then reboot. Next, run the utility again and expand the installation partition to use all of the eMMC by selecting the "Resize your root partition" option. Reboot again, then run the ODROID Utility one more time to configure and update all remaining aspects of the system such as the kernel and u-boot drivers, followed by another reboot.

Next, update the Ubuntu applications, and reboot the ODROID once all of the updates have been completed:

```
$ sudo apt-get autoremove && sudo apt-get update
$ sudo apt-get dist-upgrade && sudo apt-get upgrade
$ sudo apt-get install linux-image-c1
```

Finally, check the system version from a terminal using the following command to ensure that you have the latest release:

```
$ uname -a
Linux odroid 3.10.70-74 #1 SMP PREEMPT Wed Mar 4
04:13:57 BRT 2015 armv7l armv7l armv7l GNU/Linux
```

Verify hard float

Checking that a hard float version of Linux is installed can be done in several ways. First, the Executable and Linkable Format (ELF) information of the installed image may be verified:

```
$ readelf -a /usr/bin/readelf | grep arm
      [Requesting program interpreter: /lib/ld-linux-
armhf.so.3]
 0x00000001 (NEEDED) Shared library: [ld-linux-armhf.
so.3]
  000000: Version: 1  File: ld-linux-armhf.so.3  Cnt:
1
```

Note the specification of the necessary hard float (armhf) shared library. This confirms the presence of a hard float based operating system (OS). Alternatively, hard float can be verified by inspecting the packages installed on the system. The architecture value for each of the package will be listed alongside the name using the following command:

```
$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-
inst/trig-aWait/Trig-pend
```

```
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                Version        Architecture   Description
+++-==================-==============-==============-==============================
ii  abiword             3.0.0-4ubuntu1 armhf          efficient, featureful word
ii  abiword-common      3.0.0-4ubuntu1 all            efficient, featureful word
…
```

Note the specification of the hard-float (armhf/all) architecture in the list. Finally, the libraries installed can also be used to verify hard float architecture by looking for the "gnueabihf" label:

```
$ sudo ls -lsa /lib
…
12 drwxr-xr-x  4 root root 12288 Mar 16 12:23 arm-linux-gnueabihf
…
```

Verify Java version

OpenHAB requires Oracle Java version 8, which should be installed by default with the official Lubuntu, and can be verified using the following commands:

```
$ sudo update-alternatives --config java
There are 3 choices for the alternative java (providing /usr/bin/java).
Selection  Path           Priority  Status
------------------------------------------------------------------------------
* 0  /usr/lib/jvm/java-8-oracle/jre/bin/java   1067   auto mode
  1  /lib/jvm/jdk1.8.0/bin/java       180    manual mode
  2  /usr/lib/jvm/java-7-openjdk-armhf/jre/bin/java  1063   manual mode
  3  /usr/lib/jvm/java-8-oracle/jre/bin/java   1067   manual mode
Press enter to keep the current choice[*], or type selection number:
```

Press the Enter key to exit, after noting that current version, denoted by an asterisk, is Oracle's Java 8. Another method of verifying Java is to use the "which" command:

```
$ which java
/usr/bin/java
$ java -version
java version "1.8.0_33"
Java(TM) SE Runtime Environment (build 1.8.0_33-b05)
Java HotSpot(TM) Client VM (build 25.33-b05, mixed
mode)
```

Ensure that the version is not being over-ridden by checking for the $JAVA_HOME environment variable, which should show a null output:

```
$ echo $JAVA_HOME
```

If any output is emitted, then the environmental variable, which has been set for another Java version, should be disabled before running openHAB. Reboot after updating the variable,

then check again to make sure that Oracle Java 8 is in use.

Download openHAB

The three (3) major components of the openHAB software:

Runtime: Core, Addons and Demo (optional), Designer
Client UI: Android, iOS & WebUI

For this article, we have chosen to install version 1.7.0 (Build #879) of the openHAB software, which is the most recent at the time of this writing. It may be obtained from `http://bit.ly/1CJO36d`, as shown in Figure 1.

Download the following components (32bit Linux) to the ~/Downloads directory:

```
distribution-1.7.0-SNAPSHOT-addons.zip
distribution-1.7.0-SNAPSHOT-demo.zip
distribution-1.7.0-SNAPSHOT-designer-linux.zip
distribution-1.7.0-SNAPSHOT-greent.zip
distribution-1.7.0-SNAPSHOT-runtime.zip
```

If you wish to use the stable 1.6.2 version files instead, you can download the files from http://bit.ly/1EBtbe2.

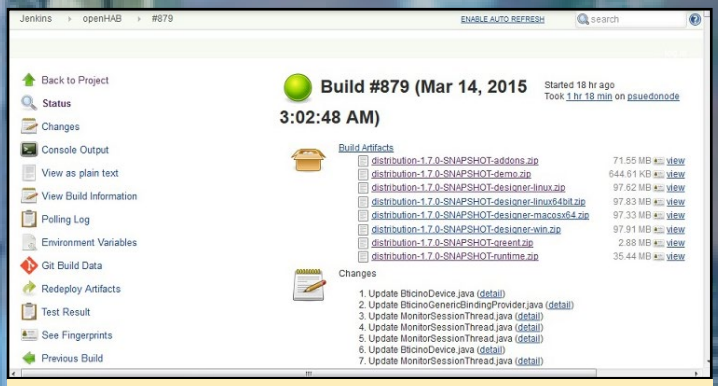


Figure I: Download the openHAB (work-in-progress) software

Install openHAB

After downloading the files, create a temporary directory in order to hold all of the related software. This temporary folder can then be moved to the system root (/) for access by any user:

```
$ cd ~ && mkdir openhab && cd openhab
$ mv ~/Downloads/distribution-1.7.0-SNAPSHOT*.zip .
```

Next, expand the runtime zip file, and check unzipped files, cleaning up directories at every step:

```
$ unzip distribution-1.7.0-SNAPSHOT-runtime.zip
$ ls -lsa
…

4 drwxrwxr-x 2 odroid odroid  4096 Mar 14 03:03 addons
4 drwxrwxr-x 8 odroid odroid  4096 Mar 14 03:03 configurations
4 drwxrwxr-x 2 odroid odroid  4096 Mar 14 03:03 contexts
4 drwxrwxr-x 2 odroid odroid  4096 Mar 14 03:03 etc
12 -rw-rw-r-- 1 odroid odroid       11232 Mar 14 03:03 LICENSE.TXT
4 -rw-rw-r-- 1 odroid odroid   626 Mar 14 03:03 README.TXT
4 drwxrwxrwx 6 odroid odroid  4096 Mar 14 04:04 server
4 drwxrwxr-x 2 odroid odroid  4096 Mar 14 03:03 sounds
4 -rw-rw-r-- 1 odroid odroid   994 Mar 14 03:03 start.bat
4 -rw-rw-r-- 1 odroid odroid  1146 Mar 14 03:03 start_debug.bat
4 -rwxr-xr-x 1 odroid odroid  1132 Mar 14 03:03 start_debug.sh
```

```
4 -rwxr-xr-x 1 odroid odroid   969 Mar 14 03:03 start.sh
4 drwxrwxr-x 4 odroid odroid  4096 Mar 14 03:03 webapps

$ rm distribution-1.7.0-SNAPSHOT-runtime.zip

Then, unpack the downloaded addons:

$ mv distribution-1.7.0-SNAPSHOT-addons.zip addons/
$ cd addons
$ unzip distribution-1.7.0-SNAPSHOT-addons.zip
$ rm distribution-1.7.0-SNAPSHOT-addons.zip
$ cd ..
```

Although numerous add-ons get unpacked, not all will be needed by the deployment. Add-ons can also be added or re-installed at any time to this folder, although system resource usage increases with every add-on used by the system. Finally, set up a basic personal configuration for openHAB:

```
$ cd ~/openhab/configurations
$ cp openhab_default.cfg openhab.cfg
$ mv openhab_default.cfg openhab_default-orig.cfg
```

We will edit the openhab.cfg configuration file later, which is expected to be present during bootup and runtime, while experimenting with our installation.

Install the demo

The demo application may be installed after backing up any files with names similar to those used in the demo application:

```
$ cd ~/openhab
$ mv README.TXT README-openhab.TXT
$ unzip distribution-1.7.0-SNAPSHOT-demo.zip
```

You will be prompted with a question as to whether it is safe to overwrite existing same name files. Select Y (for Yes) to approve the overwrite.

Prepare the experiment

In our experiment, to validate the openHAB installation, we will include the following tests:

Using the C-Tinkering kit to simulate a motion sensor in our state-of-the-art electronics workshop. The light sensor plays the role of the motion sensor. When someone approaches the motion sensor, an LED goes ON, which simulates an alarm or light. The status of the alarm will be displayed in the basic, built-in openHAB web interface in real-time. For this test, we will be using the built-in GPIO add-on.

Managing a Sonos WiFi sound system, by pausing or resuming the playing of music on a Pandora channel, and viewing the name of the current track being played. For this test,

we will be using the built-in Sonos addon.

In this experiment, we will address two basic aspects of a typical installation:

Integration of supported sensor and trigger systems
Integration of supported off-the-shelf systems

The only other aspect that remains to be covered is the integration of user-built addons. This article should go a long way in getting you started in this endeavor.

We will use the demo application to simulate a home with two floors (ground and cellar), where the cellar represents the electronics workshop with the motion sensor, and the ground floor with a Living room where the Sonos system will be set up. For the motion sensor portion of the experiment, follow the steps of the C-tinkering kit setup in the Osc1lloscope article, which was published in the March 2015 issue of ODROID Magazine. Test the setup as described there, ensuring that the LED comes ON as you move a finger towards the light sensor. This will simulate the motion sensor so that, as someone moves close to the sensor, the alarm will go off, which is represented by turning the LED to its ON state.

In a Terminal window, run the sample application that was used in the Osc1lloscope article. In another Terminal window, run the GPIO utility that was built when the Wiringpi library was installed:

```
$ sudo ./gpio readall
```

The LED in the example is attached to GPIO #101, which corresponds to pin #21 of Wiringpi. Whenever the sensor is not blocked, you see the following output (status value: 0, LOW):

```
+------+-----+----------+------+- Model ODROID-C -+------+----------+-----+------+
| GPIO | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | GPIO |
+------+-----+----------+------+---+----++----+---+------+----------+-----+------+
…
|  101 |  21 | GPIO.101 |  OUT | 0 | 29 || 30 |   |      | 0v   |     |
…
```

Whenever the sensor is blocked, you see this output (status value: 1, HIGH):

```
+------+-----+----------+------+- Model ODROID-C -+------+----------+-----+------+
| GPIO | wPi |   Name   | Mode | V | Physical | V | Mode |   Name   | wPi | GPIO |
+------+-----+----------+------+---+----++----+---+------+----------+-----+------+
…
|  101 |  21 | GPIO.101 |  OUT | 1 | 29 || 30 |   |      | 0v   |     |
…
```

For the second part of the experiment involving Sonos, connect the Sonos Bridge to the network so that it is visible to the C1 over the network, and power on the Bridge and the Sonos

Speaker systems. Wait until all flashing status LEDs on the two devices remain solid. At this point, you can look up the IP addresses of these two devices using either your router's interface or the Sonos' Windows 7 UI, shown in Figure 2, which has the relevant values circled in red.

The IP address of the Sonos Speaker in this example is 192.168.7.45, with a serial # 00:0E:58:FB:99:A2. By navigating a browser to http://192.168.7.45:1400/status/topology and substituting your actual IP address, you will see similar information to that shown in Figure 3.

Make a note of the UID information for the speaker, which is RINCON_000E58FB99A201400 in this example. Incidentally, it is constructed using the serial # of the device. This UID information will be used later, during the openHAB configuration step.

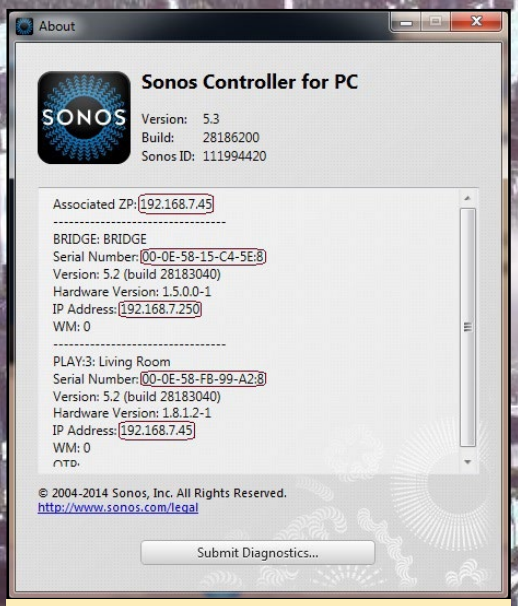


Figure 2: Sonos IP address

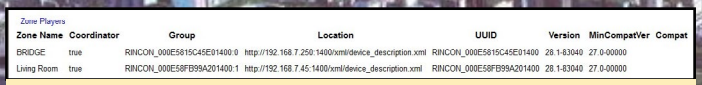


Figure 3: Sonos zone locations

Update configuration

With the suggested experiment in mind, we can now modify the configuration files. First, we will update the openHAB configuration file:

```
$ cd ~/openhab/configurations
$ medit openhab.cfg
```

Update the Sonos Binding section to reflect the Sonos Bridge & the Sonos Speaker information to match the following code, and save the changes:

```
################################ Sonos Binding ########################################
#
#Add a line for each Sonos device you want to pre-define
#The format is <name>.udn=<RINCON UID>
#
sonos:office.udn=RINCON_000E5815C45E01400
sonos:living.udn=RINCON_000E58FB99A201400

#
#Interval, in milliseconds, to poll the Sonos devices for status variables
sonos:pollingPeriod=1000
```

Next, update the items in the demo home configuration:

```
$ cd cd ~/openhab/configurations/items/demo.items
$ medit demo.items
```

Change the workshop light entry, which corresponds to the use of the GPIO via pin #21:

```
Switch Light_C_Workshop "Workshop"   (gC, Lights)
{gpio="pin:21"}
```

Then, change the date/time entry to reflect your timezone, which is Los Angeles in this example:

```
/* NTP binding demo item */
DateTime Date "Date [%1$tA, %1$td.%1$tm.%1$tY]" <calendar> { ntp="America/Los_Angeles:de_DE" }
```

Finally, add the Sonos item related entries, using one line per entry:

```
/* multimedia */
Switch Sonos_GF_Living "Sonos Play/Pause" (GF_Living, Sonos) {sonos="[ON:living:play],[OFF:living:
pause]"}

String Sonos_GF_Living_Track "Current Track [%s]" (GF_Living, Sonos) {sonos="[RINCON_000E58FB99A20
1400:currenttrack]", autoupdate="false"}
```

For the Sonos device in the simulated Living room, we have implemented a Play/Pause switch, while also displaying the current track being played.

Move the openHAB folder

To make openHAB available to any user account, move the entire directory to its final destination:

```
$ sudo mkdir /opt
$ sudo mv ~/openhab /opt
$ cd /
$ sudo chown -R root:root /opt
```

Run openHAB

To launch openHAB, type the following commands into a Terminal window:

```
$ cd /opt/openhab
$ sudo ./start.sh
```

Watch for any errors in the output, to see if they correspond to any configuration mistakes, and address them before moving on to the next step. The Sonos configuration may need to be adjusted in case the UID does not match, which would be indicated in the following entries:

```
2015-03-18 12:16:30.480 [INFO ] [.service.AbstractActiveService] - Sonos Refresh Service has been
started
2015-03-18 12:16:30.889 [INFO ] [.b.sonos.internal.SonosBinding] - Querying the network for any
other Sonos device
2015-03-18 12:16:32.797 [INFO ] [.b.sonos.internal.SonosBinding] - Found a Sonos device (S3) with
UDN uuid:RINCON_000E58FB99A201400
```

Once openHAB is running correctly, navigate a browser on the host testing computer to point to the ODROID's instance of the openHAB demo application:

```
http://<ip-address-of-c1>:8080/openhab.
app?sitemap=demo
```

The Jetty-based web server used by openHAB will present a welcome screen as shown in Figure 4. Notice that the demo home page shows the high level attributes (first floor, ground floor, and cellar/workshop) as specified in the configuration. Make sure that the date also corresponds to the previously configured time-zone specification.

To verify that the experiment is working correctly, click on the cellar item, which should list all of the devices related to it, including the Workshop light, as demonstrated in Figure 5. Next, move a finger close to the light sensor on the C-tinkering kit, and verify that the LED turns ON. This will be reflected in the openHAB UI as shown in Figures 5 and 6, simulating an intruder in your workshop.

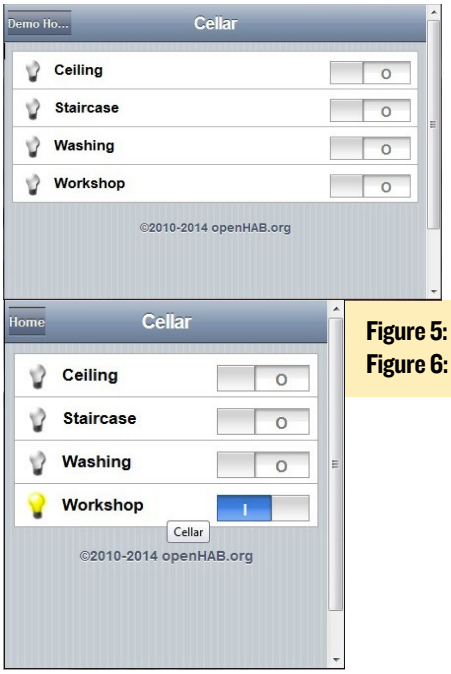
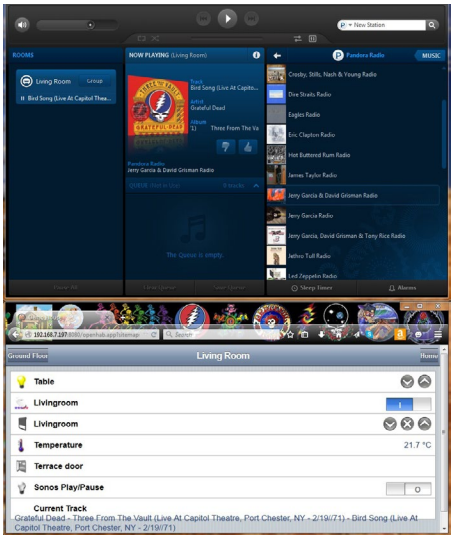


Figure 5: Workshop light (alarm) OFF
Figure 6: Workshop light (alarm) ON

In a real-world situation, you can tie other responses to these events, such as sending a text message or email. The relevant status information is being relayed via pin #21 (GPIO #101) using the GPIO infrastructure, as noted earlier

Next, start the Windows 7 Sonos application, which should

Figure 4: openHAB welcome screen



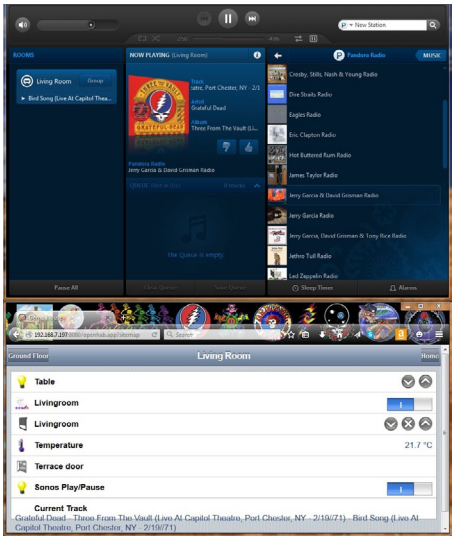


Figure 7: openHAB UI & Sonos UI reflecting play

Figure 8: openHAB UI & Sonos UI reflecting a pause

detect the Sonos Bridge and Speaker systems. As a test, start up a Pandora channel and play some music. Then, on the openHAB home web page, click on the ground floor item, which should present a screen reflecting the Sonos play/pause switch and current audio track as shown in Figure 7.

Verify that the music being played via Pandora (on Sonos) is the same as the one that is displayed on the openHAB UI. Click on the play/pause switch to stop the music, which should result in a status change, similar to Figure 8.

Further experimentation

You are encouraged to download additional components of openHAB, such as the designer and fancier web UI called greent, and experiment with their installation and usage. These files are included in the nightly Jenkins builds as distribution-1.7.0-SNAPSHOT-designer-linux.zip and
distribution-1.7.0-SNAPSHOT-greent.zip.

Once you are comfortable with the setup, you can develop your very own add-ons and deploy them to your openHAB installation. As you can see, we have just scratched the surface of the capabilities of openHAB. There is wealth of information available from the openHAB website at http://www.openhab.org as well as other websites/forums, so have fun integrating all sorts of devices and events!

Disclaimer

Please spend some time researching the topic of home automation well before implementing openHAB in your own home, and take all the appropriate precautions when automating. The information here is only for educational and fun purposes, and the author and publishers are not responsible for any damage resulting from any openHAB implementation. Make sure to check your various relevant insurance policies for accident coverage, because you are ultimately responsible for your own security and safety.

Additional resources

http://bit.ly/1G0rHOP
http://bit.ly/1BHniKe
http://bit.ly/1N3kIoh
http://bit.ly/1CJR7z2
http://bit.ly/1FQn7lS
http://bit.ly/19JuH6d
http://bit.ly/1IhiaCf
http://bit.ly/19dWx9n
http://bit.ly/1xRitO3

ODROID-C1 TRIPLE BOOT

PACK UBUNTU/DEBIAN LINUX, ANDROID AND OPENELEC INTO A SINGLE PHYSICAL VOLUME

by @loboris

I created a script to enable triple booting of Ubuntu/Debian Linux, Android and OpenELEC from the same SD card. It includes a boot menu which can be used to select the desired operating system when the ODROID boots up.

To begin, download the triple-boot build scripts from `http://bit.ly/19dGxEv`, which must be run on a host Linux PC. In order to work with SD card images, you must have kpartx installed on the host computer:

```
$ sudo apt-get install kpartx
```

You will also need 4 SD cards: a working Android copy (created by by running the Hardkernel self-installation image), an SD card with Linux installed, such as one of my minimal Linux images from `http://bit.ly/1DLugEZ`, an OpenELEC SD card, and a blank SD card which will hold the final multi-boot image, which should be at least 8GB in size.

Extract Android

Extract Android by inserting the ODROID-C1 Android SD card into a USB reader, and navigating to the directory where the script was downloaded. Edit the "extract_android" script using a text editor and update the SD card block device name (/dev/sdX) to match the name of the Android SD card. Also, change the desired output directory and save the script.

To run the script, type the following into a Terminal window:

```
$ sudo chmod +x ./extract_android && ./extract_an-
droid
```

This will copy the Android boot partition as well as the system, userdata, cache and storage partitions to your output directory. This will be the basis for the image, and the next steps will extract the other partitions (Linux and OpenELEC) to the SD card.

Extract Linux

Next, insert the Odroid-C1 Linux sd card into USB reader. While in the script directory, edit the "extract_linux" and update the SD card bock device name (/dev/sdX) as well as the desired output directory similarly to the Android procedure. Then, run the Linux script, which will extract the Linux boot, FAT and Linux (ext4) partitions to the output directory:

```
$ sudo ./extract_linux
```

Extract OpenELEC

The procedure for preparing OpenELEC is identical to the Android and Linux extraction. Use the "extract_openelec" script, then run it in order to extract then OpenELEC partitions to your output directory:

```
$ sudo ./extract_openelec
```

Prepare the SD Card

Insert the blank 8GB+ SD card to be used for triple boot,. Edit the "create_dual_boot_sd" script, then update the SD card block device (/dev/sdX), source directories and desired partition sizes. Note that the Linux partition will always be extended to the end of the SD card, using the remaining space after allocating the Android and OpenElec partitions. After saving the script, run it:

```
$ sudo ./create_dualboot_sd
```

Once the script has completed, the blank partition layout on the triple-boot SD card will be:

Android Internal SD card storage (mmcblk0p1)
System (mmcblk0p2)
Userdata (mmcblk0p3)

Extended partition
Cache (mmcblk0p5)
OpenElec (mmcblk0p6)
Linux (mmcblk0p7)

Copy files to SD card

Edit the "copy_to_sdcard" script, update your sdcard block device (/dev/sdX) and source directories, then run the copy script:

```
$ sudo ./copy_to_sdcard
```

Once the script completes, the SD card is now ready for triple boot with Android, Linux and OpenELEC using an ODROID-C1. On boot, you wil be presented with a boot menu for selecting your desired operating system.

Notes

The first SD card partition is formatted as FAT32, which allows it to be read by Windows. It is important not to delete any of the files in the root directory of the first partition, as they are required for the triple boot to work.

If your SD Card is large enough (16 or 32 GB), the scripts can be easily adapted to boot to more than one Linux installation. Further customization is available with two additional scripts:

"update_logo", which allows you to update the logo image
"update_uboot", which facilitates updating the Android boot partition files without erasing the entire SD card

Pre-built images

For convenience, you can download pre-built images from `http://bit.ly/1MY2zKv`, which are ready to be copied to the SD card. The file named multiboot.tar.xz contains all the scripts and extracted directories for Android, Ubuntu-utopic (minimal) and OpenELEC 5.0. First, extract the package as root in order to preserve ownership and permissions:

```
$ sudo tar -xpsf multiboot.tar.xz
```

Navigate to the multiboot directory, adjust your SD card block device and desired partition size by editing the script as described above, then run it to create the triple-boot system on your SD card.

```
$ sudo ./create_dualboot_sd
$ sudo ./copy_to_sdcard
```

You can install full or minimal desktop packages after booting. For questions, comments or suggestions, please refer to the original post at `http://bit.ly/1EBmLvt`.

IMPROVED XU3 FAN
STAYING QUIET

by Tomasz Nazar

The stock fan on the ODROID-XU3 runs often, even when the CPU is idling, so this script improves upon the original fan service. It has been tested on Ubuntu 14.04.1 and ARCHLinux 3.10.69-1-ARCH, and will most likely work on other distributions with slight adjustments.

Safety

I did my best to adjust the fan speed based on the current maximum temperature of any sensor. When the script quits, it sets up the fan mode to the original factory settings. Use it at your own risk!

Installation

The script is available for download from my GitHub account at `http://bit.ly/1EysWA8`:

```
$ cd /usr/bin
$ sudo apt-get install git
$ git clone https://github.com/nthx/odroid-xu3-
fan-control.git
$ cd odroid-xu3-fan-control
$ sudo chmod +x *
$ sudo ./odroid-xu3-fan-control.sh
```

Running as a service

To allow the improved fan script to automatically start when the system boots, edit the odroid-fan-controller file and add the path of the odroid-xu3-fan-control.sh script, using the full path name, then type the following commands into a Terminal window in order to add it to the service list:

```
$ cd /etc/init.d/
$ sudo ln -s /usr/bin/odroid-xu3-fan-control/
odroid-fan-controller
$ sudo update-rc.d odroid-fan-controller defaults
```

You can also use the following command to manually start and stop the controller:

```
$ sudo /etc/init.d/odroid-fan-controller start
$ sudo /etc/init.d/odroid-fan-controller stop
```

To further improve the quietness of the fan, there is an easy hardware modification that can be made, which involves installing a replacement fan.

LINUX GAMING

THE ORIGINAL WARCRAFT SERIES

by Tobias Schaaf

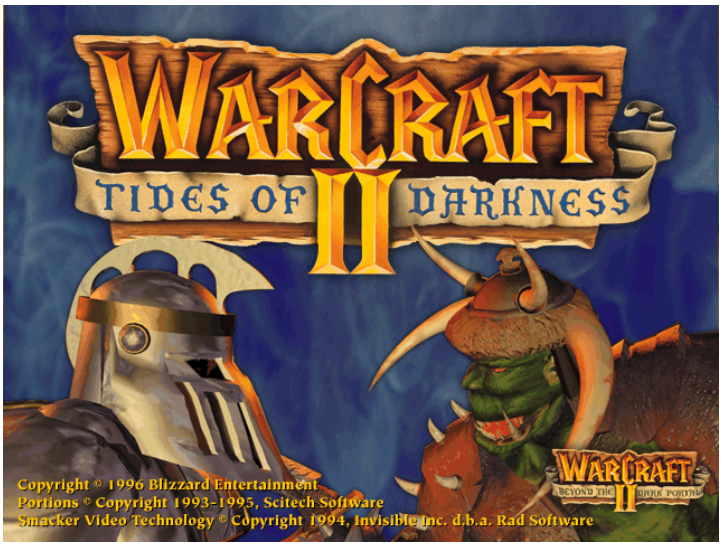
I was recently asked if I could write a feature about the WarCraft games on the ODROID. Since WarCraft, in general, is a very nice series of real time strategy (RTS) games, I want to share my findings and experience, and what to expect when playing WarCraft on the ODROID.

DOS Emulation

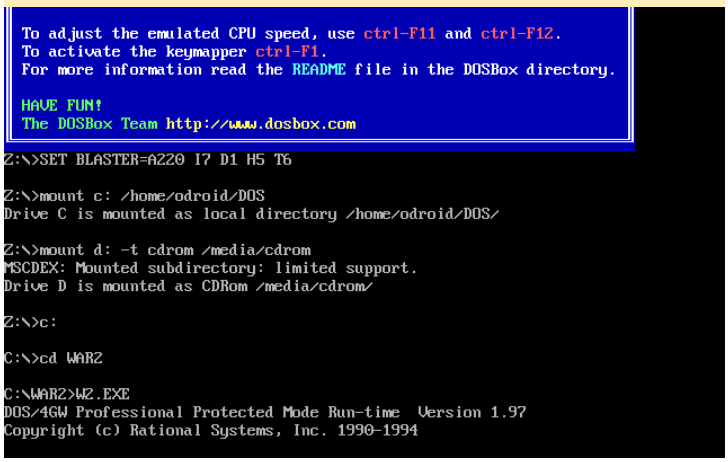
WarCraft – Orcs & Humans (1994), WarCraft II – Tides of Darkness (1995) and WarCraft II – Beyond the Dark Portal (1996) are games similar to Dune 2 and the Command and Conquer series. You lead an army that you can build in different kinds of buildings, upgrade them through different research objects, and let them fight against an enemy.

In WarCraft, this is based in a fantasy setting, with Elves, Orcs and Humans, which fight in epic battles and use resources such as gold, wood and oil to build up their armies. It's a very nice series and can provide many many hours of interesting gameplay, even more so if you play matches against other players.

Since the games are only available for DOS, I thought the easiest way to play them is to launch them using DOSBox, which provides DOS emulation, and play the games in the way in which they were originally intended, back in "the old days".



Figures 1, 2, 3 - WarCraft 2 on the ODROID U3 running in DOSBox



I started up DOSBox, loaded the game, and it worked right away, with no lags or complications. I just mounted the CD and started playing, without having to perform any programming magic at all. So, it seems like I'm done with the article, right? Nothing to do here, just boot it up and be done.

Well, if it would have been just that, I probably wouldn't even have written the article in the first place. So what is there to do if everything is working perfectly fine? Well, the game is

rather old and therefore rather limited. It runs in 640x480 resolution, so that doesn't really look very modern, and if you run the game in full screen mode, it exhibits minor issues, since 640x480 is not displayed correctly. So, I thought that I should at least check to see if I could improve the gaming experience a little bit.

To do so, I referred to my earlier article about DOSBox in the November 2014 issue of ODROID Magazine, where I explained that glshim can be used to improve DOSBox experience using the OpenGL interface. For this, we first change the following options in the DOSBox configuration file:

```
fullscreen=true
fullresolution=1920x1080
output=opengl
```

After that, I made sure that the libgl-odroid package from my repository was installed, and started DOSBox with the following command, which activates OpenGL support:

```
LD_LIBRARY_PATH=/usr/local/lib/
dosbox
```

With that, I could run the game at 1920x1080 resolution, and was able to scale it to full screen, which looked very nice. The game was still working perfectly well, and the picture was finally fitting the HD monitor properly.

There were some minor issues with the blend-over effect in the start menu, which was very slow, but the game itself was unaffected, so that glitch may be ignored. If you're looking to replay the original game as it was, this is as close as you can get, and it's really working great on the ODROID.

Still not satisfied

Although this basically solved everything, and the game was working well, I personally was not satisfied. Now that it's 2015, the game has really aged over time. You can only select nine units at once, and there is no build queue. Limitations like this make the game feel very outdated, since "modern" RTS games don't have such restrictions.

Additionally, the screen resolution is faked through glshim. It would be nice to actually run the game in high resolution rather than have 640x480 scaled up to 1080p. Therefore, I looked into another project that I knew would run on the ODROID called Stratagus. Stratagus is an RTS engine that allows you to play strategy games on almost any kind of hardware. The engine is very capable, and many projects use it, so there actually is a mod to run WarCraft, WarCraft II and even StarCraft on it. Besides that, there are quite some games that were directly developed to use the Stratagus engine. Check out their web site at http://stratagus.com to find out more about the different games using Stratagus.

Without getting too technical, Strategus works on the ODROID, but I wasn't able to get OpenGLES support to work, since the game requires SDL 1.2, which does not have OpenGLES support. Instead, it requires special versions of SDL 1.2 to use OpenGLES. These versions are very hard to find, and although I found a version of SDL_GLES which is just meant as an extension, it's not working.

There is rumored to be an SDL 1.2 version with integrated OpenGLES support available, but I wasn't able to find it (yet).

However, ODROIDs are generally fast enough to simply run the game with standard SDL without hardware acceleration, and so I could easily start Strategus by using SDL. I then created a couple of packages for easy installation of Stratagus. The packages for WarCraft I and II are called war1gus (Warcraft I) and wargus (Warcraft II), which may be found in my repository at http://bit.ly/1MKIQvQ. You can easily install the games via apt-get install by adding the package list for the all/main repository to your Linux distribution.

Installation

Installing Stratagus is very easy: just type "apt-get install stratagus", and you're done. The installation of wargus (WarCraft 2) is more tricky. First, make sure you either have a external CD/DVD drive connected, and that your CD is inserted (or an ISO with the game mounted), or simply copy the "data" folder containing the .war files to your ODROID.

Next, install wargus using the command "apt-get install wargus". After the installation is completed, it will bring up a dialog window asking you to confirm the options for the installation. Here's how you should answer these questions:

```
Extract Warcraft II data files
now? - Yes
```

Next, you have to enter the path to either where your CD is mounted or where the data folder is located.

```
Extract and convert MIDI sound
files? - No
```

This is very important because the MIDI option was removed from the tool that is used to extract the data, but they forgot to remove the question from the menu. If you select Yes, the installation will fail.

```
Extract and convert videos?
```

This option is up to you. The videos are not in a very good resolution and will take up about 5.7 MB of disk space.

```
Rip sound tracks from CD-ROM?
```

This option only works if you have the physical CD in a CD/DVD drive connected. Saying No is probably the right thing for most people here. Wargus also requires TimGM6mb.sf2 MIDI soundfonts, which were downloaded

Figure 4 - WarCraft II in 1080p on the ODROID U3

from http://bit.ly/1MHNqga. Since they are free to use, I added them to the package so that they would be automatically installed. After the game data is extracted (and, if selected, movies are converted) you are ready to play.

Benefits of Stratagus

You might wonder what you get by using Stratagus instead of the DOS version. Figure 4 shows what Warcraft II looks like in with Stratagus, which shows some of the changes such as full 1080p support instead of just 640x480 upscaled resolution. The wider screen shows much larger portions of the map as well.

Blue progress bars from workers and building indicate the progress of the current task or build order.

Health bar of units and building range from green over yellow and orange to red, so you can see the status of your units and buildings without actually having to click on every single one. There are also a few things you can't see from the picture, such as the improvement that the number of units that you can select and move around is no longer limited to nine units. If you have an army of twenty units, you can all move them at the same time. You can also have build orders of units of six and more units, although only six will be shown. If you click ten times to build a unit, ten units will be built.

Disadvantages of Stratagus

Although most of the things in the game are an improvement, there are some downsides that I want to mention. Since the program runs in SDL, it runs mostly on CPU, so it can be very processor-intensive, which can lead to some lagging on big maps with many units and building. I found that turning off the "Fog of War" option can increase performance a lot, but this also changes your gaming experience, so it's up to you if you want to do so.

While the Exynos 4 and 5 series devices from ODROID shouldn't have any issue with the game, Strategus running on the ODROID-C1 may be a bit slow because of the limited CPU power, as compared to the other ODROID models. However, I guess that it should still work in a decent speed on the C1.

The development status on stratagus and wargus is, for now, unclear. Stratagus has been on version 2.7.2 for a couple of years now, and development seems to been taken over by some Ubuntu developers on Launchpad. I found that my versions crashes at the start of level 5 of the Alliance game, but Orcs did not seem to have the same issue. It can probably be fixed easily, but that's something I haven't looked into yet.

Another indication that Stratagus seems not to be a completed project is that there's also a "stargus" project, which is StarCraft playing on stratagus. StarCraft is one of my all-time favorites, and I would love to play it, but not all parts of the games are finished. Humans and Zergs seem to be mostly done, but Protoss are completely missing. I found that you can't even play the campaign, which is also a downside, since StarCraft has a very nice story. The stratagus engine, and the games developed for it, look very promising, but they don't seem to be finished yet, and it seems that development has come to a halt on some aspects of the project.

WarCraft I – WarIgus

Although there is a version of WarCraft I called war1gus for Stratagus, which works similar to wargus, it seems to work somewhat strangely. Buildings are rendered in normal speed, but units walk in slow motion. I think you can get a better experience of playing WarCraft I by using the DOSBox variant. I will try to provide a war1gus package as well in my repository for anyone who wants to try it. Please note that this version can actually use the MIDI convert options, which can takes quite some time to run: about 30mins on a U3, and probably longer on a C1.

StarCraft – Stargus

StarCraft 1 is really nice and works with Stratagus as well. It requires the old CD version rather than the downloadable version from the Internet. It will extract all the files needed, which only takes a few minutes. Stargus does not offer any campaigns, although the campaign missions are extracted anyway. The graphics are very buggy: Protoss use graphics from Zerg and Humans when they build something, and the game also crashes quite often. Only Humans are working well enough to play.

Conclusion

Stratagus is a nice engine which seemed to be made for WarCraft II, since it's running very well with it. There are some developments on other games based on this engine, which seem to work great as well. If you try to run games that were not meant to work on the engine, such as WarCraft I or StarCraft, the engine exhibits issues.

However, playing WarCraft II is a very nice experience with Stratagus, and WarCraft I is also running well enough to be playable. If you want to have the original gaming experience, you should instead use the DOS version running on DOSBox scaled through glshim, as mentioned above. If you want a more modern experience and have some good multiplayer matches, use Stratagus and re-experience WarCraft II on a modern platform.

Warcraft is a highly addictive game, so make sure to play it in moderation, and go outside every once in a while!

ODROID-CI TUTORIAL VIDEO CONTEST

SHOW OTHERS WHAT YOU KNOW

by Bo Lechnowsky

How would you like to earn $25 to $100 in ameriDroid store credit? All you have to do is:

1. Come up with an idea for a tutorial video that hasn't been done yet, or one that you think could be done better
2. Make a video showing the tutorial
3. Mention that the products in the tutorial can be purchased at ameriDroid.com or other Hardkernel distributors worldwide
4. Upload your video to a video sharing site like YouTube or Vimeo
5. Submit the link to your video using our Contact page at http://ameridroid.com/contact

We'll watch your video and judge it based on:

- Quality of teaching
- Quality of video
- Quality of audio

Make sure to speak clearly, and that any background music is well below the voice volume. Keep your video to the point, making it as short as possible without rushing. Keep your tutorial positive and family-friendly. If we post your video (or a link to your video), we'll give you $25 in store credit. By submitting your video (or link), you agree to allow ameriDroid to edit and/or repost the video. In addition, the best video of the week will earn an additional $75 bonus credit!

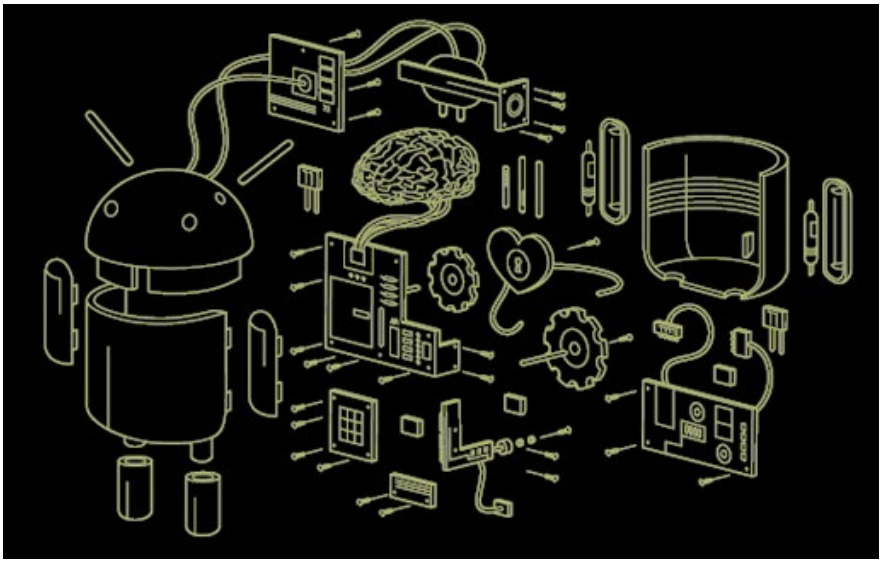
Please note that AmeriDroid reserves the right to withhold the additional $75 bonus credit offer if none of the submitted videos during the target week meet certain quality levels. A new week starts every Sunday at 12:01am Pacific Time, and runs until the following Saturday at midnight Pacific Time. AmeriDroid also reserves the right to change the contest rules or discontinue the contest at any time. We really want to see the community create some great tutorial videos to help each other out!

For more information and to ask questions, please visit the Ameridroid website at http://bit.ly/1NKcIed.

ANDROID DEVELOPMENT

INTRODUCING THE NEW LOW MEMORY KILLER DAEMON

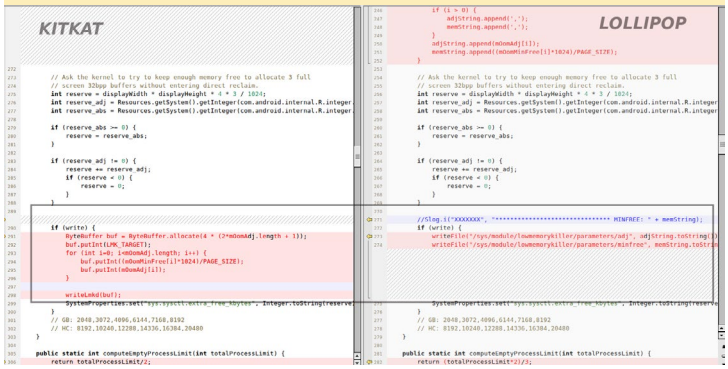
by Nanik Tolaram



Android has matured quite a bit since its first version, and over the course of the releases since Ice Cream Sandwich, a lot of new features have been added. In every updated version, the features depend on what Google are targeting. In the case of the latest Android release (Lollipop), the main focus was on design and performance. One of the new tools that was introduced internally in Lollipop is LMKD, which stands for Low Memory Killer Daemon. In the previous versions of Android, this tool was part of the framework that communicated with the kernel driver layer, but now it is separated out. We will take a look at what LMKD is, and how it's used internally.

In Android, even when the user quits the application, the application process still exists in the system in order to facilitate the restarting of the process. However, with the increase in the number of open applications, system memory can become insufficient, which makes it necessary to kill a part of the process in order to release the memory space. It is up to the Low Memory Killer mechanism to make decisions. LMKD receives memory pressure notifications, and then kills appropriate tasks when memory resources become low.

Figure 1 : KitKat vs Lollipop different code using lmk



LMKD

The heart of the low memory killer resides in the file located in drivers/staging/android/low memory killer.c, which is loaded as part of the kernel. Please refer to the Android Development article in the August 2014 issue of ODROID Magazine for more information about the internal workings of this driver. There are 2 parts of low memory killer: the daemon with which the framework communicates by encapsulating the communication to the driver, and the main kernel driver that takes care of the memory shrinking.

As shown in Figure 2, the framework communicates with the low memory killer daemon which, in turn, sets the correct parameters via the filesys that has been exposed by the kernel driver.

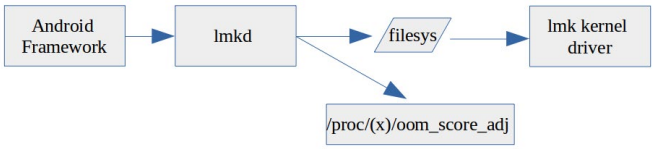


Figure 2 : Communication between different parts

Filesys

The filesys that LMKD uses to communicate with the low memory killer kernel module is shown in Figure 3.

```
#define INKERNEL_MINFREE_PATH "/sys/module/lowmemorykiller/parameters/minfree"
#define INKERNEL_ADJ_PATH "/sys/module/lowmemorykiller/parameters/adj"
```

Figure 3 : Filesystem to communicate with kernel

Init.rc

Like everything in Android, the LMKD application is started during the initialization process, which means that it needs

to be declared in the init.rc, or in one of the available .rc files. Since this forms the main part of the framework initialization, it is actually declared inside the init.rc located in the system/core/rootdir directory, which is shown in Figure 4.

```
service lmkd /system/bin/lmkd
    class core
    critical
    socket lmkd seqpacket 0660 system system
```

Figure 4 : LMKD declaration in init.rc

Framework

The main file that communicates between the Android application and the LMKD app is located inside the framework. There is only 1 file that takes care of this communication, which resides inside the ProcessList.java file. Internally, the framework determines what needs to be communicated to LMKD via command codes, which are defined as shown in Figure 5.

```
// Low Memory Killer Daemon command codes.
// These must be kept in sync with the definitions in lmkd.c
//
// LMK_TARGET <minfree> <minkillprio> ... (up to 6 pairs)
// LMK_PROCPRIO <pid> <prio>
// LMK_PROCREMOVE <pid>
static final byte LMK_TARGET = 0;
static final byte LMK_PROCPRIO = 1;
static final byte LMK_PROCREMOVE = 2;
```

Figure 5 : LMKD command code

The LMK_PROCPRIO is the command that is used most often by the framework, which informs LMKD to modify and create new out-of-memory stats for a particular app every time an application is being processed (started, stopped, or paused).

Another interesting command called LMK_TARGET is specifically used for setting up the global out-of-memory information for the low memory killer driver, which uses the filesys shown in Figure 2. When the kernel starts calling the low memory killer shrinker function, it uses the parameter that has been passed via this command.

ANDROID GAMING

by Jeremy Leesmann

Iron Force
Explosive Tank battles

Have you ever wanted to command a tank in battle? Well now you can with Iron Force. Battle others in an intensive fire fight, win battles, and upgrade to more powerful tanks to become the Top Commander. So go forth and blow some stuff up!

https://play.google.com/store/apps/details?id=com.chillingo.ironforce.android.ajagplay

Beach Buggy Racing
Action cartoon–style racing in the best location on earth... the beach!

Beach Buggy Racing combines the racing styles of Beach Buggy Blast and Mario kart into a fun romp on the beach. If you love racing and battles, then you'll love this game. The controllers and graphics work great on both the ODROID-U3 and C1, and up to 4 players can join in on a single monitor.

https://play.google.com/store/apps/details?id=com.vectorunit.purple.googleplay

MEET AN ODROIDIAN

ANTHONY SALTER (@VIRIDIANGAMES) DEVOTED FATHER AND ENTERTAINMENT EXPERT

edited by Rob Roy

Anthony shows us that you don't realize how tiny the U3 is until you set it next to a credit card

Please tell us a little about yourself.

My name is Anthony Salter. I'm 44, and working as a professional game developer in Austin, Texas.

Your blog, viridiangames.com, features reviews and experiences with modern and retro gaming, and has been active for over 10 years. How did you get started in gaming?

Back in 1978, my father brought home a Magnavox Odyssey 3000 video game system. It hooked up to the TV, had two joysticks wired right into the device, and played eight different versions of Pong. You chose which version you wanted by sliding a switch on the front of the console. My sister and I played it every chance we got. Later, when I was 9 or 10, my class got a TRS-80 Model III computer. Once I realized that I could write programs to make computers play games, that was about it!

What are some of your favorite games?

Woo, that's a long list. My absolute favorite games are ones that allow flexibility of play and can show you something new no matter how many times you play them. So, games like the Elder Scrolls series, the Grand Theft Auto series, and Deus Ex. But right now I'm also playing Battlefield 4, World of Warcraft and Dota 2. Basically, the only genre I don't play at all is sports games, except for NBA Jam.

You mentioned on Reddit that you have been setting up an ODROID system for your son to get more involved in computers. Can you describe your project and its goals?

I have a wonderful wife and three children, one of whom (my son) is autistic. Thinking back to my own behavior, it's entirely possible that I was on the scale myself when I was younger, but of course back then, no one knew what the scale was. Just like me, David took to computers very quickly. However, as is common with autistic children, he can get frustrated very easily, and it seems that in so many ways, modern technology is designed to frustrate.

As an example, our PS3 recently stopped working. So when he came to me holding our Blu-ray disc of The Lego Movie, I had to tell him that he couldn't watch it. When he got frustrated and asked me why, I honestly could not think of a good reason. I mean, sure, I could have told him, "Because we don't have any other devices in the house that read the proprietary data format the movie is encoded in" but he wouldn't have understood it, and that's not really a good reason.

The goal with my ODROID project is to eliminate, or at least minimize, the frustrations that legalities and platform splits create when you're just trying to watch, listen to or play something you already own. And while David was the impetus of the project, I've no doubt that the rest of my family will also appreciate having easier access to their media and games. To be specific, the device needs to be able to access video and audio files over a network file share and play any format it finds there without complaint. It also needs to access Netflix and YouTube, and play both emulated and native games. None of this is hard, per se, because media players are a very common use for single-board computers. But, I wanted to go the extra yard and ensure that David could do everything using the tools and methods that he was already familiar with, such as using a gamepad controller. I knew this was going to be a challenge and would require a lot of programming, scripting and possibly even some GUI design work.

When I started the project, I was pretty sure that I was going to use a Raspberry Pi. The Pi 2 had just been announced and seemed like the perfect platform, but I noticed some people mentioning that while the Pi 2 was a great improvement, while "the ODROID is still better". So I looked

Anthony's beautiful daughters, Megan and Jewel

up this mysterious ODROID and I'm quite glad I did! The more I researched the project, the more it exposed the Pi and Pi 2's limitations. Ultimately, I felt that the native Android support would give me all the software tools I needed to get the device going, leaving me with the controller and GUI integration, which is exactly what I wanted. I initially ordered a C1, but due to an error in the shipping address I had to cancel it. I then ordered a U3, because of its faster CPU.

My one concern about going with ODROID was the fact that the community might not be as robust, and that I might find myself without anyone to help me. I shouldn't have worried. I initially had difficulty getting Rob Roy's "Pocket Rocket" image working, and got help from Rob himself! The fact that Hardkernel's Android distributions all have native Xbox 360 controller support meant that a whole bunch of stuff worked "out of the box". Last month's issue of this very magazine even had an article about getting the Xbox 360 controller working consistently with various emulators. All that really remains is the integration, and at this point stuff works well enough that I have trouble getting my kids to stop playing on it so I can work on it!

Anthony's son, David, for whom his ambitious ODROID media center project is primarily intended

What hardware or interactivity improvements do you see in the future for gaming?

We need no-glasses 3D. VR systems like the Oculus Rift and Vive are big in the news and have a nice "wow" factor, but I don't see a family of four sitting on the couch with those headsets on in the near future. Of course, don't ask me how to do it – I have no idea!

Have you written any of your own programs or games?

Yes, I'm actually a professional game developer. I work on porting games like Borderlands 2 and Civilization: Beyond Earth to Linux. I also write my own little games on the side, because I'm crazy. You can see most of what I've been doing at my blog, though work has sadly left me little time for my own stuff recently.

Are you involved in any other projects apart from computers?

I have a few other hobbies, such as cooking and paper-and-pencil RPGs, but to be honest, my family takes up most of my time.

Do you have a wish list for future ODROID improvements?

Better graphics hardware, possibly with dedicated RAM, would be fantastic, although I realize that it may be difficult to keep a low price point.

Check out Anthony's blog at http://viridiangames.com for many years of gaming and programming related articles.