ISCSi • Cyanogenmod for U3/XU/XU3/XU4 • Overscan Fix in Ubuntu

ODROID Magazine

Year Three
Issue #26
Feb 2016

The next generation of Portable Power

The ODROID-C0 is the newest mini wonder computer

• Fire detection security system

• MUNIN system monitoring
• Windows 2000 gaming
• The OWEN Robot Kit

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

Suche - Pollin Electronic

www.pollin.de/shop/suchergebnis.html?S_TEXT=odroid&log=internal

Elektronik, Technik und Sonderposten zu Superpreisen!

mein Konto | Kontakt & Service | Versand & Bezahlen | Katalog | Newsletter | Gutscheine

Merkliste | Neu registrieren | Anmelden

Direktbestellung

0 Artikel: 0,00 € Einkaufswagen

Suchen nach: Suchbegriff oder Bestellnummer Los

Sie sind hier: Startseite > Suche

Suchergebnis Ihre Suche nach "odroid" ergab 18 Treffer

1 - 18 von 18 Ergebnissen sortieren nach: Relevanz (gewählt) Seite: 1 von

Kategorie wählen

Filtereinstellungen

[-] alle Filter entfernen

Kategorie

[+] Bausätze / Module (17)

[+] Stromversorgung (1)

Preis 1 € 80 €

Ihre Vorteile

✓ Innovative Produkte

✓ Optimales Preis-/Leistungsverhältnis

✓ Kompetente Beratung

✓ Schnelle Lieferung

Noch Fragen?

Wir sind gerne auch persönlich für Sie da! +49 (0) 8403 / 920-920

Sicher Einkaufen

ODROID-U3 BACKUP BATTERY
RTC Lithium-Backupbatterie für den ODROID XU, XU+E, XU Lite und U3. Wird direkt auf den RTC-Port des ODROID gesteckt.
2,95 €* Best.Nr. 810 319

ODROID-U3 USB-UART MODULE, Schnittstellenkarte
USB/UART System-Interface (IC: CP für die ODROID Einplatinencomput X2, U2, U3, XU, XU+E, XU Lite). I Entwicklung und Debugging.
15,95 €* Best.Nr. 810 326

ODROID-U3 Einplatinen-Computer, Cortex-A9 QuadCore, 2 GB
Kleiner als eine Scheckkarte aber riesig in der Leistung! Der ODROID-U3 verfügt über eine leistungsstarke SAMSUNG QuadCore-CPU, 2 GB Arbeitsspeicher und
69,95 €* Best.Nr. 810 264

ODROID-U3 CASE Kunstst
Passgenaues Gehäuse mit Lüftungsschlitzen für den Maße (LxBxH): 87x52x29
4,95 €* Best.Nr. 810 317

ODROID-U3 ODROID-UPS, USV, LiIon, 3 Ah
Unterbrechungsfreie Stromversorgung mit zwei LiIon-Akkus. Das Board besitzt die gleichen Außenmaße wie der ODROID-U3 Einplatinencomputer und kann direkt per
31,95 €* Best.Nr. 810 328

ODROID-U3 eMMC Android
Superschnelles N den ODROID-U3 Betriebssystem aufgespielt un
39,95 €* Best.Nr. 810

ODROID-U3 eMMC Modul, 64 GB, mit Linux
Superschnelles NAND-Speichermodul für den ODROID-U3 Einplatinencomputer. Das Linux-Betriebssystem XUBUNTU ist bereits fertig aufgespielt und kann direkt

ODROID- Linux
Supersch den OD Linux- bere
39

We are now shipping the ODROID-U3 device 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

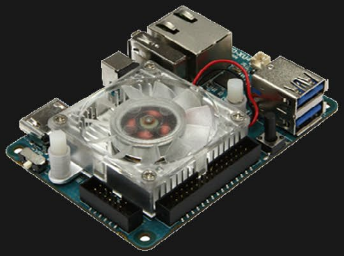
Our ODROID products can be found at
http://bit.ly/1tXPXwe

Many ODROID users loved the ODROID-W, which was discontinued in 2014. Since then, many requests were made for a similar IoT computer that could be used for industrial applications, wearables, and anything that required a minimal hardware footprint and low power consumption. Hardkernel has responded with the ODROID-C0, which has the computing power of the popular ODROID-C1 with some of the interfaces removed, so that users can build the perfect device to exactly fit their needs. An example application of the ODROID-C0 is the OWEN compact robot, designed by Bo at Ameridroid, that can walk, dance, and show facial expressions via a web interface. The ODROID-C0 is available at http://bit.ly/1WFYKOL for USD$25.

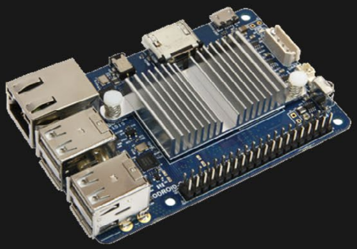
Did you know that you can run Windows 2000 on an ODROID? Tobias shows us how to install it using QEMU, so that you can run your favorite Windows programs and games! Josh introduces us to the stock Android alternative called Cyanogenmod, Ilham helps us stay safe with a robust fire detection kit, Nanik demystifies the Android Bluetooth stack, Adrian explains how to use Munin, and David details the iSCSI protocol for LVM.

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/1ypImXs.
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

ameriDroid
High-Performance Embedded Computers



ODROID-XU4



ODROID-C1+

Hundreds of products available online for the professional developer and hobbyist alike

Hardkernel's EXCLUSIVE North American Distributor

ODROID Magazine

Rob Roy, Chief Editor

I'm a computer programmer in San Francisco, CA, designing and building web applications for local clients on my network cluster of ODROIDs. My primary languages are jQuery, Angular JS and HTML5/CSS3. I also develop 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`.

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.

Andrew Ruggeri, Assistant Editor

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

Bruno Doiche, Senior Art Editor

Bruno is now used to losing to David regularly on Magic the Gathering. But he will get some gaming skills doing a quick trip to Las Vegas.

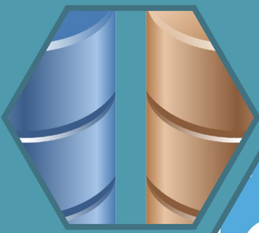
Nicole Scott, Art Editor

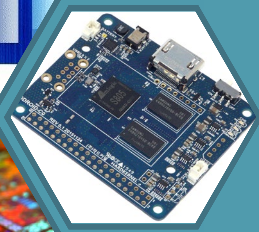
Nicole is a Digital Strategist and Transmedia Producer specializing in online optimization and inbound marketing strategies, social media management, and media production for print, web, video, and film. Managing multiple accounts with agencies and filmmakers, from web design and programming, Analytics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

James LeFevour, Art Editor

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

INDEX





ISCSI - 6

CYANOGENMOD - 8

OVERSCAN FOR UBUNTU - 9

FIRE DETECTION - 10

LINUX GAMING: WINDOWS 2000 - 13

ODROID-C0 - 18

ODROID-XU4 MANUAL - 20

RETROGAMING ARCADE - 21

MUNIN - 24

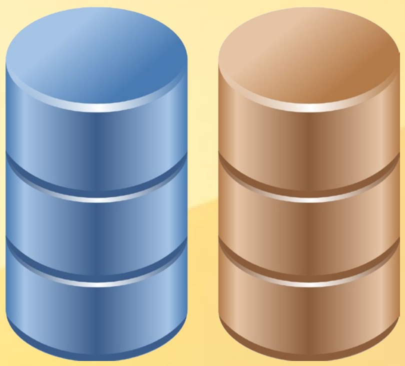
OWEN ROBOT KIT - 28

ANDROID DEVELOPMENT - 31

MEET AN ODROIDIAN - 33

INTERNET SMALL COMPUTER SYSTEM INTERFACE

ISCSI MADE SIMPLE



by David Lima

If you have one large hard drive, and would like to share its available space with another ODROID, or want to setup a file server and consolidate all storage in a single point, the first thing you may think to use is Network File System (NFS). However, there are security problems with NFS, since it does not inherit the permissions properly, and you probably don't need the data shared with a bunch of clients. In this article, I will demonstrate an alternative, which is to use iSCSI.

iSCSI, which is an acronym for Internet Small Computer Systems Interface, is an Internet protocol used to link data storages across facilities. With iSCSI, you can create a virtual disk on a server and map it to a client as a block device and do whatever you want with it, like creating a file system of any type or using it as a swap area as if it was a physical local drive. Instead of sending remote procedure calls like NFS does, iSCSI will send SCSI instructions over the network. These are the same set of instructions your system uses to write data blocks to your disk drive.

This way, you have better security, since you can set the permissions normally and don't have to worry about "root squashing" in order to prevent root account access to exported file systems. You can also improve performance since the system skips one layer of interaction and just sends a write SCSI request.

Background

The virtual disk we are going to create is called LUN (logical unit number). The LUN is actually the number used to identify the virtual disk, but is is also often used to talk about the disk itself. There is also the SCSI initiator and SCSI target. SCSI target will act as a server, where the storage device is located, and the SCSI initiator is the client, which is the one receiving the virtual disks. You can think of the initiator as being a SCSI bus adapter card, but instead of cables, it will be using your network. First, install the required dependency on target:

```
$ sudo apt-get install tgt
```

Then, do the same for the initiator:

```
$ apt-get install open-iscsi
```

Starting on the target side, we have to create a target to be accessed by the clients:

```
$ tgtadm --lld iscsi --op new \
  --mode target --tid 1 \
  -T iqn.2016-01.com.server-
iscsi:vdisk.1
```

The -T option specifies the initiator address, which is how the initiator will see the LUN you create. The initiator address format is iqn.yyyy-mm.<reverse_domain_name:LUN name, where iqn means iSCSI qualified name, which his is the addressing name-format type. The date (yyyy-mm) specifies when the naming authority took ownership of the domain, which is required whenever the iSCSI will be used over the internet. For local networks, this can be any date. The reverse domain name of the target server is also used for Internet-specific configurations, and can be any name on local networks. The LUN name can be any label that will be easy to identify by the administrator, such as client name and purpose. You can have multiple targets on the same server by changing the target id(tid) and iqn.

Check the configuration so far with the following command:

```
$ tgtadm --lld iscsi \
  --op show --mode target
```

You will see that LUN ID 0 is already attached. This is for internal control of the utility and does not need to be changed. Next, we have to assign a block device to create the LUN for our client(s). For this, we have three options:

1. Use a regular partition from any device attached to the system
2. Create a logical volume using LVM to attach it to the target
3. Create a data file on a mounted file system to use as if it was a block device

I would recommend option 2 so that you can take advantage of all LVM features for your target LUN. Option 1 is also good if you don't want to use LVM, and option 3 is useful for testing purposes when you want to configure an iSCSI and don't want to change partition tables or logical volumes, but is not recommended for long-term settings.

For the purposes of demonstration, I will choose option 3. To create the storage file, type the following commands, which requires root privileges:

```
$ su
# dd if=/dev/zero \
  of=/home/odroid/iscsi-lun \
  bs=4096 count=1024000
```

This will create a 4GB file, which can then be attached to a new LUN:

```
# tgtadm --lld iscsi \
  --op new --mode logicalunit \
  --tid 1 --lun 1 \
  -b /home/odroid/iscsi-lun
```

Next, check your newly created LUN:

```
# tgtadm --lld iscsi \
  --op show --mode target
```

If you are creating the LUN with an actual block device, just replace the file name with the device path, such as /dev/sdb1 or /dev/rootvg/homelv. We are now ready to allow clients to bind to this target:

```
# tgtadm --lld iscsi --op bind \
  --mode target --tid 1 -I
192.168.0.2
```

This will allow only clients with IP 192.168.0.2 to login and bind to this LUN. You can also replace the IP with "ALL" so that any initiator can bind to this target, although this is not recommended because of security issues. With the "ALL" option, anyone on the same network would be able to read data from this target's LUN, and if you login to it from two different initiators and they starting writing to it, the whole thing may become corrupted. Even if you have a cluster solution and need the LUNs to be shared, make sure to specify the IP addresses instead of just setting it to be viewable by all.

This configuration will only be retained until you reboot your server. If you want to persist the changes through a reboot, dump the configuration to /etc/tgt/conf.d as any configuration file under this path is automatically included:

```
# tgt-admin --dump | \
  grep -v default-driver > \
  /etc/tgt/conf.d/targets.conf
```

Moving back to the initiator, we need to scan for new available LUNs. To do so, use the following command:

```
# iscsiadm --mode discovery \
  --type sendtargets \
  --portal 192.168.0.1
```

You will see the LUN created back on the target, which can now be logged into:

```
# iscsiadm --mode node --target-
name \
  iqn.2016-01.com.server-
iscsi:vdisk.1 \
  --portal 192.168.0.1:3260
--login
```

If you want to initiator connect automatically after a reboot, edit the following file:

```
# vi /etc/iscsi/iscsid.conf
```

and change this line from manual to automatic:

```
node.startup = automatic
```

Now, if you type the command "fdisk -l", the target LUN will be visible as a new device, for example /dev/sdb. All you have to do then is to create a file system of your preference on it or add it to your Logical Volume Management (LVM) list.

ODROID Magazine is now on Reddit!
ODROID Talk Subreddit
http://www.reddit.com/r/odroid

A LOOK AT CYANOGENMOD

GETTING STARTED WITH A CLEAN, LIGHTWEIGHT FLAVOR OF ANDROID

by Joshua Sherman

Android has stormed the world in the last nine years, invading devices of all shapes and sizes, including our ODROIDs. While Android itself comes from Google, the open source, Linux-based operating system has countless versions designed and modified by smartphone manufacturers, each with their own unique take on the OS. CyanogenMod is one such version, designed by a community of more than 50 million people, and is available for your ODROID.

What is CyanogenMod?

Since Android is an open source operating system, versions are available for anyone to compile and use through the Android Open Source Project (AOSP). Thousands of devices are compatible, including most ODROIDs with versions compiled by Hardkernel. While many enjoy the clean, lightweight experience of AOSP, it varies from device to device and can lack some features and fine tuning you may want.

This is where CyanogenMod comes in. CyanogenMod is a third party version of Android designed and compiled by a community of over 50 million users all over the world. At its core, CyanogenMod is the same, bare bones AOSP that's available to everyone. The CyanogenMod community then fine tunes it, adds some useful features, and packages it all into a clean, enhanced UI.

Best of all, CyanogenMod stays true to Android's core philosophy by remaining an open source project under the Apache license. This makes CyanogenMod available for anyone to compile and build for new devices, like an ODROID, while giving each device the same experience. This is in contrast to most versions of Android, which often come included with licensed, third party software. For example, Samsung's TouchWiz UI is licensed to Samsung devices only, and can't be shared to new devices without their permission.

Some of the features that CyanogenMod adds to AOSP Android include a fresh UI, improved performance, USB tethering, VPN connectivity, FLAC audio support, native theme support, and a host of other benefits. CyanogenMod is also pre-rooted for access to advanced Android functionality. Check out the full details at http://bit.ly/1nk2aKz.

What CyanogenMod versions exist?

Community member @voodik helps maintain versions of CyanogenMod for use on ODROIDs. There are currently two major versions of CyanogenMod that are up to date and available for ODROIDs:

CyanogenMod 13 is the latest and greatest version available, which is based on Android 6.0.1 Marshmallow, the most recent Android OS available. It has all the latest features that come with Android 6.0.1, such as advanced power and permission controls. The CM13 builds for ODROIDs do not yet support USB 3G modems or Bluetooth adapters, so it would be better for those those who require that type of connectivity in their projects to use CM12 instead.

CyanogenMod 12.1 is the tried-and-true version based on Android 5.1 Lollipop. Similar to Lollipop, It's about a year old, which means that it's had some more time to go through bug fixes and optimization. It supports most peripherals, including Bluetooth and 3G modems, and is a good choice for applications that have compatibility issues with Android 6.0.1. You will, however, miss out on some of the new permissions and power features in CM13.

CyanogenMod 11 also exists on some ODROIDs, but you will probably prefer CyanogenMod 12 or 13 unless there is a specific reason you need to stay on CM11. There are also some flavored builds of CyanogenMod designed for Android TV, which offer a slightly different user experience.

Remember, there's no "best version", since it comes down to using what suits your needs. In the end, they're all designed around the same core experience: giving users a fast, clean, open source version of Android.

Getting started with CyanogenMod

Once you pick whichever version of CyanogenMod you're interested in, you can go ahead and check them out in the respective forums where they're available. Version availability can vary from device to device, so keep that in mind when seeing what's available for your device:

ODROID-XU3/XU4:

`http://bit.ly/1niFjiA`

ODROID-U3:

`http://bit.ly/1WIQYn8`

ODROID-XU:

`http://bit.ly/1KyX4Q5`

Just like installing Android or Ubuntu, CM12 and CM13 follow more or less the same procedure to get up and running. Then all you need to do is explore the operating system to see if it's what you're looking for in an OS.

Notes

CyanogenMod is 100% open source, which means it won't include any Google applications by default, since they're considered closed source. This includes the Google Play Store, which you've probably used to for downloading applications. Don't worry, you can install the Play Store on your own through a GApps package in each CyanogenMod forum topic, or use the Universal One-Click Installer available from Hardkernel at http://bit.ly/1nL6ymp. It includes all of the usual Google apps so that you can use the Play Store if you wish, or stick to the core open source experience if that's what works for you.

OVERSCAN FIX IN UBUNTU FOR ODROID-C1/C1+

SYNCHRONIZE YOUR ODROID WITH YOUR MONITOR

edited by Rob Roy

If the display used on your ODROID-C1/C1+ shows a slight cropping of the visible image on the screen, you may be experiencing overscan. This is not an uncommon problem, and especially so for LCD TV monitors. The fix is usually a simple one, and the underlying issue is most likely due to a setting with the LCD monitor. Some PC Monitors with HDMI inputs will also apply overscan to the HDMI input, assuming that a broadcast TV signal is being used.
Monitors that are used for broadcast television usually have overscan enabled by default. This is a normal feature of TV monitors and has been present from the very beginning of television. Overscan is used to crop the edges of the video frame in order to remove any erratic or distorted edges that often exist with broadcast video. To the viewer, this results in a cleaner picture, and the overscan simply isn't noticed. For a computer display however, this can be an issue. For this reason, computer LCD monitors usually do not provide overscan, and if they do have this feature, it is disabled by default.
In order to accommodate monitors that may not have an overscan option built into the hardware, the steps below will allow you to enable software overscan in Ubuntu for the ODROID-C1/C1+.

1. Get and Set the overscan values of left, top, right and bottom using trial and error method, by typing the following commands into a Terminal window:

```
$ su
# cat /sys/class/graphics/fb0/win-
dow_axis window axis is [100 100
1919 1079]

# echo 100 100 1919 1079 > /sys/
class/graphics-/fb0/window_axis
# echo 0x10001 > /sys/class/graph-
ics/fb0/free_scale
```

2. Create a BASH script called overscan.sh, as shown below, that includes your left, top, right and bottom values:

```
# cat overscan.sh
#!/bin/bash

echo 100 100 1919 1079 > /sys/
class/graphics/fb0/window_axis
echo 0x10001 > /sys/class/graph-
ics/fb0/free_scale
```

3. Save the script to the /etc/init.d directory so that it automatically starts on boot:

```
# cp overscan.sh /etc/init.d/
# update-rc.d overscan.sh defaults
```

If you have questions, comments, or suggestions, please visit the original article at http://bit.ly/236QKum.

FIRE DETECTION FOR SURVEILLANCE CAMERAS

TAMING FIRE WITH A WEBCAM AND AN ODROID

by Ilham Imaduddin

Our ancestors tamed fire's power hundreds of thousands of years ago. However, fire is one of our most dangerous tools, and can become lethal when it's out of control. Today fires can be caused by appliance malfunctions, electrical failure, or even a lightning strike. In 2014, there were almost 500,000 structure fires in the US alone, which caused thousands of civilians injury and deaths.

I hope those facts warmed you up, because today we will protect our homes from a fire. In this tutorial, we will create a surveillance application which will detect a fire and notify us. Just imagine you are in the middle of a boring meeting and no one is home. Instead of getting a call from the fire department telling you your house is gone, this program will notify you when the fire initially starts, thus giving you time to save your house!

To do all of that, we will write a simple Python script using the OpenCV module. OpenCV is a popular image processing library that packages many image processing algorithms into an easy-to-use interface. This tutorial will show you how an ODROID board can be much more than a daily computing device.

The hardware

You will need an ODROID such as the XU4 or C1+, a USB webcam, and an Internet connection to your board. For my setup, I used an ODROID-C1, a Logitech C525 camera, and a generic WiFi module.

The software

This tutorial will go over a program written in Python 2. The Python program requires three modules: the cv2 OpenCV module which is used for computer vision, NumPy which is required by cv2 and handles large matrices and arrays, and the envelopes module, which assists with email handling. There are several excellent guides on the ODROID forums at http://bit.ly/1E66Tm6 that detail how to install OpenCV. You can also download a customized version of ROS that already includes OpenCV at http://bit.ly/1JvIxK1 (XU3/XU4) and http://bit.ly/1ZLMpID (C1/C1+). An installation guide for NumPy is available on their official site at http://bit.ly/1KpXV5B. You can install the envelopes module with the steps shown below.

First, if pip is not installed, it can be installed with the following apt-get command:

```
$ sudo apt-get install python-pip
```

Then, install envelopes:

```
$ pip install envelopes
```

Finally, in the Python code, import the three modules into the application with the following lines:

```
import cv2
import numpy as np
from envelopes import Envelope
```

Taming fire

Before we can detect a fire and flames with our camera, we need to know the specific properties of a fire that make it distinct from other objects the camera sees. A flame has at least two properties that make it unique and easy to work with which are intensity and color. The plan is to filter by both intensity and color in each frame, then merge both filters to make a new "fire mask". To make sure you understand the code, you should be familiar with OpenCV.

Intensity filter

Fire is bright, and often it is often the brightest object in the image frame. This means that we can identify fire by selecting pixels with highest intensity levels. To do that, we will convert the captured webcam frame from BGR color space, which OpenCV uses as default, to YCrCb. Once the image is in the YCrCb space, we can filter by the Y component, which is the luminance value.

```
def filter_intensity(frame,
threshold_y):
    ycrcb = cv2.cvtColor(frame,
cv2.COLOR_BGR2YCrCb)
    threshold_low =
np.array([threshold_y, 0, 0])
    threshold_high =
np.array([255, 240, 240])

    return cv2.inRange(ycrcb,
threshold_low, threshold_high)
```

The cv2.cvtColor method is used to convert a frame from one color space to another. In OpenCV, an image is represented as a NumPy array, and so we create the threshold arrays using np.array. After creating the threshold arrays, the filtering process is done with the cv2.inRange method, which takes the frame as the first parameter and thresholds as the second and third parameters. The method then returns a filtered mask based on the parameters.

Color filter

Fire is not necessarily always pure red. Sometimes it's a mix of orange or even white. Regardless of the mixture, the primary color component of fire is red-based. This means that we can filter the frame by selecting pixels with a high red value as a way to identify any flames. Filtering for red given a frame in the BGR color space is easy and is shown in the following code:

```
def filter_color(frame,
threshold_r):
    threshold_low = np.array([0,
0, threshold_r])
    threshold_high =
np.array([threshold_r-20,
threshold_r-10, 255])

    return cv2.inRange(frame,
threshold_low, threshold_high)
```

The threshold_high is set to make the blue and green components smaller than the red component. Just like with the intensity filter, the color filter process is done with cv2.inRange method, which take the frame and thresholds as parameters and returns a filtered mask.

Extract the fire

Now that we have both the intensity and color filtering done, we can start to extract fire from a frame. First, we will make an intensity mask and then a color mask by calling the filter_intensity and filter_color functions respectively. Then, we do some processing to improve the quality of both masks. After that, we merge both masks together to create a new "fire" mask.

```
def extract(frame, threshold_y,
threshold_r):
    intensity_mask = filter_
intensity(frame, threshold_y)
    color_mask = filter_
color(frame, threshold_r)

    kernel = np.ones((5, 5),
np.uint8)
    intensity_mask = cv2.
dilate(intensity_mask, kernel,
iterations=1)

    kernel = np.ones((10, 10),
np.uint8)
    color_mask = cv2.
dilate(color_mask, kernel, itera-
tions=1)


    return cv2.bitwise_and(color_
mask, color_mask, mask=intensity_
mask)
```

The problem with the filter_intensity and filter_color function is that sometimes they may return a broken mask. We can apply the cv2.dilate method to fix this. Dilate increases the area of the mask, filling the gap between the broken masks.

Wrapping up

Let's start by initializing the camera in code, then make a loop to continually get new frames from the camera. Within this loop, we will not only be getting a new frame from the camera, but will be processing that frame as well. The first this thing we will do after receiving a new frame is to call the extract function that we previously created:

```
cam = cv2.VideoCapture(0)
counter = 0

while (True):
    frame = cam.read()[1]
    fire_mask = extract(frame,
250, 230)

    # Fire handling

    if cv2.waitKey(1) & 0xFF ==
ord('q'):
        break

cam.release()
```

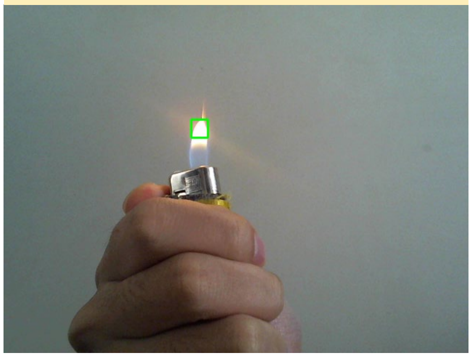
We attach the camera by creating a VideoCapture object. Its argument is the camera index, which will be 0 if only one camera is connected. You can select the second camera by passing 1, and so on for additional cameras. In the loop, we read a frame from camera, then filter it for any flames. Finally, we apply some additional code to help minimize false positives.

```
# Fire handling
contours = cv2.findContours(fire_
mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[1]

if len(contours) > 0:
    for i, c in
enumerate(contours):
        if cv2.
contourArea(contours[i]) > 100:
            x, y, w, h = cv2.
boundingRect(contours[i])
            cv2.rectangle(frame,
(x, y), (x+w, y+h), (0, 255, 0),
2)

    counter = counter + 1
    if counter == 100:
        # Send Notification
else:
    counter = 0
```

The first method being used, cv2.findContrours, finds any contours, as the name implies. Contours can be simply defined as "an outline, especially one representing or bounding the shape or form of something." The cv2.findContrours takes a frame, as well as a few parameters for adjustments, and returns an array of all the contours found within that frame. When we pass the cv2.findContrours function our fire mask, we expect the mask to be empty, meaning no there will be no shapes or contours. If there happens to be a flame or some fire, the mask would not be empty and we would get a returned array with the flame contour.

If a contour is returned, we check whether the contour is large enough to be considered fire. This is important because it helps to minimize false positives of the application reporting a fire when there is none. Finally, if the detected contour is large enough, we draw a rectangle around it in the frame. An example of a detected flame is shown in Figure 1.

Figure 1 - Demonstration of fire that will be detected by the camera

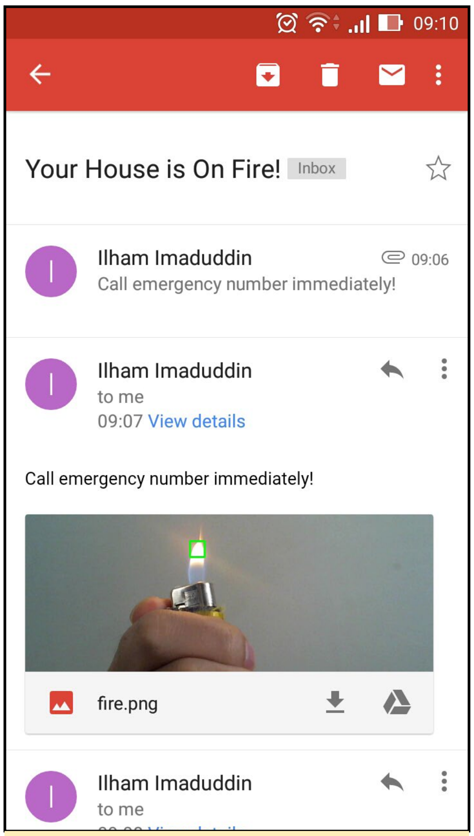


The counter feature is also used to help minimize false positives. Instead of sending an alert immediately after the camera's first frame which detects fire, the program will wait for several frames to positively identify a fire before sending an alert. If the detected fire lasts long enough, we will save that image, and send it along with the email alert.

```
# Send Notification
cv2.imwrite('fire.png', frame)

mail = Envelope(
    from_addr=(u'from@email.com',
u'Name'),
    to_addr=(u'to@email.com',
u'Name'),
    subject=u'Your House is On
Fire!',
    text_body=u"Call emergency
number immediately!"
)
mail.add_attachment('fire.png')
mail.send(
    'your.smtpserver.com',
    login='your@email.com',
    password='your password',
    port=587,
)

print "Email sent"
break
```

The image is saved to disk with the cv2.imwrite method, which takes the filename and the frame as parameters. Using an Envelope object, we send an email notification that a fire was detected, and include the saved image. For the sake of simplicity, after we send our email alert, we break from the loop in order to end the application.

If you are sending from a Gmail address, there is a setting which prevents logins that don't meet certain security requirements. You will need to disable this setting in order to send an email using Envelope.



09:10

Your House is On Fire! Inbox

Ilham Imaduddin 09:06
Call emergency number immediately!

Ilham Imaduddin
to me
09:07 View details

Call emergency number immediately!

fire.png

Ilham Imaduddin
to me

Figure 2 - Notification email from fire detection program

Burn

Now, it's time to put our program to test. Since I'm not crazy enough to light my house on fire just for the sake of testing, we will not see my house being burned today. Instead, we will use a lighter for our flame source. Figure 1 shows the alert email that I receive when I test the program.

If you decide to test it yourself, you might need to change the threshold values. They are set to an ideal threshold best suited for my house. The values may need to be changed due to lighting differences in the areas being monitored.

Expansion

There are a lot of possibilities for extending this basic application. For example, we can record the fire, send SMS instead of email, or even call emergency services automatically. We could also create another algorithm for our surveillance camera, such as movement detection or face detection.

LINUX GAMING

RUNNING WINDOWS 2000 GAMES ON AN ODROID

by Tobias Schaaf

Currently, we are able to play thousands of games on the ODROID through emulation, including many console games such as SNES, PS1, N64, and more. If we are playing a game originally published for the PC, we still have a few options. A large number of DOS-based games may be run on the ODROID using the DOSBox emulator. Besides that, there are native ports for Linux from older Windows DOS games such as Arx Fatalis, Homeworld, Jedi Knight 2 and 3, Quake 3, and many others.

Still, if we want to play our old Windows games, the number of games we can play is somewhat scarce since only a few of them have actually been ported to Linux. In this article, I want to outline what other options are available, as well as find out if we can get a Windows system to work on our ODROID.

This guide is mostly for "trying things out", and it will not give you a full-featured fast Windows machine on your ODROID. The performance of this setup, which is a Windows 2000 virtual machine (VM) running in QEMU, is very low, and certain tasks will take a long time. If you read this in the hope to run Battlefield 3 directly on an ODROID, this guide is not for you. It's only a study on what's possible and how far we can go.

Requirements

Since ARM CPUs are not x86 CPUs, running an x86 operating system on ODROID will require some preparation, so we need to install a few programs on our system and set them up before we can try to install Windows and any Windows programs:

- QEMU Emulator for emulating an x86-compatible PC on our ODROID. QEMU comes in many different versions and is constantly being improved. I use the version of QEMU from my own repository, which is currently at version 2.5.0.
- The ODROID-XU4 offers the fastest CPU at the moment, and is required to get the best possible experience. An ODROID-U3 would probably work as well, but will definitely be slower. An ODROID-C1 can not be used for this, as it simply lacks the CPU speed and adequate RAM required for Windows 2000 emulation.
- At least 4 GB of free space on SD, eMMC, or an external storage.
- Windows 2000 install CD/ISO. Generally all Windows versions would work, but in previous tests I found that Windows 95, while still working, lacks good DirectX support, which prevents most games from starting. Windows 98, which does have DirectX, lacks sound support and won't be able to run properly because of this limitation. The QEMU project website states they no longer support any Windows OS below Windows 2000.
- Some Windows games and programs for testing.
- A lot of time, since the installation process can take several hours.

Installation

First, we need to install the required software, set it up, and then install Windows 2000 in a emulated x86 environment provided by QEMU.

```
# apt-get install qemu-ODROID
```

We start by creating a folder in which we will work. I decided to create a folder called Windows in the home folder of the user ODROID and place everything I need there, so I copied over the ISO of my Windows 2000 installation CD to that directory. Then, I created a new disk image for our Windows 2000 installation:

```
# qemu-img create -f qcow2 win2k.
img 10G
```

A qcow2 image slowly grows in size and does not allocate 10GB right from the start like the RAW format does. It also offers advanced features such as snapshots. Windows itself only needs about 1-2GB of space. Anything between 3-4 GB should be enough. After that, I created a launch script that allowed me to start the QEMU virtual machine more easily. I made it executable and placed it in the Windows folder:

```
# /home/ODROID/Windows/start_qemu
#!/bin/bash
export QEMU_AUDIO_DRV=pa
qemu-system-i386 -hda ./win2k.img
-cdrom ./win2k.iso -boot d -m 512
-localtime -soundhw ac97 -moni-
tor stdio -net nic,model=ne2k_pci
-net user -vga std -cpu pentium
-no-ahci
```

The parameters are explained below:

QEMU_AUDIO_DRV=pa

This parameter specifies using pulseaudio as the default sound driver. It works with ALSA as well, but PulseAudio seems to be faster with QEMU than ALSA. Other available options are "sdl" and "alsa".

qemu-system-i386

This tells the system that this is an entire i386 compatible system with everything that belongs to it such as BIOS and hardware. There's also a qemu-i386 binary, which can be used to start a single i386 binary instead of emulating an entire PC, but we won't be using this.

-hda ./win2k.img

This specifies that our hard drive image is the win2k.img file that we created earlier.

-cdrom ./win2k.iso

Similar to the hard drive image parameter, this specifies the ISO file of the Windows 2000 install CD.

-boot d

This tells the system to boot from drive D:, which in this case is our Windows 2000 ISO. The Windows 2000 ISO should be a self-booting image. With earlier versions of Windows, such as Windows 95, you often needed a floppy disk to boot and load a CD-ROM driver in order to access the CD and install from it. With later versions of Windows, this was no longer necessary, which is why we can boot directly from CD instead. Once installed, it will also skip the CD and boot from the hard drive unless you press a key.

-m 512

This parameter gives the virtual machine 512MB of RAM for use. You can use 768MB or 1024MB, but you might have to activate swap or zRAM for this, since it uses more than 1024 MB of actual RAM. You can also probably only use 128 or 256MB instead, since Windows 2000 will work just fine with less RAM too.

-localtime

This tells the virtual machine to use the time from the host system.

-soundhw ac97

This uses the virtual Intel 82801AA AC97 audio sound card. There are other models as well, but AC97 should work fine.

-monitor stdio

This tells the virtual machine to include a command line console, which may be used for changing virtual machine parameters such as mounting other CDs or disk images.

-net nic,model=ne2k_pci -net user

This is the network configuration, which I did not actually test.

-vga std

Use the standard VGA with Bochs VBE extension, which is an advanced graphics cards offering more features than the default option. Alternatively, for optimum compatibility, albeit with less features, you could use the "-vga cirrius" option, which uses a virtual Cirrius Logic GD5446 SVGA graphics adapter.

-cpu pentium

This specifies the CPU that should be used. There are different types available, like pentium, pentium2, pentium3, and 486. We use pentium2 for installation, but later you might want to switch to qemu32, which was developed as a 32-bit CPU directly for QEMU, in order to use advanced settings such as "-smp 2", which creates a PC with 2 CPUs. You can also more specific with options such as "-smp 2,cores=2,threads=1,sockets=1". By using "qemu-system-i386 -cpu ?", you can get a list of all available CPU models.

-no-acp

This option disables ACPI (Advanced Configuration and Power Interface) support, since this can cause issues during hardware detection

Installing Windows 2000

After you have copied your win2k.iso to the same folder as the start script and created the win2k.img in the same folder, you can start QEMU with the following command after navigating to its folder:

```
$ ./start_qemu
```

A new window will pop open, and you will see the Windows 2000 setup manager.

The partitioning and copying of the installation files for Windows will go

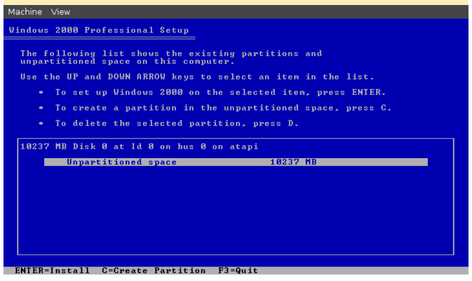
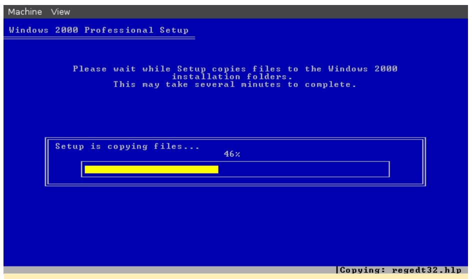


Figure 1 - Selecting the install partition of our Windows 2000 installation

```
Windows 2000 Professional Setup

  The following list shows the existing partitions and
  unpartitioned space on this computer.

  Use the UP and DOWN ARROW keys to select an item in the list.

     •  To set up Windows 2000 on the selected item, press ENTER.

     •  To create a partition in the unpartitioned space, press C.

     •  To delete the selected partition, press D.

 10237 MB Disk 0 at Id 0 on bus 0 on atapi

        Unpartitioned space                10237 MB

 ENTER=Install  C=Create Partition  F3=Quit
```



Windows 2000 Professional Setup

Please wait while Setup copies files to the Windows 2000 installation folders.
This may take several minutes to complete.

Setup is copying files...
46%

Copying: regedt32.hlp

Figure 2 - Copying the setup files

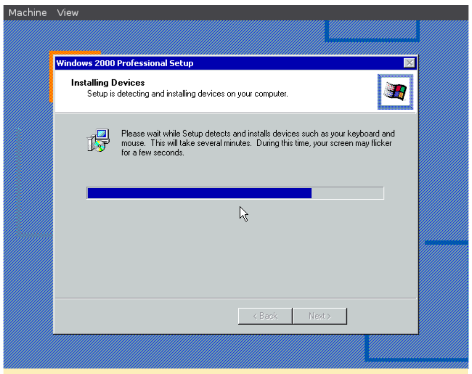
Machine View
Microsoft
Microsoft Windows 2000 Professional
Built on NT Technology
Starting up...
Copyright © 1985-1999 Microsoft Corporation

Figure 3 - The Windows 2000 boot screen brings back good memories of a very fast and stable OS which I used for many years

relatively quickly, and will be done in a couple of minutes. Afterwards, the system will reboot and setup the system.

This part will take rather long, so sit back and let the ODROID do whatever it needs to do. You can probably go watch a movie or a couple episodes of your favorite TV show. The CPU of the ODROID will not be used much during this process, so you can use your ODROID for other tasks such as watching a movie, playing music, or browsing the web. Keep in mind that if you have chosen a high value of RAM for the virtual machine, you might not have a lot of RAM to spare, so browsing websites like YouTube or other sites could use up what's left of the RAM very quickly. I'm still not sure what causes this long delay, but you can see the same behavior on all x86 emulators like DOSBox, QEMU, and ExaGear. They all seem to be really slow at reading and writing while the CPU doesn't seem to be 100% utilized. After a while, the system will start installing drivers and try to auto-detect the hardware of the QEMU system.

After a while, the hardware detec-



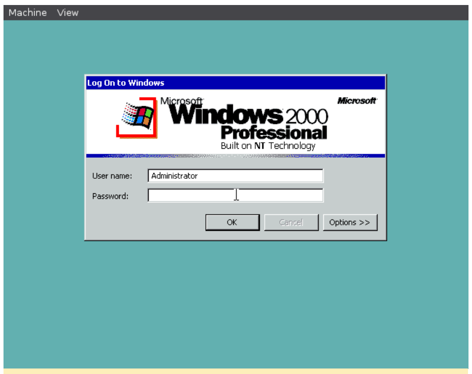
Windows 2000 Professional Setup

Installing Devices
Setup is detecting and installing devices on your computer.

Please wait while Setup detects and installs devices such as your keyboard and mouse. This will take several minutes. During this time, your screen may flicker for a few seconds.

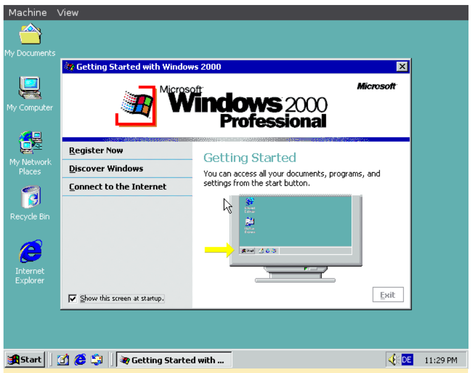
Figure 4 - Auto detecting hardware in QEMU environment can be a really long process

tion will be completed, and the system will ask a couple of questions about keyboard layout, network settings, administrator password, and so on. Follow the prompts and wait for it to be completed, which can take over an hour. Be patient, and after another reboot, you will finally be presented with the login screen followed by the lovely Windows 2000 intro sound and desktop.



User name: Administrator
Password:

Figure 5 - Windows 2000 Login Screen



You can access all your documents, programs, and settings from the start button.

Figure 6 - Windows 2000 Desktop on first boot

At this point, we have a Windows Desktop running Windows 2000, but if you look closely, you will notice that the current screen is only in 640x480 and 16 colors, so let's try to fix that by copying additional drivers into the system. I use 7zip and Universal VESA/VBE Video Display Drivers for Windows NT architecture from http://bit.ly/200rAyC. Both programs can be found in my repository under http://bit.ly/20pYup0. There are a couple of different ways that we can get these files into the system. I want to discuss two of them, since they work on nearly every system and can be useful for other purposes as well. The first is to create your own ISO file and put the files you need into an ISO, then mount the ISO in QEMU. The second is to mount the QEMU hard disk file directly into your Linux environment in order to perform file operations on it directly.

Creating an ISO file

If we want to change the ISO file that's in the CD drive of our QEMU virtual machine, we first need to create a new .ISO file with the files that we want to see in our system. To accomplish this, we're going to install a program called "genisoimage", which may already be include in your Linux distribution.

```
# apt-get install genisoimage
```

Create a new folder such as "myfiles" on your hard drive, then copy all of the necessary files into that folder. You can also put the games and programs you want to install later into the same folder. Then, use the following command line to create an ISO file out of that folder:

```
$ Genisoimage -o myiso.iso \
  -V MYISO -r -J myfiles/
```

This command will generate files called myiso.iso with the files and folders included in your "myfiles" folder, making sure to replace the name "myfiles" with the name of the folder you created. Next, we need to mount that ISO file into our QEMU virtual machine. We already know that we can define what

ISO file to use with the command line script that we created, and would only need to replace the the ISO file in that command line, but I want to demonstrate a different method. When we created that command line, we added a "monitor stdio" parameter, which allows us to input commands from the terminal to control the QEMU virtual machine.

If you haven't noticed this yet, you should be able to navigate to the terminal window that you used to start your QEMU virtual machine, which should display a "(qemu)" prompt that indicates that it is waiting for a command. Click in that terminal window, then press ENTER once or twice to see if it's working. If your mouse is hooked into your Windows virtual machine, just press "CTRL + ALT + G", and you well get your mouse cursor back and can go to that terminal. If you type "help" here, you will get a list of different options for interacting with the virtual machine. We only want to mount a new ISO, which can be done with the following command:

```
(qemu)change ide1-cd0 /home/
ODROID/Windows/myiso.iso
```

Replace the path of the ISO file with the file you created in the steps above. Check that the ISO file is correctly in place with the following command:

```
(qemu)info block
```

You can even be a little bit more gentle on the system, and rather than just switching the CDs directly, you can eject the CD first:

```
(qemu)eject ide1-cd0
(qemu)change ide1-cd0 /home/
ODROID/Windows/myiso.iso
(qemu)info block
```

After you've done that opened "My Computer" on your Windows 2000 virtual machine, you should see the CD you created and can interact with the image.

Mounting the hard disk file

Creating an ISO file is very flexible and allows you to quickly change programs on the ISO file. It is also rather fast, so in general it's a good thing to do. However, a downside of the ISO option is that a CD is always write protected, which means that the files that are on the CD cannot be extracted directly on the CD, and you always have to copy your stuff over to the hard drive first. Since it's coming from a CD, everything that you copy over will automatically be flagged as write-protected or read-only since it was the same on the CD, so you have to update the permissions before using the files.

Another approach is to copy the files directly on the hard drive image of your QEMU system. Although this can be done while the system is running, it's better to do this when the system is turned off, so shutdown your Windows system, or close the window directly, and open a root terminal.

If you created a harddrive image in RAW format, you can probably directly mount the image using a command like this:

```
# mount -o loop,offset=32256
win2k.img /mnt
```

After that, you should be able to access the disk image under the /mnt directory and perform file operations on it like creating folders and copying files. Don't forget to unmount the image after you're done:

```
# umount /mnt
```

If you created a qcow2 image, or the previous attempt did not work for you, there's a different approach you can try:

```
# modprobe nbd max_part=16
# qemu-nbd -c /dev/nbd0 ./win2k.
img
# partprobe /dev/nbd0
# mount /dev/nbd0p1 /mnt
```

After that, you can also perform disk operations on /mnt, which will hold the contents of your hard drive image. After you're done, remember to unmount the device and also close the connection to the hard drive image:

```
# mount /mnt
# qemu-nbd -d /dev/nbd0
```

Fixing the graphics

No matter how you have copied the files into your system, you should end up with the 7zip installer available, as well as the vbempj.zip file on your Windows 2000 virtual machine. The next step is to install 7zip, then extract the zip file. After that has been completed, right-click on "My Computer" and choose Properties. In the new window that opens, we can select the Hardware tab and then click the Device Manager button.

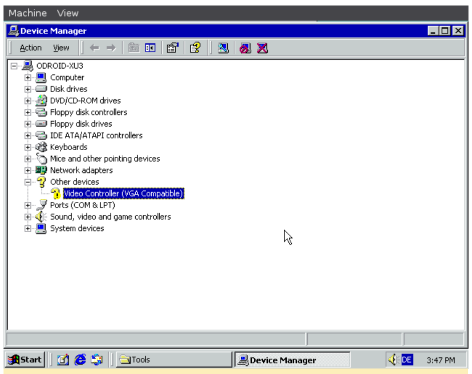
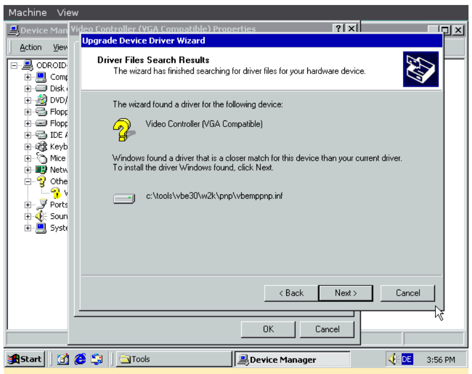


Figure 7 - Missing VGA drivers for "std" graphics card of QEMU

You should see a question mark of a standard VGA adapter, as shown in Figure 7. Double-click it to open the properties window, then click the Reinstall Driver button. When asked, select "Search for a suitable driver for my device (recommended)", followed by the Next button. Check the "Specify a location" option and select Next again. A window will pop up, asking where to search for the driver. Click the Browse button and navigate to the folder where you extract-


Driver Files Search Results
The wizard has finished searching for driver files for your hardware device.

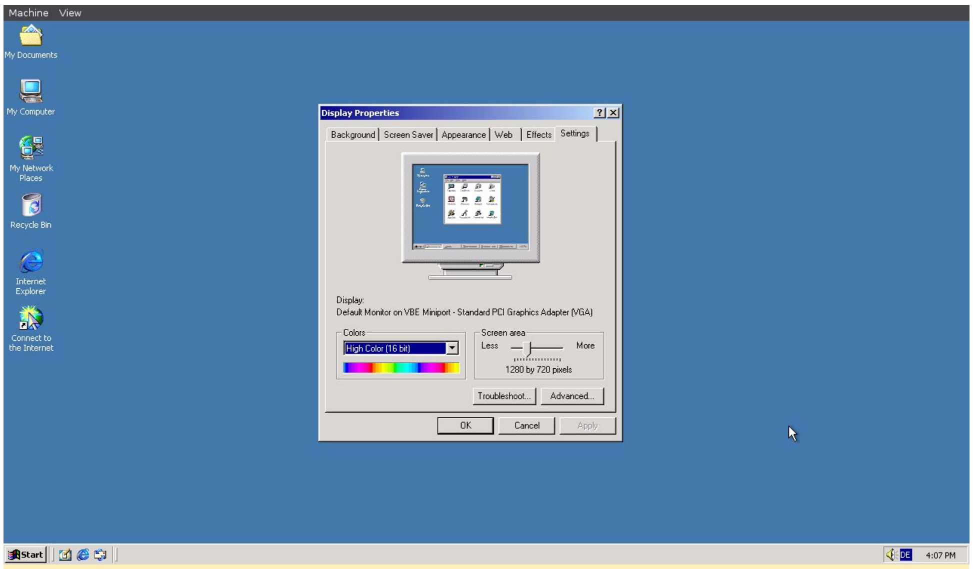
The wizard found a driver for the following device:

Video Controller (VGA Compatible)

Windows found a driver that is a closer match for this device than your current driver. To install the driver Windows found, click Next.

c:\tools\vbe30\w2k\pnp\vbemppnp.inf

Figure 8 - Windows found the new VGA drivers for the QEMU graphics adapter


Display:
Default Monitor on VBE Miniport - Standard PCI Graphics Adapter (VGA)

Colors: High Color (16 bit)

Screen area: 1280 by 720 pixels

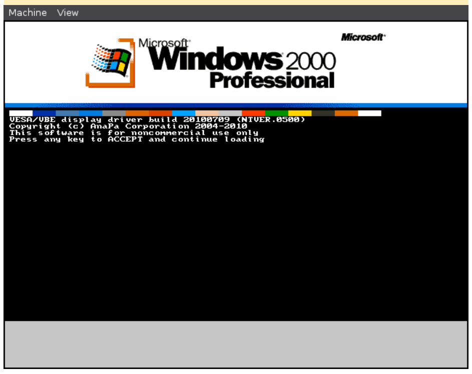
Figure 10 - Thanks to the new VGA driver you can run Windows 2000 in 720p, 1080p and above

ed the vbempj.zip file. You should see a couple folders, so go to VBE30/W2K/PNP/ and select the vbemppnp.ini file. Confirm the selection, then let Windows search for the driver. After a short while, you should see that it has found the drivers and is ready to install them.

Let the setup finish the installation of the drivers. The Window will flicker a couple of times while initializing the new graphics drivers. Although the system doesn't tell you to do so, restart the system at this point in order to properly load the new VGA drivers. Once the reboot has completed, you will see a notification telling you that this driver is for non-commercial use only.

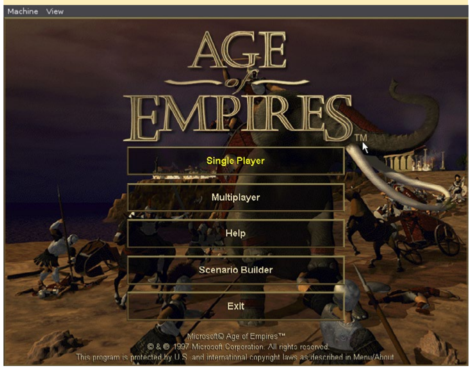
Although it says to press any button, just keep waiting, and the notification will go away on its own. Immediately after login, you should notice the difference in video quality. You can now modify the screen settings even more by simply pressing the right mouse button on an empty space of the desktop and selecting Properties. Switch to the Settings tab and configure your screen resolution.

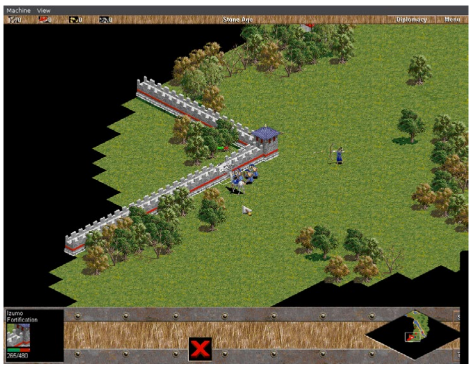
Figure 9 - New VGA drivers notification


```
VESA/VBE display driver build 20100709 (NTVER.0500)
Copyright (c) AnaPa Corporation 2004-2010
This software is for noncommercial use only
Press any key to ACCEPT and continue loading
```

Although nothing prevents you from choosing choose 16, 24, or 32 bits of color for your Windows 2000 machine, I found that everything besides 256 colors will corrupt the Start menu and make it unusable. While this is probably not an issue, since you still can run programs and games directly from the hard drive, it makes it nearly impossible to turn off or restart the machine properly, so I tend to use 256 colors instead. This has one downside: only 4:3 screen ratio supports 256 colors.

Figures 11 and 12 - Age of Empires I running via a Windows 2000 QEMU virtual machine on an ODROID





So what can you do with this Windows 2000 virtual machine? Check out Figures 11 and 12!

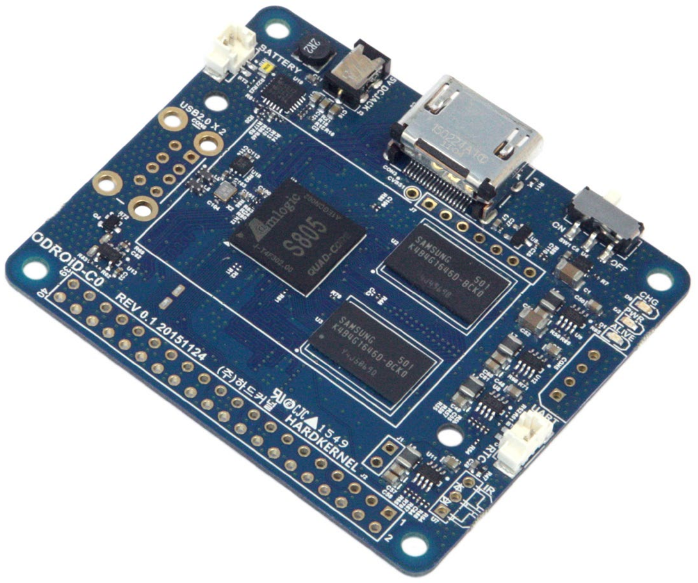
Don't expect too much performance from your VM, because it's rather slow. In fact, in my previous test, I was running it on Windows 98, which was faster than what I experienced here in Windows 2000. However, as mentioned previously, Windows 98 was unable to use audio at all. You can probably speed up things a little bit by using a different CPU or more RAM. Now that you know the basics, you can experiment with different settings.

You can now also install other programs such as Microsoft Office, WinAmp or any Windows applications that you want to try. If you are adventurous, you can use the same methods to install Windows 95, Windows 98, Windows ME, or even Windows XP, and try to find the best option for you.

ODROID-C0

A COMPACT BOARD FOR PORTABLE AND LIGHTWEIGHT APPLICATIONS

by Justin Lee



The ODROID-C0 is a computer intended for those who wish to make more flexible and portable applications. It is a minimized hardware version of the popular ODROID-C1, and may be purchased from the Hardkernel store at http://bit.ly/1WFYKOL for USD$25. It is highly suitable for IoT projects, wearables, and other applications that require a lightweight device such as drones, as shown in Figure 2.

All of the ODROID-C1/C1+ OS images are fully compatible with the ODROID-C0. Some of the modern operating systems that run on the ODROID-C0 are Ubuntu, Android, Arch Linux, Debian, and OpenELEC, with thousands of free open-source software packages available. There are also plenty of custom operating systems available for using the ODROID-C0 as a multimedia center, Kodi, gaming station, headless server, and much more in the ODROID forums at http://bit.ly/1SEUP5p.

Figure 1 - ODROID-C0 with battery

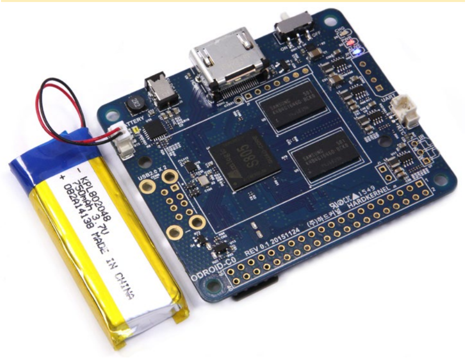
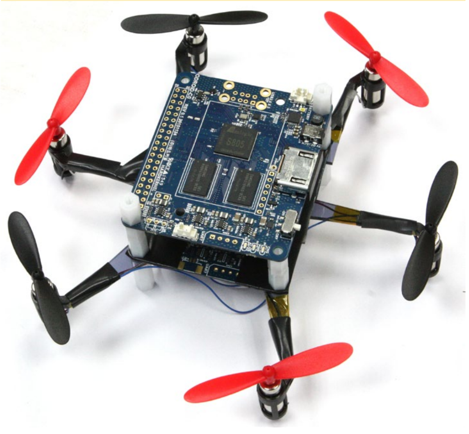


Figure 2 - example drone project using an ODROID-C0



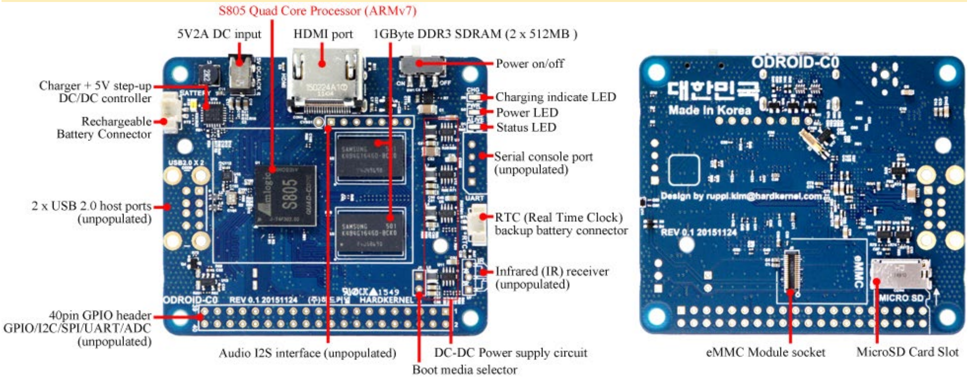
Technical specifications

The key features and improvements over the original ODROID-C1 are shown below:

- Amlogic ARM® Cortex®-A5(ARMv7) 1.5Ghz quad core CPUs
- Mali™-450 MP2 GPU (OpenGL ES 2.0/1.1 enabled for Linux and Android)
- 1Gbyte DDR3 SDRAM
- eMMC4.5 HS200 Flash Storage slot / UHS-1 SDR50 MicroSD Card slot
- 40pin + 7-pin GPIOs (unpopulated)
- USB 2.0 Host x 2 (unpopulated)
- Infrared(IR) Receiver (unpopulated)
- Li+ rechargeable battery charger for wearable and robots application
- Battery voltage level is accessible via ADC in the SoC
- DC/DC step-down converters for higher power efficiency
- DC/DC step-up converter for 5Volt rails (USB host and HDMI) from a Li-Polymer battery
- DIY friendly C0 Connector Pack is available for handy prototyping (described below)
- Fully integrated battery power circuit, so the unit can be made mobile by attaching a 3.7V Li+ battery

For more information, please refer to

Figure 3 - Board detail


S805 Quad Core Processor (ARMv7)
5V2A DC input
HDMI port
1GByte DDR3 SDRAM (2 x 512MB)
Power on/off
Charger + 5V step-up DC/DC controller
Rechargeable Battery Connector
Charging indicate LED
Power LED
Status LED
Serial console port (unpopulated)
2 x USB 2.0 host ports (unpopulated)
RTC (Real Time Clock) backup battery connector
Infrared (IR) receiver (unpopulated)
40pin GPIO header GPIO/I2C/SPI/UART/ADC (unpopulated)
Audio I2S interface (unpopulated)
DC-DC Power supply circuit
Boot media selector
eMMC Module socket
MicroSD Card Slot

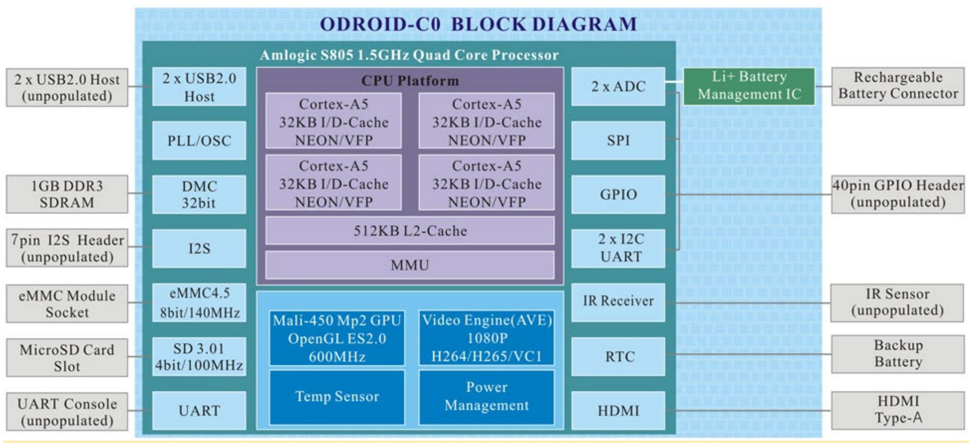

ODROID-C0 BLOCK DIAGRAM
Amlogic S805 1.5GHz Quad Core Processor
CPU Platform
Cortex-A5 32KB I/D-Cache NEON/VFP
Cortex-A5 32KB I/D-Cache NEON/VFP
Cortex-A5 32KB I/D-Cache NEON/VFP
Cortex-A5 32KB I/D-Cache NEON/VFP
512KB L2-Cache
MMU
Mali-450 Mp2 GPU OpenGL ES2.0 600MHz
Video Engine(AVE) 1080P H264/H265/VC1
Temp Sensor
Power Management
2 x USB2.0 Host (unpopulated)
2 x USB2.0 Host
PLL/OSC
1GB DDR3 SDRAM
DMC 32bit
7pin I2S Header (unpopulated)
I2S
eMMC Module Socket
eMMC4.5 8bit/140MHz
MicroSD Card Slot
SD 3.01 4bit/100MHz
UART Console (unpopulated)
UART
2 x ADC
SPI
GPIO
2 x I2C UART
IR Receiver
RTC
HDMI
Li+ Battery Management IC
Rechargeable Battery Connector
40pin GPIO Header (unpopulated)
IR Sensor (unpopulated)
Backup Battery
HDMI Type-A

Figure 4 - Block diagram

the technical details page at http://bit.ly/1RGqrrl.

Documentation

The ODROID-C0/C1/C1+ Wiki at http://bit.ly/1KRKoGV contains information about operating systems, software, and peripherals available for the device. The full schematics may be downloaded from http://bit.ly/1KxguEX, the mechanical drawings are at http://bit.ly/1QlLP1F, and the datasheet is available at http://bit.ly/1dFEHhX.

Specifications

Processor

Amlogic S805 Quad Core Cortex™-A5 processor with Dual Core Mali™-450 GPU

RAM

Samsung K4B4G1646D 1GByte DDR3 32bit RAM (512MByte x 2pcs)

eMMC module socket

8GB/64GB: Toshiba eMMC

16GB/32GB: Sandisk iNAND Extreme

The eMMC storage access time is 2-3 times faster than the SD card. There are 4 storage size options: 8GB, 16GB, 32GB and 64GB. Using an eMMC module will greatly increase speed and responsiveness, similar to the way in which upgrading to a Solid State Drive (SSD) in a typical PC also improves performance over a mechanical hard drive (HDD).

Micro Secure Digital (MicroSD) Card slot

The ODROID-C0 can utilize the newer UHS-1 SD model, which is about 2 times faster than a normal class 10 card.

5V2A DC input

The DC input is designed for 5V power, with an inner diameter of 0.8mm, and an outer diameter of 2.5mm. The ODROID-C0 consumes less than 0.5A in most cases, but it can climb to 2A if many passive USB peripherals are attached directly to the main board.

Rechargeable Battery connection

This is for 3.7V Li-ion or Li-Polymer battery connection. There is a charging indicator LED which turns on when the battery is being charged or unattached. A slide switch is used to turn on/off the board, and the charging circuit still works even you turn the board off.

Battery connector

Molex 53398-0271 1.25mm pitch Header, Surface Mount, Vertical type (Mate with Molex 51021-0200)

USB host ports

There are two USB 2.0 host ports. You can plug a keyboard, mouse, WiFi adapter, storage and many other devices into these ports. You can also charge your smartphone with it! If you need more than 2 ports, you can use a powered external USB hub to reduce the power load on the main device.

HDMI port

Type-A standard HDMI connector for video/audio output.

Status / Power LEDs

The ODROID-C0 has three indicator LEDs that provide visual feedback:

1. The red LED indicates whether the board is attached to 5V power.
2. The blue LED, when it is lit solid, shows that the U-boot program is running. When it is flashing like a heartbeat, the kernel is running.
3. The green LED, when it is lit solid, shows that the battery is charging. When the light turns off, the battery has been fully charged. If there is a fast flashing of the green LED, the battery is not connected or is not functioning properly.

Infrared (IR) receiver

This is a remote control receiver module that can accept standard 37.9Khz wireless data in NEC format.

General Purpose Input and Output (GPIO) ports

These 40-pin GPIO ports can be used as GPIO/I2C/SPI/UART/ADC for electronics and robotics. The 40 GPIO pins on an ODROID-C0 are a great way to interface with physical devices like buttons and LEDs using a lightweight Linux controller. If you're a C/C++ or Python developer, there's a useful library called WiringPi that handles interfacing with the pins. We've already ported the WiringPi v2 library to ODROID-C0. Please note that pins #37, #38 and #40 are not compatible with Raspberry Pi B+ 40pin header, and are dedicated to analog input function. All of the GPIO ports are 3.3Volt, but the ADC inputs are limited to 1.8Volt.

Serial console port

ODROID-XU4 USER MANUAL

A GUIDE FOR ALL EXPERTISE LEVELS

edited by Rob Roy

The official user manual for the ODROID-XU4 was recently released on the Hardkernel website, and is available for direct download at `http://bit.ly/1U9Q8yg`, via the forums at `http://bit.ly/1RykBrT`, and on the Google Play Store at `http://bit.ly/1WrIeSd`.
The ODROID-XU4 is one of the most powerful low-cost Single Board computers available, as well as being an extremely versatile device. Featuring an octa-core Exynos 5422 big.LITTLE processor, advanced Mali GPU, and Gigabit ethernet, it can function as a home theater set-top box, a general purpose computer for web browsing, gaming and socializing, a compact tool for college or office work, a prototyping device for hardware tinkering, a controller for home automation, a workstation for software development, and much more. Some of the modern operating systems that run on the ODROID-XU4 are Ubuntu, Android, Fedora, ARCHLinux, Debian, and OpenELEC, with thousands of free open-source software packages available. The ODROID-XU4 is an ARM device, which is the most widely used architecture for mobile devices and embedded 32-bit computing.

Connecting the serial console port to a PC gives access to the Linux console. You can see the log of the boot, or login to the C0 to change the video or network settings. The serial UART uses a 3.3 volt interface, so we recommend the USB-UART module kit from Hardkernel, available at http://bit.ly/1nhQuIm, which is 100% compatible with the interface.

RTC (Real Time Clock)

There is a backup battery connector if you want to add RTC functions for logging or keeping time when offline by connecting a Lithium coin backup battery (CR2032 or equivalent). All of the RTC circuits are included on the ODROID-C0 by default. It is a Molex 53398-0271

1.25mm pitch Header, Surface Mount, Vertical type (Mate with Molex 51021-0200).

USB VBUS controller

NCP380 Protection IC for USB power supply from OnSemi.

Boot media selector

If this port is opened, the first boot media is always eMMC. If this port is closed, the first boot media is always SD-card.

Power on/off switch

You can turn on/off the system power of the ODROID-C0 using this switch. The charging circuit works regardless of the switch state.

Power supply circuit

Discrete DC-DC converters and LDOs are used for the CPU/DRAM/IO power supply. MP2637 IC is used to implement the charging function and 5V step-up booster.

Connector Pack

The ODROID-C0 connector pack is perfect for various DIY projects with the ODROID-C0. You will need some soldering skills in order to use the pack, and the cost is USD$1.80. Most of the connectors are identical to the ones that are built into the ODROID-C1+.

There are two different USB host connectors in this package. If you want to create a lower profile board, use a single layer connector. If you want to have full connectivity, use a dual layer connector.

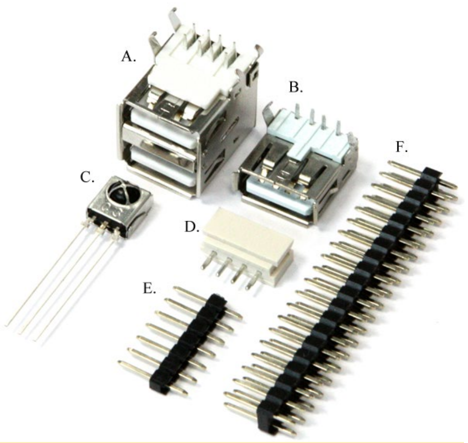


Figure 5 - Connector Pack detail

A. Double Layer USB Type-A 8-Pin DIP Jack Connector

B. Single Layer USB Type-A 4-Pin DIP Jack Connector

C. IR(InfraRed) Receiver Sensor

This is a remote control receiver module that can accept standard 37.9Khz carrier frequency based wireless data.

D. UART Console Connector

A Molex 5268-04a (2.5mm pitch) is mounted on the PCB. Its mate is Molex 50-37-5043 Wire-to-Board Crimp Housing.

E. 7-pin Strip Male Header

Add pins to the audio I2S interface holes in the ODROID-C0.

2.54mm (0.1inch) pitch

F. 20x2-pin Strip Male Header

Add pins to the 40-pin GPIO port.

2.54mm (0.1inch) pitch

OS SPOTLIGHT:
ODROBIAN RETROGAMING ARCADE (ORGA) FOR ODROID-C1+

by @Xeosal

A complete gaminig emulation solution, ORGA will keep you gaming

I recently released my customized entertainment and gaming operating system called Odrobian RetroGaming Arcade (ORGA). It uses the Direct Frame Buffer GPU drivers that I compiled specifically for retro-emulation and multimedia. It is based on the Kodi media center using EmulationStation as the main front end, along with some aspects of RetroPie. ORGA is built on the following formula:

ORGA = (Mali-GPU-Fbdev + SDL2-Fbdev) * (EmulationStation-Fbdev + RetroArch-Fbdev + Kodi-Fbdev) / (Handy X11 GUI)

The features of ORGA include:

Fully themed entertainment system with pre-configured EmulationStation V2

Compiled for Mali-Fbdev assuring maximum FPS performance possible with best CPU efficiency

Fully hardware-accelerated RetroArch configured with SDL2 video drivers as output device for Mali-Fbdev

EmulationStation and RetroArch working through SDL2 on 32bpp color-depth without alpha blending issues

Pre-compiled RetroArch emulation cores pre-installed

Kodi 15.2 Isengard compiled and optimized for AMlogic VPU with OpenGLESv2, configured to use EmulationStation

Increases logical-RAM from 1GB, up from 2GB

All entertainment applications can be launched directly from TTY consoles

You can run the X11 GUI from console anytime employing the Mali X Video drivers

Updates are available directly from the Odrobian Repository at http://bit.ly/1OTGdck

All software is compiled specifically on Odrobian Jessie (Debian 8.2) with optimization flags

Please note that this project is still a work-in-progress effort, and so feedback and enhancements are welcome.

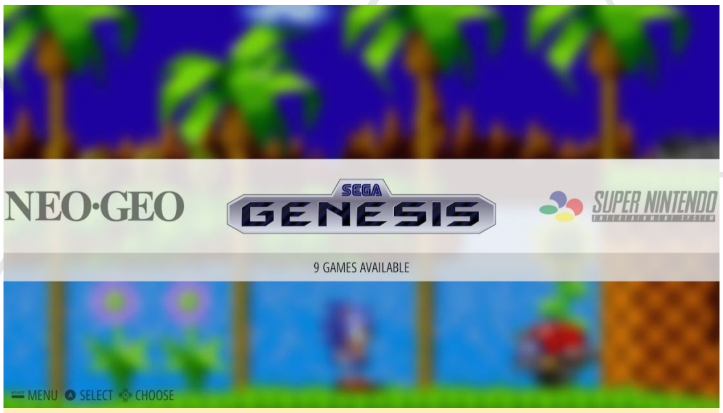
Getting Started

This software is based on the main Odrobian Jessie MATE distribution, which can be converted to an ORGA entertainment system with a single command. To do so, visit the forum thread titled "Odrobian Jessie for ODROID-C1/C1+" at http://bit.ly/1ZVzERa in order to download the latest supported MATE edition. Then, flash the image to your boot media and boot the system with it. Start a terminal session (Ctrl + Alt + T) and run the following commands to install ORGA:

```
$ sudo -s
# apt-get update && apt-get install \
  odrobian-platform-s805
# insta-ORGA
```

Wait until the ODROID-C1+ reboots automatically to a TTY1 console. The X11 desktop will be disabled, and zRam will be activated automatically. Next, upgrade the kernel with the command:

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



Odrobian EmulationStation screenshot

Note that you may already have had the latest regular software and packages updated during the previous "insta-ORGA" command.

Tips

Always log into the system as the "odroid" user and not as "root". When required, use "sudo" for enhanced permission needs. The "odroid" user password is also "odroid".

Always start EmulationStation from TTY1 (Ctrl + Alt + F1), which is the default console on boot. If you want the X11 GUI, it is preferable to start it via TTY3 (Ctrl + Alt + F3). You can utilize the X11 GUI in order to browse the Internet using the Chromium browser, or to configure your ROMs. Emulators and BIOS files can be accessed through the Odrobian MATE desktop environment with the "Caja" File Manager.

You can make Odrobian auto-start EmulationStation directly after the boot process by editing the "/etc/rc.local" file and include the following command just above (before) the "exit 0" statement:

```
runuser -l odroid -c 'emulationstation'
```

Direct FrameBuffer device drivers will display text when opening multimedia files and while performing some video functions. If you do not wish to see the text, you can launch Kodi as a GUI application through the X-server using the following command:

```
$ startx /usr/local/bin/kodi
```

You can also bypass the EmulationStation mechanism if you plan to use the system solely as a media center by making Odrobian auto-start Kodi directly after the boot process has completed. To do so, include the following command just above (before) the "exit 0" statement of the "/etc/rc.local" file:

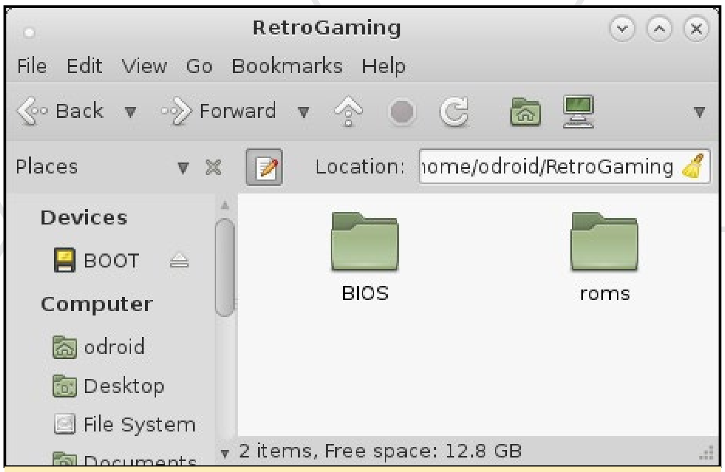
```
runuser -l odroid -c 'kodi'
```

Retro gaming

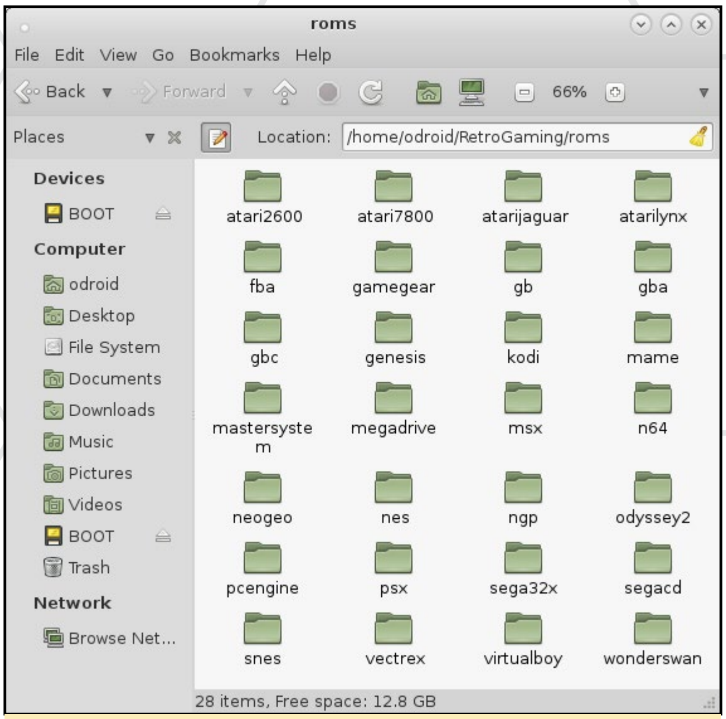
You can switch to TTY3 (Ctrl + Alt + F3) so you can use the X11 GUI to configure the system. To initiate X11, run the following command:

```
$ startx
```

As shown in Figure 2, you should see your emulation system directory at your main home folder in /home/odroid/RetroGaming, which is where you will place your files. For example, if you have a BIOS file for PCSX (PS1) emulation, you can place it directly inside the ~/RetroGaming/BIOS folder, and it will be automatically detected. Emulator game ROMs currently run as RetroArch cores by default. Additional ROMs will be added in the future as they become available.


RetroGaming
File Edit View Go Bookmarks Help
Back Forward
Places
Location: nome/odroid/RetroGaming
Devices
BOOT
Computer
odroid
Desktop
File System
Documents
BIOS
roms
2 items, Free space: 12.8 GB

Home folder for ROMs



ROMs

When you wish to start your EmulationStation front end, you can switch back to the TTY1 console. Press the shortcut key for TTY1 (Ctrl + Alt + F1), then run the following command:

```
$ emulationstation
```

Emulators

I have successfully tested the following games and emulators:

Crash Bandicoot (Play Station I) - Maximum (60 FPS)
Super Mario 64 (Nintendo 64) - Maximum (60 FPS)
Sega MasterSystem - Maximum (60 FPS)
Sega MegaDrive - Maximum (60 FPS)
Sega Genesis - Maximum (60 FPS)
GameBoy - Maximum (60 FPS)
Atari2600 - Maximum (60 FPS)
Atari7800 - Maximum (60 FPS)
Atari Jaguar - Maximum (60 FPS)
Atari Lynx - Maximum (60 FPS)
NES - Maximum (60 FPS)
SNES - Maximum (60 FPS)

After adding the ROMs to the default folder of your desired emulator, they should automatically show up in the Kodi media center. You may be asked to by EmulationStation to configure your joypad or attached keyboard for multimedia usage.

Feature Development

If you wish to contribute new features and functionality to this project, you can find source code and configuration details at the following locations:

EmulationStation Systems:
`/etc/emulationstation/es_systems.cfg`
EmulationStation Themes:
`/etc/emulationstation/themes/`
RetroArch Cores:
`/usr/local/share/retrogaming/cores/`
RetroArch Configurations:
`/etc/retroarch/retroarch.cfg`
Game ROMs:
`/home/odroid/RetroGaming/roms`
Game System Configurations and BIOS:
`/home/odroid/RetroGaming/BIOS`
Other:
`/home/odroid/.config/*`

Here is the list of default RetroArch Cores provided by the "retroarch-default-cores" package:

```
fb_alpha_libretro, fb_alpha_neo_libretro
fceumm_libretro, fmsx_libretro
gambatte_libretro, genesis_plus_gx_libretro
handy_libretro, mednafen_ngp_libretro
mednafen_pce_fast_libretro, mednafen_vb_libretro
mednafen_wswan_libretro, mgba_libretro
mupen64plus_libretro, o2em_libretro
pcsx_rearmed_libretro, picodrive_libretro,
prosystem_libretro
snes9x_next_libretro, stella_libretro
vecx_libretro, virtualjaguar_libretro
```

The ORGA packages tree is organized as follows:

```
odrobian-retrogaming-arcade
libsdl2-fbdev
libsdl2-2.0-0
libsdl2-dev
es2-fbdev
retroarch-fbdev
retrogaming-default-cores
retrogaming-mame-core
kodi-odrobian-fbdev
mali-fbdev
xboxdrv-odrobian
```

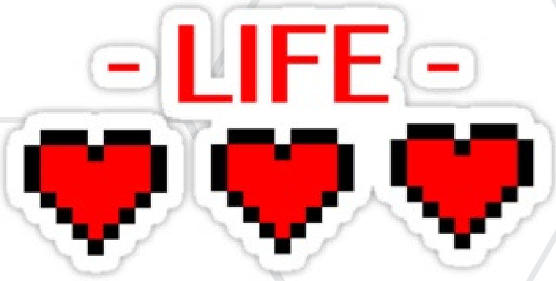
Acknowledgements

Special thanks go the the following forum members and the HardKernel team:

@RetroPie (for ES2 themes and the basic idea)
@memeka (for his SDL2 Mali-Fbdev patch)
@mdrjr (for hosting Odrobian repository)
@owersun (for kodi 15.2 isengard source code)
@meveric (for technical support and references)
@robroy (for community support)

References

If you have questions, comments, or suggestions, please visit the original thread at http://bit.ly/1KjJKif. The original Odrobian Jessie image is also available at http://bit.ly/1ZVzERa.



MUNIN
AN OPEN SOURCE PERFORMANCE ANALYZER

By Adrian Popa

Munin is an open source application capable of recording system and network performance data. It is written in Perl, and uses RRD databases to create graphs which are viewable over a web interface. In addition to creating usage graphs, Munin can generate email alerts when certain thresholds are reached. Its purpose is to give the user a baseline of normal system status to help them understand what went wrong.

Munin can be used in a master or node architecture with multiple nodes reporting performance data to the master. The master node stores the data in RRD database files and generates the graphs for easy viewing. Currently, Munin has over 500 plugins which are available at bit.ly/1P6YW7T. The plugins allow you to monitor things from basics like CPU/memory usage, IO performance, system temperature and network interface throughput, to more the complex things such as MySql slow queries or Transmission's download queue. The simple API, which may be downloaded from bit.ly/1OH7COA, can be used for creating new plugins, and allows any user with basic shell experience to create new graph types.

Installation

To install Munin on an ODROID running Linux, type the following command:

```
$ sudo apt-get install munin
```

If you just want to configure your system as a node, without a web interface, you can execute this instead:

```
$ sudo apt-get install munin-node
```

Web interface configuration

Once you've installed the Munin package, there is some configuration required. Munin's default configuration is stored in /etc/munin. Initially, Munin allows access to the web interface only from localhost, so if you want to access it from your LAN, you'll need to make some changes.

If you're running Ubuntu 14.04, Munin's configuration file for apache is /etc/munin/apache.conf. Unfortunately, its syntax is for Apache 2.2, but Ubuntu 14.04 comes with Apache 2.4. This means you will need to add a "Require ip 192.168.1.0/24" entry under the "Directory" directive to allow access from 192.168.1.0/24, replace it with your desired address or subnet, and also comment out the line with "Order allow,deny", or you can run these one-liner commands in terminal:

```
$ sudo sed -i -r ‘s:^\s+Allow \
  from localhost.*:&\n Require \
  ip 192.168.1.0/24:g’ \
  /etc/munin/apache.conf
$ sudo sed -i -e ‘/Order \
  allow,deny/ s/^#*/#/’ \
  /etc/munin/apache.conf
$ sudo service apache2 reload
```

If you're on Ubuntu 15.04 instead, you can edit /etc/munin/apache24.conf and add your ip/subnet as needed:

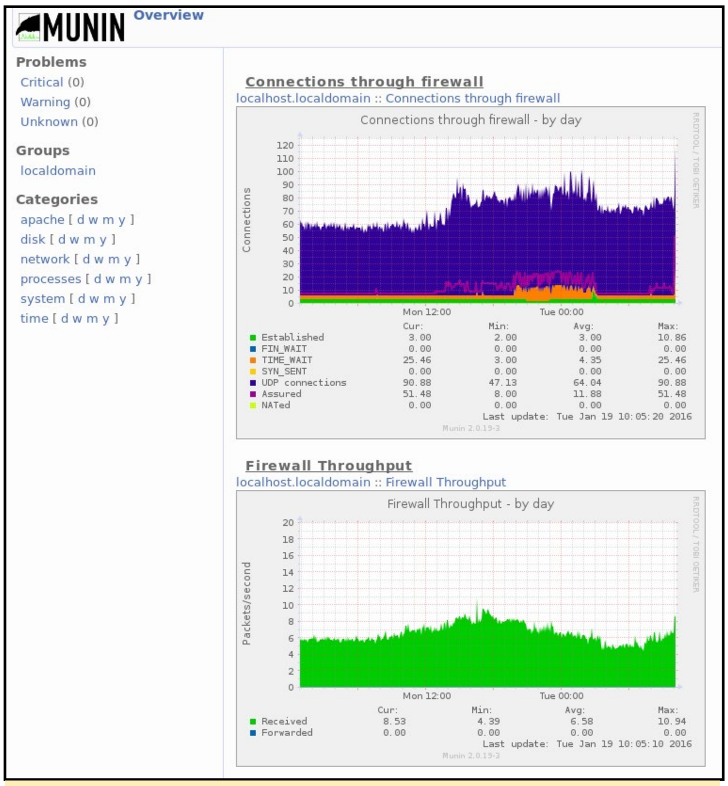
```
$ sudo sed -i ‘s:Require \
  local:&\n Require ip \
  192.168.1.0/24:g’ \
  /etc/munin/apache24.conf
```

To enable the web interface in apache, you can create a symlink between Munin's config and the web server configuration:

```
$ sudo ln -s /etc/munin/apache24.conf \
  /etc/apache2/sites-enabled/010-munin.conf
```

```
$ sudo service apache2 reload
```

If you browse to http://<your-odroid-ip>/munin, you should be presented with a default page where you can view performance data.



```
                        Cur:       Min:       Avg:       Max:
■ Established           3.00       2.00       3.00      10.86
■ FIN_WAIT              0.00       0.00       0.00       0.00
■ TIME_WAIT            25.46       3.00       4.35      25.46
■ SYN_SENT              0.00       0.00       0.00       0.00
■ UDP connections      90.88      47.13      64.04      90.88
■ Assured              51.48       8.00      11.88      51.48
■ NATed                 0.00       0.00       0.00       0.00
                         Last update:  Tue Jan 19 10:05:20 2016
```

```
                 Cur:       Min:       Avg:       Max:
■ Received       8.53       4.39       6.58      10.94
■ Forwarded      0.00       0.00       0.00       0.00
                  Last update:  Tue Jan 19 10:05:10 2016
```

Munin overview - network page

Node configuration

By default, Munin starts monitoring the local system (localhost), and enables a default set of plugins. You should take the time to review the configuration file and tweak settings such as the path to the rrd database or the list of managed nodes. The configuration resides in /etc/munin/munin.conf.

The dbdir, htmldir and logdir entries point to paths in your file system where data will be written frequently, about every 5 minutes. Readings are stored in Round-Robin Databases (RRD), which is a file format with a fixed length where oldest readings are overwritten by new data. This means that write operations will keep writing data to the same sectors on disk, possibly reducing the disk's lifespan. If your system runs from eMMC, then you should be OK, since Hardkernel's eMMCs have internal controllers that provide wear-leveling. However if you're running from an SD card, you risk damaging it over time. An alternate solution might be to keep the data on a USB drive, or on a mounted network share if available. Also, you might want to store the data in RAM, using either /tmp or a dedicated ramdisk, and write it to disk periodically, such as every hour, to reduce wear. Keep this in mind when editing the configuration.

Another important configuration section is the hosts section, which is where you can define which node or nodes you want to monitor. By default, only localhost is listed. You can change the hostname and add additional hosts by running munin-node. Communication between the master instance, which is the one running the web interface, and all other nodes is done over TCP port 4949. You can find detailed instructions on how to setup multiple nodes at http://it.ly/1Noa0YJ. An example section looks like this:

```
# a simple host tree
[procyon]
    address 127.0.0.1
    use_node_name yes
```

Once you are satisfied with the changes, don't forget to reload the munin-node service with the following command:

```
$ sudo service munin-node restart
```

By default, Munin updates the graphs and the web interface files after every 5 minute polling interval. This guarantees that you always see the latest data with the minimum delay. However, unless you monitor the web interface 24/7, it is wasteful to keep generating all the graphs. An improvement you could do is to generate the graphs only on demand when accessing the web interface. This will mean less writing to your storage medium and a slightly increased wait time when browsing. To do this, follow the instructions at http://it.ly/1Noan5w:

```
$ sudo apt-get install libapache2-mod-fcgid
```

Make sure to set the following options in your Munin configuration file:

```
graph_strategy cgi
html_strategy cgi
```

Graph management

At this point, you should be getting graphs from your system or systems, but you haven't yet selected which items to graph. When first installed, Munin scans your system and activates all the plugins that seem suitable. If you install a new service afterwards that has a corresponding plugin, such as mysql-server, Munin will not automatically generate graphs for it. To correct this problem, you should run the following command:

```
$ sudo munin-node-configure --suggest
```

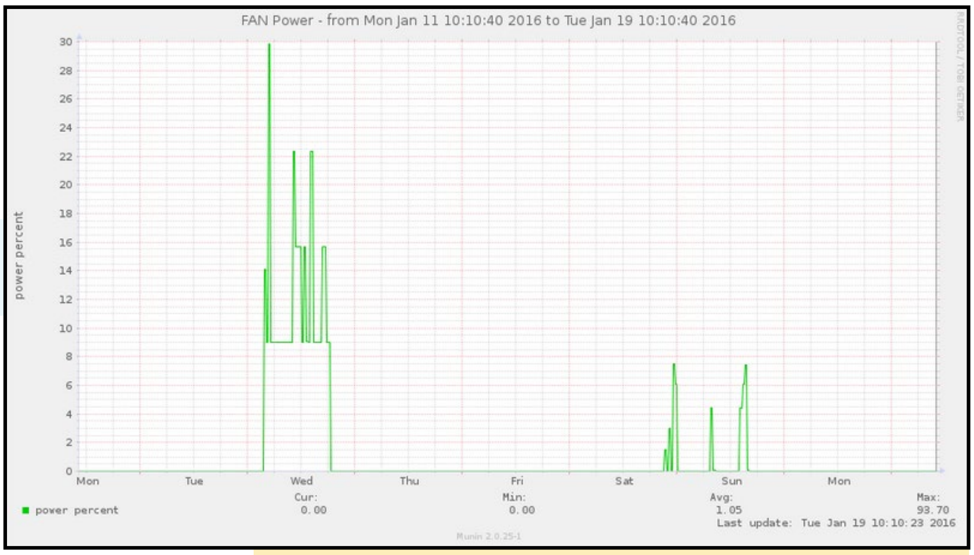
This command will suggest which plugins you can enable based on your current system settings. You can also see why other plugins haven't been enabled, which may happen if they have missing dependencies. To actually enable the plugins you can run the following command:

```
$ sudo munin-node-configure --shell
```

This command will print out some commands that you will need to run in order to enable a specific plugin. You should copy and paste the commands in order to create symlinks in /etc/munin/plugins that point to the actual plugins under /usr/share/munin/plugins. Once the symlinks are in place, the new plugins will be used in the next polling cycle.

Since you're using Munin on an ODROID, you might want to graph ODROID-specific data such as system temperature or fan power percentages. You can get the those plugins, with more coming soon, from my GitHub repository at http://it.ly/1PrTKY8. To enable these extra plugins and fix their dependencies, type the following commands:

```
$ sudo apt-get install bc
$ sudo wget http://bit.ly/1PI0Gkr
  -O /usr/share/munin/\
  plugins/odroid-temp
$ sudo wget http://bit.ly/1OJWYa6 \
  -O /usr/share/munin/\
  plugins/odroid-fan
```



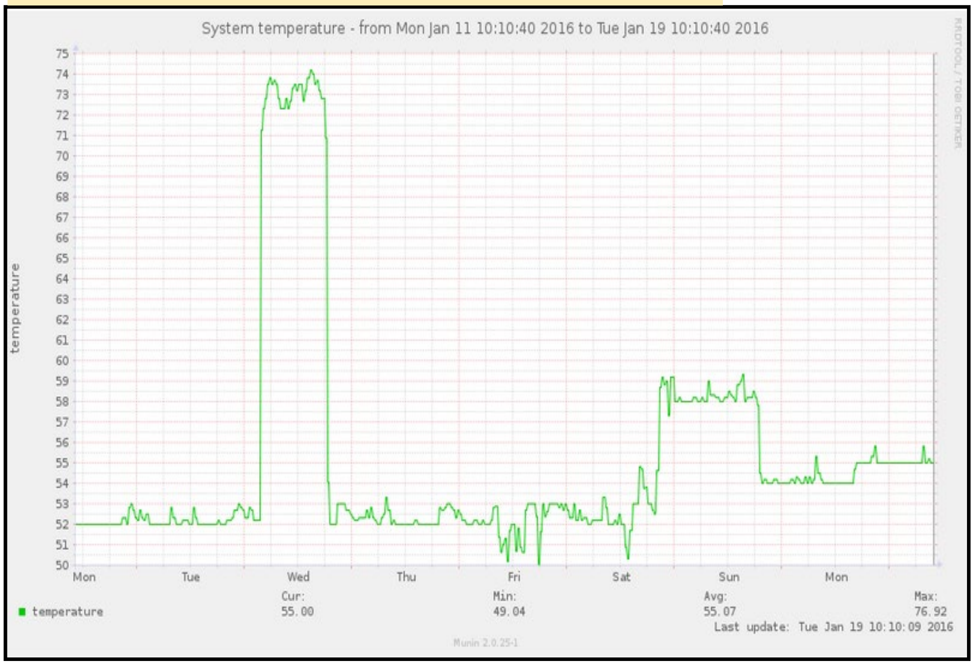
Odroid Fan power percent - weekly

```
$ sudo chmod a+x \
  /usr/share/munin/\
  plugins/odroid*
$ sudo munin-node-configure --shell
```

The last command should suggest what plugins are appropriate for your system, so that you can enable the temperature plugin for C1+ and XU3/4 and both the temperature and fan for XU3/4:

```
$ sudo ln -s \
  '/usr/share/munin/plugins/odroid-temp' \
  '/etc/munin/plugins/odroid-temp'
$ sudo ln -s \
  '/usr/share/munin/plugins/odroid-fan' \
  '/etc/munin/plugins/odroid-fan'
```

System temperature graph - weekly



Please report any issues with these plugins via Github's issue tracker or on the Munin support thread on the ODROID forums at http://bit.ly/1K0rRu9.

Troubleshooting

You might notice that from time to time things may break. Graphs may not update anymore, or they may hold the wrong values, or you may want to understand why a specific plugin reports a particular value. Here are some basic troubleshooting steps:

1. Make sure cron still runs. Polling for data, and graph generation are dependent on cron. You can check that Munin is executed every 5 minutes with this command:

```
$ sudo tail -300 /var/log/syslog |\
  grep munin-cron
```

2. Check when the rrd files were last updated, what kind of data they hold, and what permissions they have. You can do this manually, or by running munin-check:

```
$ sudo munin-check
…
# /var/lib/munin/datafile : Wrong permissions (664 !=
644)
# /var/lib/munin/limits : Wrong permissions (664 !=
644)
# /var/lib/munin/munin-graph.stats : Wrong permis-
sions (664 != 644)
# /var/lib/munin/munin-update.stats : Wrong permis-
sions (664 != 644)
# /var/lib/munin-node/plugin-state : Wrong owner
(root != nobody)
# /var/lib/munin-node/plugin-state : Wrong permis-
sions (755 != 775)
# /etc/munin/plugin-conf.d : Wrong permissions (750
!= 755)
```

Check done. Please note that this script only checks most things, not all things.

Note that munin-check checks the issues, but doesn't fix them, so you'll need to update permissions yourself wherever indicated. If you are investigating actual rrd files, check the file's last change date, and also peek inside to see when it was last updated and what values were written:

```
$ cd /var/lib/munin/procyon
$ sudo ls -l
procyon-uptime-uptime-g.rrd
-rw-rw-r-- 1 munin munin 50664 Jan 16 17:05 procyon-
uptime-uptime-g.rrd
$ sudo rrdtool info \
 procyon-uptime-uptime-g.rrd | \
  egrep 'last_update|value' | head -2
last_update = 1452956716
ds[42].value = 5.6960000000e+01
```

3. Read the logs in /var/log/munin/. Take the time to explore them and see if you can pinpoint the source of the problem.

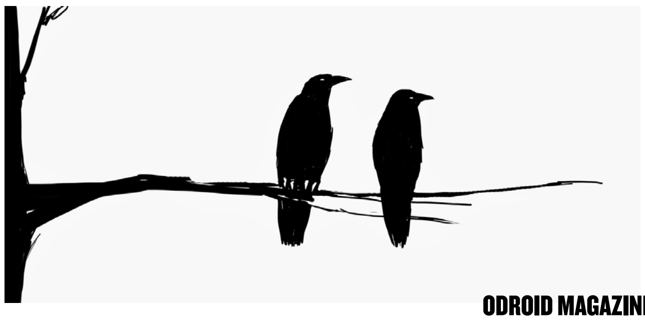
4. If you're troubleshooting a specific plugin, you can directly interact with the munin-node process and ask for feedback (user input is in bold):

```
$ telnet 127.0.0.1 4949
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
# munin node at procyon

nodes
procyon
.

list procyon
cpu df df_inode if_eth0 load memory munin_stats
odroid-fan odroid-temp proc_pri processes uptime us-
ers vmstat

fetch odroid-temp
temperature.value 53
.

quit
Connection closed by foreign host.
```

If the steps above don't fix the issue, you can consult the Munin FAQ for additional assistance at `http://bit.ly/1P9tMwT`.

Conclusion

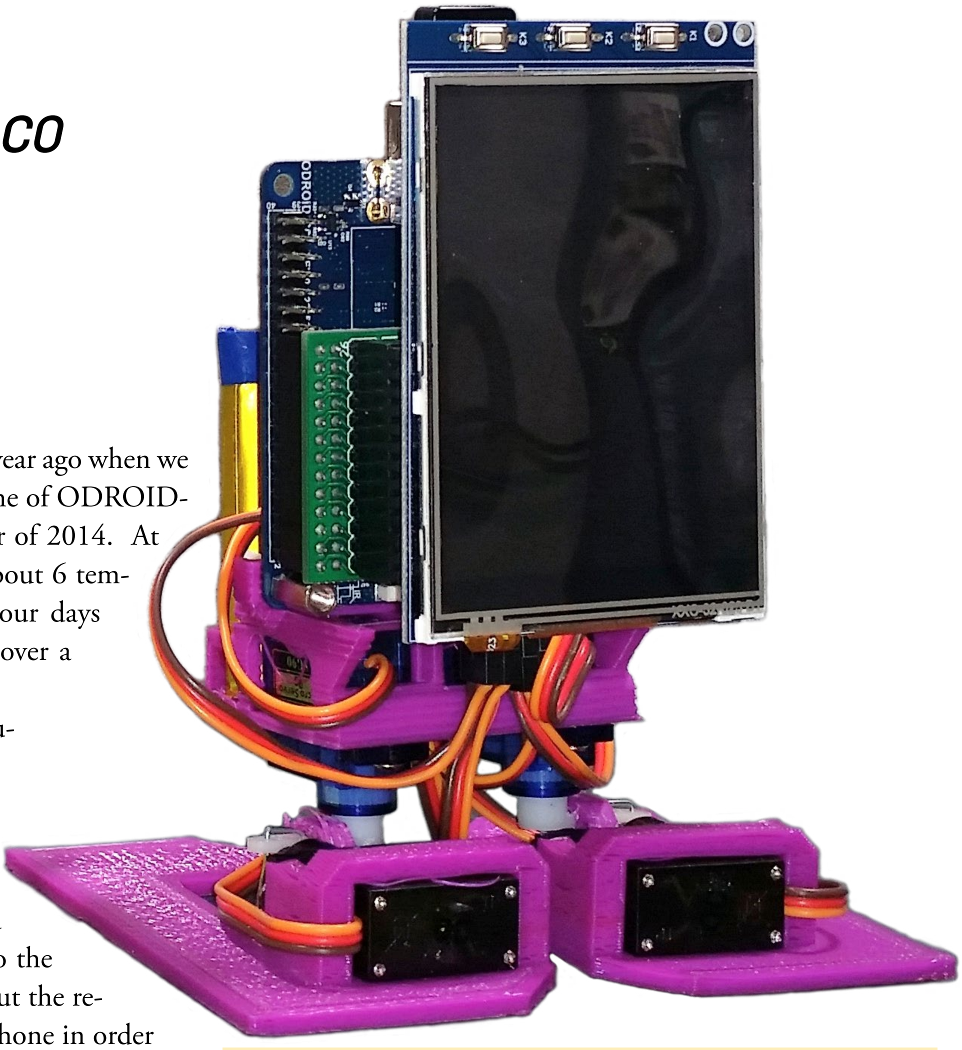
Why would you want to use Munin? Once you have your system running the way you want, you can use the information recorded by Munin to spot what changes over time. For example, maybe your cloudshell has slow performance when playing videos, in which case you could add graphs to keep an eye on your network, IO status or NFS to identify the problem. Also, you could create your own graphs based on custom parameters such as the state of the GPIO ports. Please note that the RRD databases only holds one reading for every 5 minutes, so if your data changes frequently you will only be able to capture snapshots of its state, so it's not suitable for graphing data that changes state several times a second. The API for adding new graphs is easy to follow, and having historical data that you can correlate with other parameters is very helpful in the debugging process.



THE OWEN ROBOT KIT

A SAMPLE PROJECT FOR THE NEW ODROID-C0

by Bo Lechnowsky



The rise of the machines will come in form of the cutest robot ovelords, you have been warned. Get yours now!

The OWEN project started out about a year ago when we finally got a break from the high volume of ODROID-C1 sales that started in mid-December of 2014. At that time, the entire ameriDroid staff, plus about 6 temporary workers, were putting in 12 to 14 hour days shipping C1 orders, 6 days a week, for well over a month!

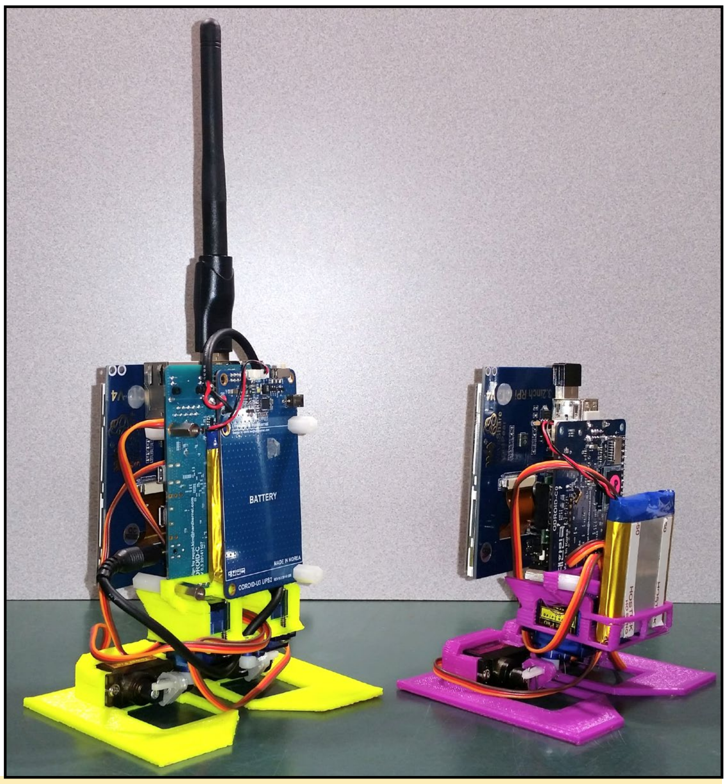
OWEN stands for ODROID Walking Educational uNit, and was coincidentally named after one of our technical interns at the time, also named Owen. The inspiration for the robot was an Arduino-controlled smart phone walking platform called "MobBob". We wanted to see if we could do the same thing with an ODROID-C1, but without the requirement of harnessing an expensive smart phone in order to use the robot.

We designed the pelvis, feet and ankles using Autodesk's great (and free) 123D Design program, and printed the parts on our Solidoodle 3D printer in ABS plastic. It took many attempts to work out all the details, and we continue to make minor tweaks to the designs. Once the C1+ was released in the Summer of 2015, we made a few changes and we were able to use it instead of the C1.

Because we wanted to make use of the C1 3.2" Touchscreen, this ruled out Android as an operating system. Therefore, we utilized Hardkernel's standard Ubuntu distribution. In order to make it easier to install the screen, Owen and I wrote the automated ameriDroid installer for the touchscreen and released it to http://bit.ly/1NtDk09 so that others could use it for their own projects as well.

We needed a servo controller, so we looked around the web for a nice one. We were able to secure distribution of the C1 Chroma Servo module, so we purchased a large quantity of them and started offering them on the ameriDroid.com website. It's a great 8-channel servo controller, but one minor issue is that the 2x13 pin pass-through is offset to the side slightly, causing the screen to be off to the robot's left side by about 6mm. This affects OWEN's balance a little bit, but most of its effects are negated by tweaks in the control software.

The next hurdle was providing battery power, but by the time we were faced with the challenge, Hardkernel had already released the UPS2-C1 kit. One great thing about the UPS2 is that it has an on/off switch, making it easy to shut down the robot. A challenge with the UPS2 is that the servos can draw a lot of momentary current, and the UPS2 is designed to protect the C1 from this type of situation. This caused the robot not to be able to boot up unless it was connected to a wall charger. The solution of removing the power protection IC chip was quickly provided by Justin Lee, and we were up and running. Early tests allowed us to run the robot off the UPS2 for well over an hour!



The OWEN assembled and in 2 configurations: wired and wireless,

We wanted to be able to control the robot, including programming it, using a web browser alone. One challenge in having control software running from inside a web server is the restrictive permissions needed to keep web-launched software from causing damage to the OS. We had to write intermediate software in order to allow the control software to do things like launch and kill processes. I had used Lighttpd web server on other ARM projects before, so it was an easy choice for this project as well.

The robot's control software is mainly written in Rebol3, including the SVG graphics of the face – a built-in feature of Rebol/View. Rebol is a great language because it does not differentiate between data and code. This allows code entered in a web form to be executed without issues. Rebol also has great dialecting capabilities, known as DSL (Domain-Specific Language). So we wrote a DSL to allow the end user to easily program the robot through the web interface.

As a result, the robot can be controlled using any device that has a web browser, including a smartphone, tablet, laptop, desktop computer or other single-board computer. If the proper port is allowed through the network router/firewall, the robot can even be controlled from anywhere in the world.

We brought a few prototypes to ARM TechCon in Silicon Valley where we shared a booth with Hardkernel. Because we knew the wireless network at the convention center would be overtaxed, we brought our own VoCore OpenWRT microrouter to allow the robots, a laptop, smartphones and some ODROID-VU7's to communicate with each other.

We started offering OWEN kits for sale on ameriDroid.com shortly after the TechCon due to quite a few requests from people wanting to know how they could get one. If you'd like to take a video tour of the Hardkernel booth at ARM Techcon 2016, check out the following YouTube links:

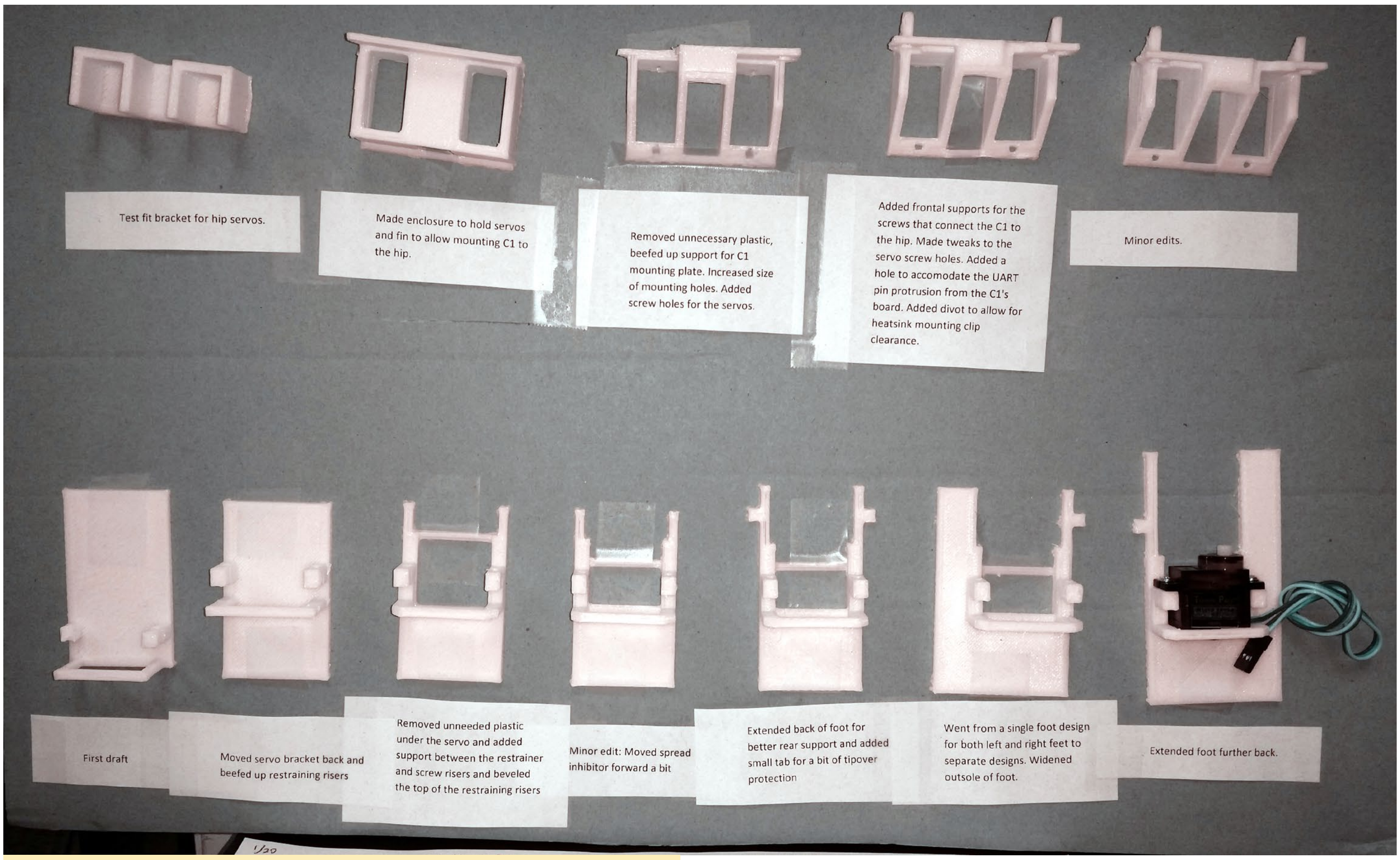
ARMDevices.net http://bit.ly/1OGhQRw

Android Authority http://bit.ly/1PLSxLQ

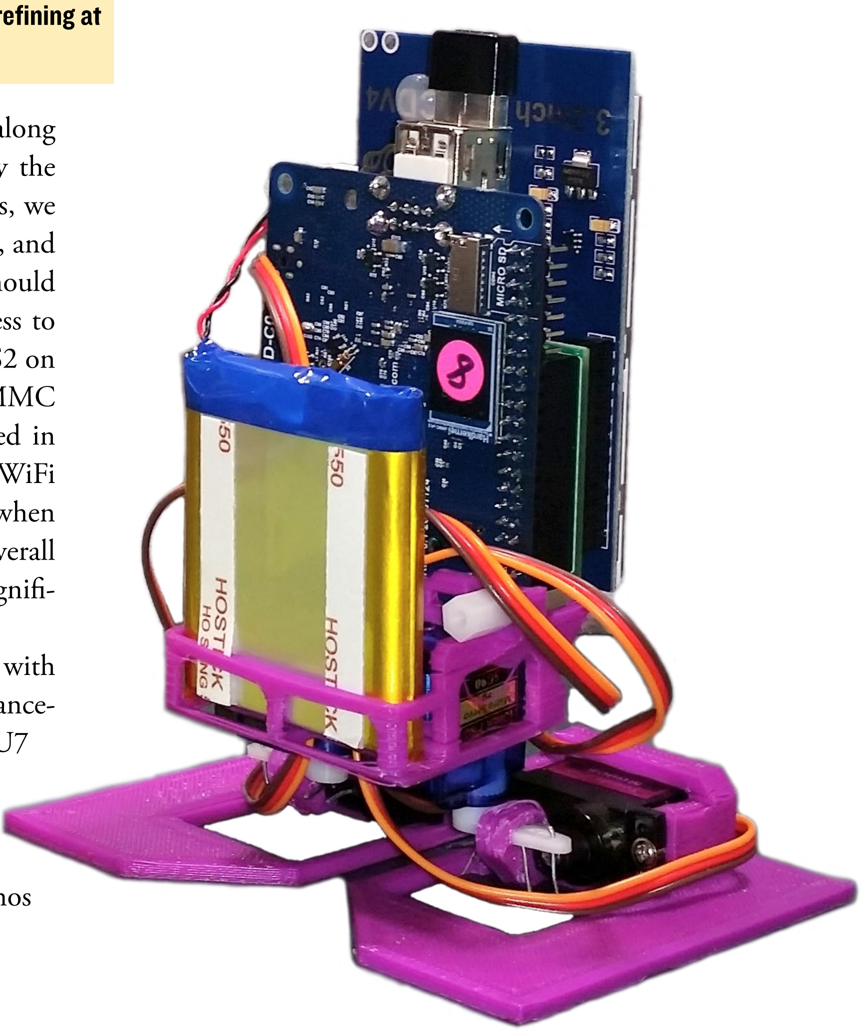
Shortly after the TechCon, Hardkernel informed us that they were coming out with the ODROID-C0, and wanted us to modify OWEN to run off that. It was easy! Some minor changes to the pelvis was all it took to help OWEN shed quite a bit of weight and complexity. Because the C0 has an integrated battery-charging circuit and on/off switch, we bypassed the requirement of needing a UPS2 altogether, with no need for the extra USB and Ethernet ports.

Now, we're working on adding speakers to OWEN. It can already talk with the help of Festival or Flite TTS software for Ubuntu and can play MP3's with mpg321, so dancing to music is no longer a problem, but having good quality sound is a challenge. It should also be possible to use a USB Bluetooth Module 2 to transmit sound from OWEN to a bluetooth speaker, but we wanted to make it completely separate if possible.

In addition, we're developing a smaller, lighter and less expensive servo controller in-house using an ATmega328 Pro


Test fit bracket for hip servos.
Made enclosure to hold servos and fin to allow mounting C1 to the hip.
Removed unnecessary plastic, beefed up support for C1 mounting plate. Increased size of mounting holes. Added screw holes for the servos.
Added frontal supports for the screws that connect the C1 to the hip. Made tweaks to the servo screw holes. Added a hole to accomodate the UART pin protrusion from the C1's board. Added divot to allow for heatsink mounting clip clearance.
Minor edits.
First draft
Moved servo bracket back and beefed up restraining risers
Removed unneeded plastic under the servo and added support between the restrainer and screw risers and beveled the top of the restraining risers
Minor edit: Moved spread inhibitor forward a bit
Extended back of foot for better rear support and added small tab for a bit of tipover protection
Went from a single foot design for both left and right feet to separate designs. Widened outsole of foot.
Extended foot further back.

The secret to good robot design? Prototyping often and refining at each iteration.



Mini board with a custom sketch. This should go along nicely with the smaller form factor of the C0. By the time we're done with the C0-inspired enhancements, we expect OWEN's overall cost to drop by at least 25%, and the front-to-back dimensions of OWEN's body should drop from 60mm to 50mm while adding easy access to the eMMC and microSD slots on the C0. The UPS2 on the C1 version of OWEN blocked access to the eMMC and microSD slots, so the UPS2 has to be removed in order to access the modules. The availability of the WiFi Module 0 will also greatly decrease OWEN's height when compared to the WiFi Modules 3 and 4, and the overall center-of-gravity of the OWEN-C0 model will be significantly lower than OWEN-C1+.

As Hardkernel continues to make new products with great features, we expect to continue making enhancements to OWEN and our other projects, like the VU7 tablet kit. You can keep up with what we're doing by subscribing to our ameriDroid YouTube channel at http://bit.ly/1OGhQ4b. We try to post at least one video every week ranging from tutorials to demos to project videos.

ANDROID DEVELOPMENT

ACCESSING THE BLUETOOTH STACK

by Nanik Tolaram

Android has a plethora of sensors and wireless capabilities for application developers to work with, and one of the standard wireless connection that has been around for a long time, besides WiFi, is Bluetooth. Almost all Android devices support Bluetooth, and there are many Bluetooth products that can be used with Android devices. Android provides a rich and easy-to-use API for working with Bluetooth. In this article, we will take a look at the Bluetooth stack as well as a sample application.

Primarily, there are two types of Bluetooth: Bluetooth Classic and Bluetooth Low Energy. These classifications also differ in the way in which the API works in Android.

Bluetooth Classic

Classic emcompasses "old" Bluetooth v2.1/3.x. The majority of the devices in the market fall into this category. This version of bluetooth is targeted for high-bandwidth data transfer that does not care about power consumption, such as speakers, MIDI, and headphones.

Bluetooth Low Energy

Low Energy is the "new" Bluetooth v4.x, where it is targeted mostly for devices that requires low bandwidth data transfer and long-lasting power such as a beacon.

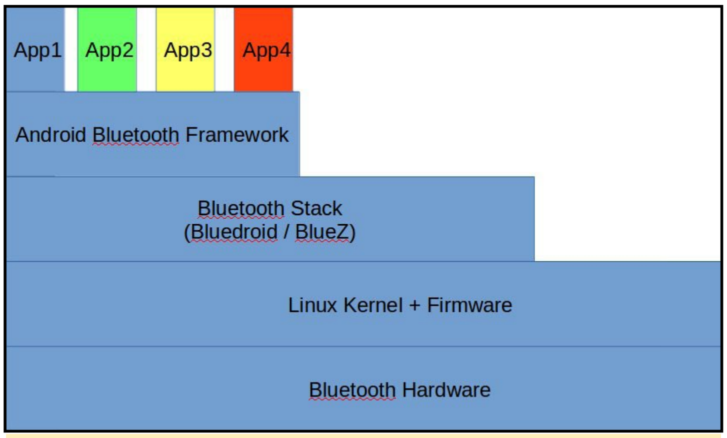


Figure 1 - Bluetooth Stack

Bluetooth Stack

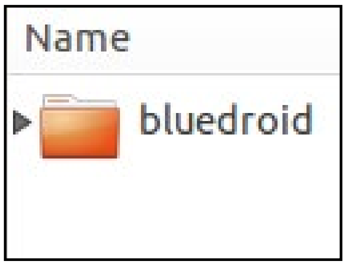
Figure 1 illustrates a high level diagram of the different layers in Android that work together to make your Bluetooth application function.

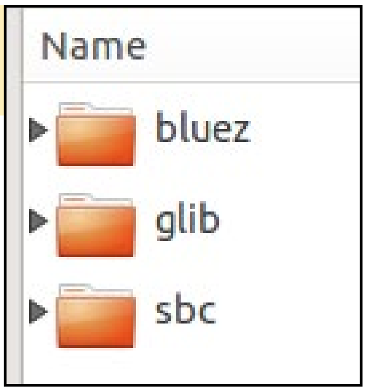
Android Bluetooth Framework – This layer is the "bridge" that closes the gap between your application and whatever lies below it. Most of the time, Bluetooth applications will interact with this layer via the API. For example, the following is the most common code you will use to get a bluetooth adapter to be enabled in your application

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapt-
er.getDefaultAdapter();
```

Bluetooth Stack – This is the "heart" of the whole stack, since without this stack you can assume there will be no Bluetooth in Android. There are 2 different projects in the open source community that are widely used – BlueZ and Bluedroid. BlueZ is the defacto standard in Linux world for any kind of Linux distribution, whereas the primary project used in Android is Bluedroid. BlueZ has very rich support for Android, and is plug-and-play when used inside different version of Android. This software stack resides inside the external/bluetooth directory of Android source code. Figure 3 details the different BlueZ and Bluedroid source structures. The only open source Android variant that uses BlueZ is the Android-x86 project.

Figures 2a and 2b - BlueZ / Bluedroid Source Directory





Linux Kernel + Firmware – This is the final software layer that takes care of the communication between the upper software stack and the hardware. Most Bluetooth devices contain proprietary firmware that are used inside the Linux Kernel.

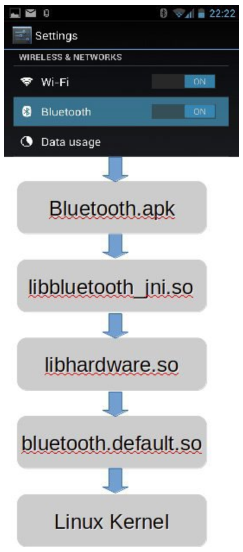
Bluetooth System Files

Both the Android Bluetooth Framework and Bluetooth Stack produce a number of binary files that are loaded up by Android during runtime. Table 1 outlines some of the files that are generated and used by Android.

Binary Filename	Source Code Location
bluetooth.default.so	external/bluetooth/bluedroid
libbt-utils.so	external/bluetooth/bluedroid
libbt-vendor.so	hardware/<vendor>
libbt-hci.so	external/bluedroid/main
libbluetooth_jni.so	packages/app/Bluetooth/jni

Table 1 - Bluetooth System Files

Switching ON



Programming Bluetooth is fun, but have you ever wondered how the internals work? How does the app that "communicates" with the lower level to enable my application send and receive data wirelessly? Figure 3 shows the high level view of what happens internally when you turn on Bluetooth in your device. Turning Bluetooth on manually is done via the Settings application in your Android device.

The only Java application that is used internally is called Bluetooth.apk, which resides in the Android source code under packages/apps/Bluetooth folder.

Figure 3 - Bluetooth Turn ON Process

Bluetooth Programming

As previously mentioned, there are two ways of Bluetooth programming in Android, depending which devices you want to support. Tables 2 and 3 outline some of the classes that are normally used to program Bluetooth.

Class	Usage
BluetoothAdapter	This class represent the local Bluetooth device adapter in your Android device.
BluetoothDevice	This class represent remote Bluetooth device
BluetoothSocket	This class is used when connecting or connected to a Bluetooth device. This class is similar to TCP Socket class
BluetoothServerSocket	This class is similar to TCP ServerSocket class. Used normally as a listening socket
BluetoothHeadset	This class is the API allowing application to control the headset service and also handset profile
BluetoothClass	Generic class describing the characteristics and capabilities of a Bluetooth device

Table 2 - Classic Bluetooth API

Table 3 - Bluetooth Low Energy API

Class	Usage
BluetoothAdapter.LeScanCallback	Callback interface that application normally will implement to receive results from scanning operation
BluetoothGatt	Public API for app to use the GATT functionality enabling communication to LE devices
BluetoothGattCallback	Callback interface for GATT events. For example connection change, services discovered, etc
BluetoothGattServer	This class provides Bluetooth GATT server role functionality, allowing applications to create Bluetooth Smart services and characteristics. This means you can make your Bluetooth adapter inside your Android device as a GATT server to broadcast beacon or some other functionality
BluetoothGattCharacteristic	This class represents a Bluetooth GATT Characteristic
BluetoothGattService	This class represents a Bluetooth GATT services

We will not go in-depth on how to program using Bluetooth in Android, but you can visit Google's Android website, which contains comprehensive documentation along with sample applications for learning more. Visit http://bit.ly/19NYRE3 for classic Bluetooth guidance, and http://bit.ly/1nj9p5Z for BLE tutorials.

Bluetooth Demo

As a demonstration of Android Bluetooth programming, download the Bluetooth Chat sample application available at http://bit.ly/1SuBe7R, which is a basic chat application that provides 2-way chat communication between two Bluetooth device using classic programming. The application is split into 3 main sections:

Scanning/Querying – The application scans nearby Bluetooth devices and displays them using a listview that includes paired devices. If the device has not been paired, the framework will automatically pop up a dialog box allowing the user to enter the PIN number for pairing the device. The way the UI receives result from the framework for a scanned Bluetooth devices is by listening to a BluetoothDevice.ACTION_FOUND intent, then using the method getBondedDevices() to get a list of devices that have been paired. Upon completion of the scanning operation, the app will stop the scanning process.

Connection – Once the user selects the device, the application will spawn a new thread in order to connect to the device. The way to connect to a classic device is by using the createRfcommSocketToServiceRecord (for secured connection) or createInsecureRfcommSocketToServiceRecord (for unsecured connection), and once the connection has been established, the app will spawn another thread to listen for incoming packet from the external device. The spawned thread will be in a blocked state while it waits for an incoming packet from the external device.

Data Transfer – Incoming messages and packets are processed by the application via a message handler in the UI.

MEET AN ODROIDIAN

DAVID LIMA: QUINTESSENTIAL ENTERPRISE STORAGE EXPERT AND SYSTEM ADMINISTRATOR

edited by Rob Roy

Please tell us a little about yourself.

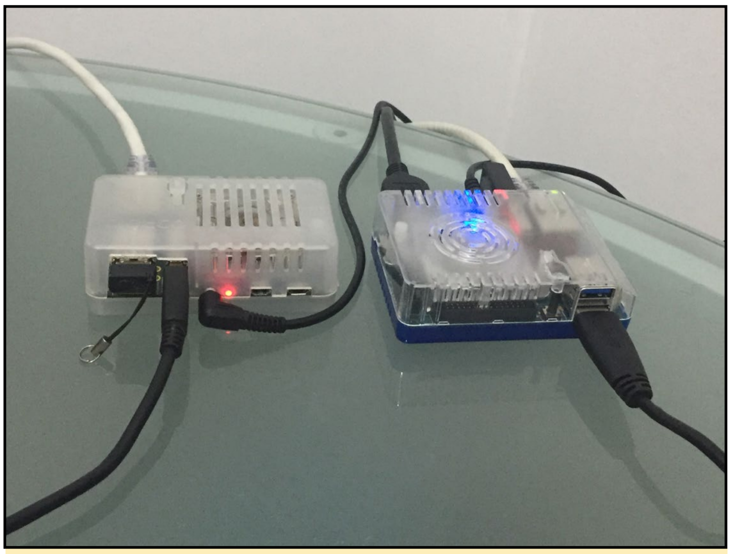
My name is David Lima, and I am 24 years old, living in São Paulo, Brazil. I graduated with a degree in Computing with Software Development, and work as a SAN Disk Storage Specialist dealing with enterprise storage devices and the infrastructure behind them. I am not married but I intend to have two children, then retire to a small peaceful town.

How did you get started with computers?

I remember having a computer at home since I was six, but by the time I noticed that's what I wanted to do for living at about 10 years old, the computer was so outdated that it stopped working all the time. I became curious about how it functioned, and what I could do to fix it. Eventually I was assembling and disassembling that computer like it was made of LEGOs!

What attracted you to the ODROID platform?

I am actually new to ODROIDs. I have always been curious about development boards, and wanted one for myself, but was kind of lazy when it came to learning more about them. About a year ago, a friend of mine introduced me to his ODROID boards and I thought, "Dude, that's awesome. Why didn't I find this before?" I have tried many things on them so far, but still I have a lot to learn and experiment everything this great platform has to offer.



All LVM articles were tested by David's deceptible simple U3 x XU4 setup, showing how far you can go on storage on ODROIDS

David decided to troll his photographer friend by giving all his photos where he never LOOKS AT THE CAMERA!

How do you use your ODROIDs?

My ODROIDs are configured as torrent clients, media servers, Samba servers, local DNS servers, and game stations, with an extra one for experiments where if everything goes wrong, I just flash the image on it again.

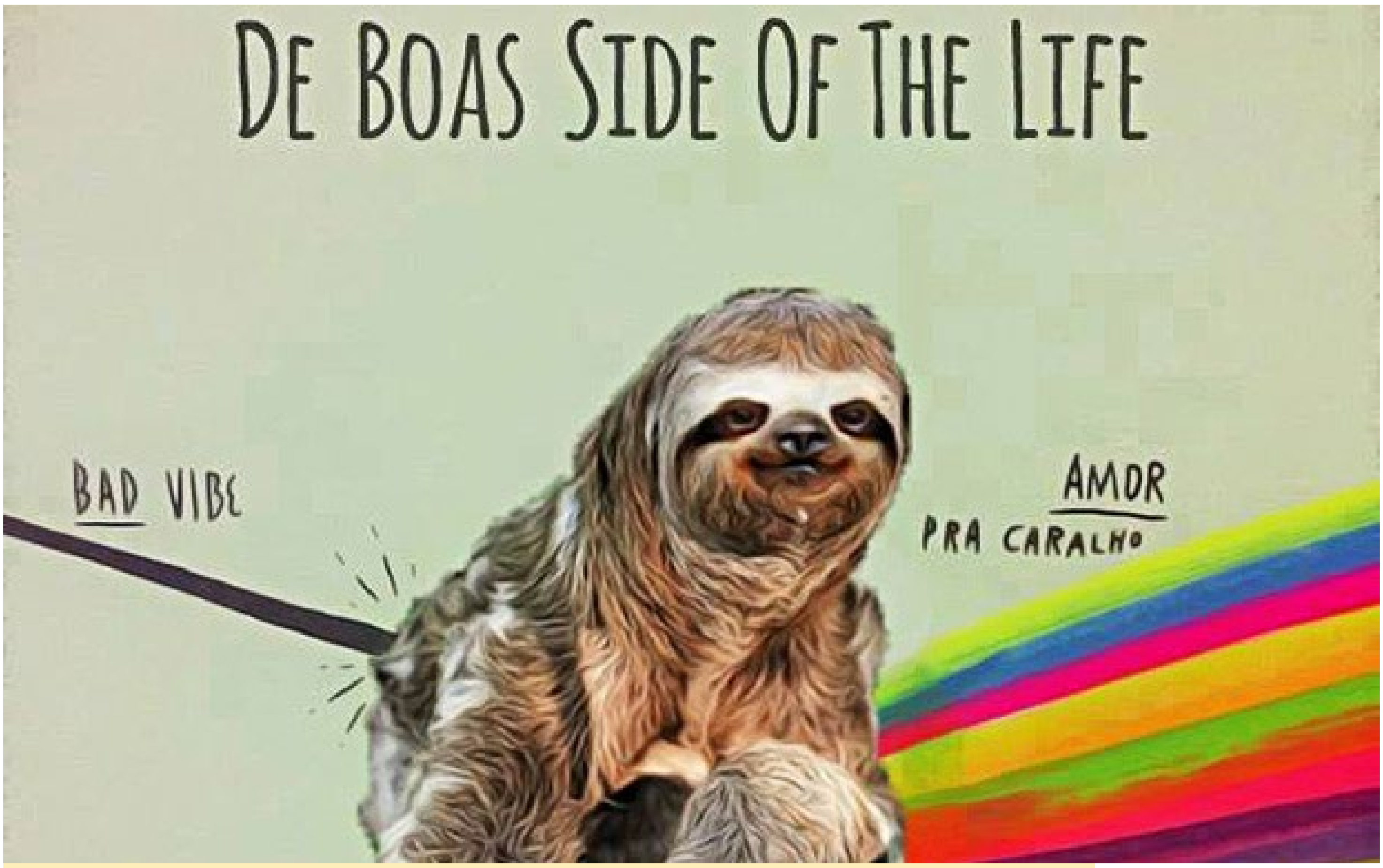
Which ODROID is your favorite?

I still think U3 is the most stable and has the perfect balance between price and processing capacity.

Your Logical Volume Management column is very informative. How did you become an expert in server administration?

I think of myself more like an enthusiast than an expert. I used to work as a Linux system administrator supporting large environments, so that's where most of my experience comes from. I also have had Linux systems at home for quite a while, and am curious when it comes to features for it.

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



Known by all his friends as a zen-life master, he sent this image that represents his philosophy of tranforming bad vibes in positive energy. There is no task that he doesn't tacke with calm and good humor

I am looking forward to having more RAM on the ODROIDs, as well as a 64-bit CPU architecture. Then, I can start creating production-quality virtual machines on it and have a datacenter the size of a credit card. Also, having bluetooth and Wi-Fi embedded on the boards would save a few USB ports.

What hobbies and interests do you have apart from computers?

I am kind of a complete geek, so apart from computers, I enjoy playing a card game called Magic The Gathering. I also love going to the movies and reading superhero comics. My favorite sports are volleyball and swimming, but those are two things I haven`t done in quite a while.

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

The secret to learning anything, which is no secret at all, is to have fun. Do what you like and you won't even notice that you are learning from the experience. Having an objective also helps a lot. If you identify a problem, you will learn a lot on your way to solving it. When you do, unless you want to be Daniel-san from the Karate Kid movie and do wax on/wax off repeatedly, write down how you did it, as you will almost certainly forget after a while. This saves a lot a of time in the future.

Ruric Thar, the Unbowed
Legendary Creature — Ogre Warrior
Vigilance, reach
Ruric Thar, the Unbowed attacks each turn if able.
Whenever a player casts a noncreature spell, Ruric Thar deals 6 damage to that player.
6/6
Boss
"He displays a perverse charisma fueled by avarice. Highly dangerous. Recommend civil sanctions."
—Agmand Sarv, Azorius hussar

Wanna know the dark side of our zen-life master? Play Magic against him and you will find an extremely agressive player, you have been warned!