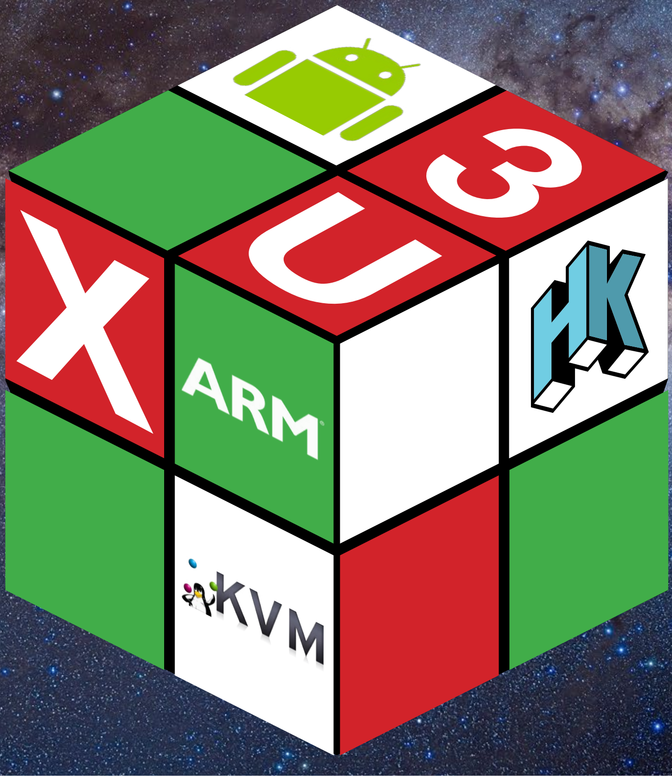
Weather Board Application • Open Media Vault • Installing Node.js

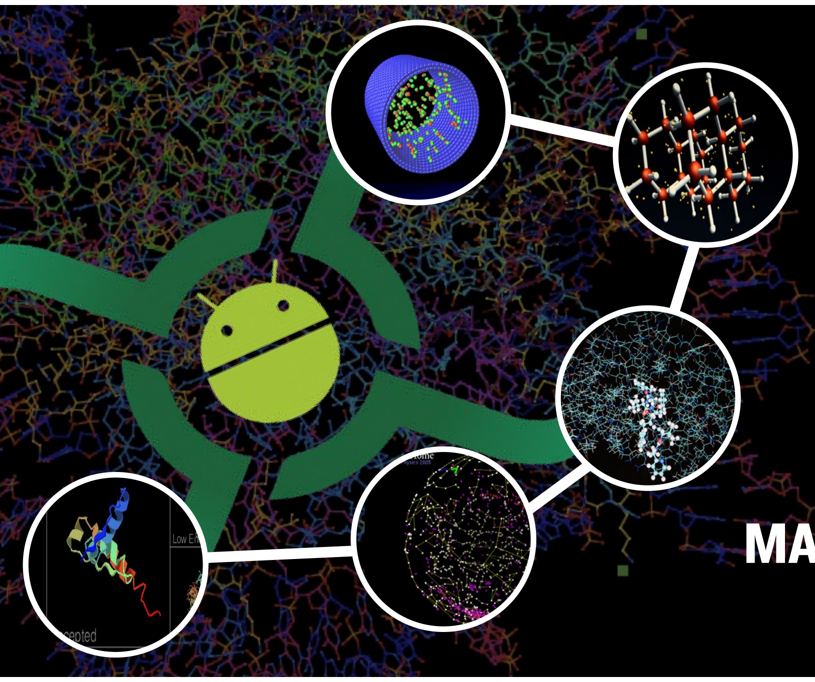
ODROID Magazine

Year One
Issue #11
Nov 2014



VIRTUALIZE NOW!

DISCOVER A UNIVERSE OF POSSIBILITIES WITH KVM TECHNOLOGY

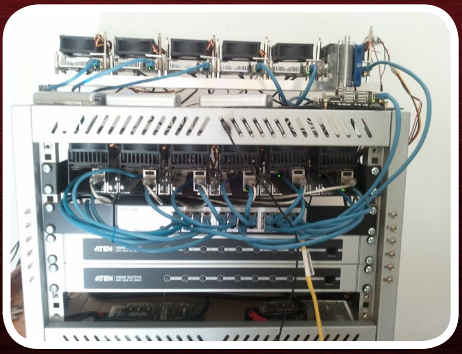


BOINC

THE DISTRIBUTED PROCESSING PLATFORM THAT MAKES THE MOST OF AN ODROID'S LOW POWER CONSUMPTION

- OS SPOTLIGHT: CODE MONKEY
- LINUX GAMING: DOSBOX
- UNMANNED GROUND VEHICLE: GPS NAVIGATION PROGRAMMING

BOINC MONSTER: A WHOPPING 135W CLUSTER WITH 96 CORES



What we stand for.

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

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

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

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

HARDKERNEL

Pollin ELECTRONIC

We are now shipping the ODROID U3 devices to EU countries! Come and visit our online store to shop!

Address: Max-Pollin-Straße 1
85104 Pförring Germany

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

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

Suchergebnis Ihre Suche nach "odroid" ergab 18 Treffer

sortieren nach: Relevanz (gewählt)

1 - 18 von 18 Ergebnissen

Seite: 1 von

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

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 CASE Kunstst
Passgenaues Gehäuse mit Lüftungsschlitzen für den Maße (LxBxH): 87x52x29

4,95 €*
Best.Nr. 810 317

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

ODROID-Linux
Supersch den OD Linux- berei
39

The exciting news this month is that **ODROIDs** are now available for sale in the United States from http://www.ameridroid.com! Based in California, Ameridroid offers affordable shipping for domestic customers, and **US** residents will receive packages much faster. Here is an excerpt from their website:

"Maybe your story is the same. Santa answered a 7-year-old's letter with a shiny soldering iron. Before long, he was taking apart electronics and salvaging the parts to make a crystal radio. He hooked up the antenna lead to his aluminum window screen next to his bed (for better reception) and listened to **AM** radio stations through an earphone (mono, of course) as he drifted off to sleep dreaming of his next electronics project.

We live in a wonderful time. A time where technology is less a deciding factor on what we can invent, as long as we can imagine it. A time when powerful computing starts way less than $100. We love inventing things. A lot of our ideas require computing power. Therefore, we love single-board computers. When it comes to single-board computers, and you want raw computing horsepower for cheap, **ODROID** is the clear leader. We want to hear about your inventions! Send a description and a picture, and we may feature it on our site!"

If you've been waiting for a more affordable version of the **XU3**, Hardkernel has just announced the **$99 XU3-Lite**. It comes with all of the features of the original **XU3**, except that it removes the DisplayPort, the current and voltage sensors, and uses an Exynos 5422 processor clocked at 1.8/1.3 GHz instead of 2.0+/1.4 GHz for the original board. For the original press release, please visit http://bit.ly/1sf7bji.

This month, we feature **BOINC**, the application that lets you participate in a worldwide supercomputing network with an **ODROID**, some **X86** emulators, a step-by-step installation of the **KVM** virtualization application for the **XU3**, as well as the third installment of the popular Unmanned Ground Vehicle series. Venkat brings us guides to Node.js and Open Media Vault, and the **OS** spotlight this month is about Code Monkey, an all-in-one developer's image. It comes packed with compilers, development environments, and code management tools so you can get started with programming on the **ODROID** right away.

ODROID Magazine, published monthly at http://magazine.odroid.com, is your source for all things ODROIDian.
Hard Kernel, Ltd. • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815
Makers of the ODROID family of quad-core development boards and the world's first ARM big.LITTLE architecture based single board computer.
Join the ODROID community with members from over 135 countries, at http://forum.odroid.com, and explore the new technologies offered by Hardkernel at http://www.hardkernel.com.

HARDKERNEL

ODROID Magazine

Rob Roy, Chief Editor

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

Bo Lechnowsky, Editor

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

Bruno Doiche, Art Editor

Secured his computing necromantic skills after bringing a fiber optics switch back to life, getting his Macintosh back from death, getting a PS3 back from death, getting his fiancee T400 back from death (that was a old style dd data transplant), and managing how to handle the cold innards of his steady job data center.

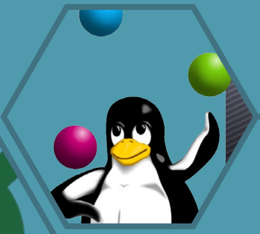
Nicole Scott, Art Editor

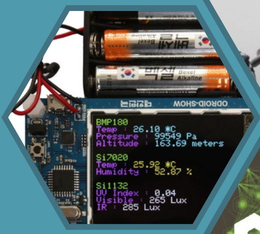
I'm a Digital Strategist and Transmedia Producer specializing in online optimization and inbound marketing strategies, social media directing, and media production for print, web, video, and film. Managing multiple accounts with agencies and filmmakers, from Analytics and Adwords to video editing and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

Manuel Adamuz, Spanish Editor

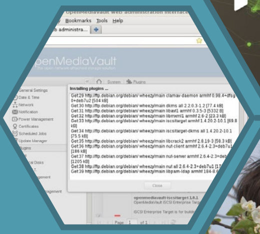
I am 31 years old and live in Seville, Spain, and was born in Granada. I am married to a wonderful woman and have a child. A few years ago I worked as a computer technician and programmer, but my current job is related to quality management and information technology: ISO 9001, ISO 27001, and ISO 20000.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

INDEX





```
BMP180
Temp     :  26.10 *C
Pressure :  99549 Pa
Altitude :  163.69 meters
Si7020
Temp     :  25.92 *C
Humidity :  52.87 %
Si1132
UV Index :  0.04
Visible  :  265 Lux
IR       :  285 Lux
```



KVM VIRTUALIZATION ON THE ODROID-XU3

A BRAVE NEW WORLD OF COMPUTING

By Mauro Ribeiro

The open-source Kernel-based Virtual Machine (KVM) projects allows an ODROID, and many other computers, to host a second operating system while using the same basic kernel. With KVM, one can run multiple virtual machines running unmodified Linux or Android images. Each virtual machine has a private virtualized network card, hard disk, and graphics adapter.

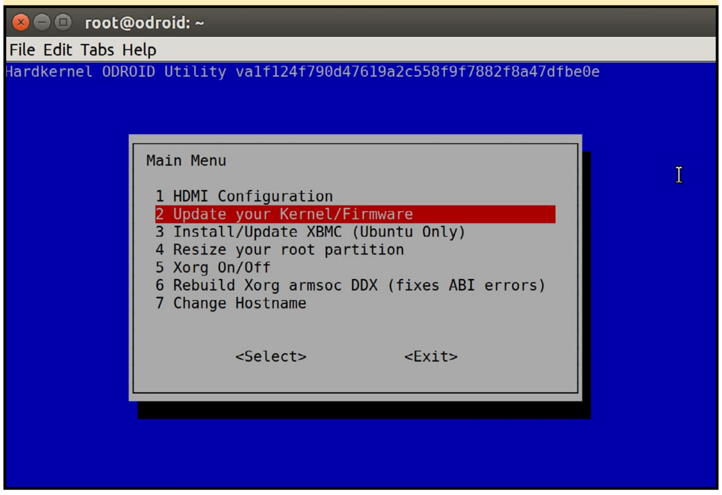
This article demonstrates how to test KVM/Virtualization support on XU3. Virtualization on ARM is still on its early steps and this should not be taken into production environments. Setting up KVM requires at least some Linux knowledge since we'll cover kernel rebuild, bootloaders update and qemu building. All the instructions below should be performed on the board itself. Instructions were written based on our Ubuntu 14.04 image.

Bootloader Update

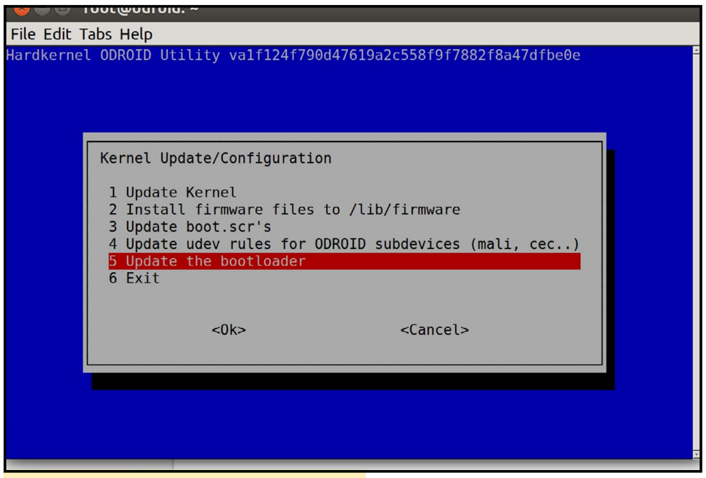
Open a terminal window in Ubuntu 14.04 and type the following:

```
$ sudo odroid-utility.sh
```

The ODROID utility comes with all offical Hardkernel images



```
Hardkernel ODROID Utility va1f124f790d47619a2c558f9f7882f8a47dfbe0e

Main Menu

 1 HDMI Configuration
 2 Update your Kernel/Firmware
 3 Install/Update XBMC (Ubuntu Only)
 4 Resize your root partition
 5 Xorg On/Off
 6 Rebuild Xorg armsoc DDX (fixes ABI errors)
 7 Change Hostname

        <Select>          <Exit>
```



```
Hardkernel ODROID Utility va1f124f790d47619a2c558f9f7882f8a47dfbe0e

Kernel Update/Configuration

 1 Update Kernel
 2 Install firmware files to /lib/firmware
 3 Update boot.scr's
 4 Update udev rules for ODROID subdevices (mali, cec..)
 5 Update the bootloader
 6 Exit

        <Ok>              <Cancel>
```

Updating the bootloader

Navigate to option 2 as shown in the screenshot, then select option 5 in order to update the bootloader.

Adding KVM support

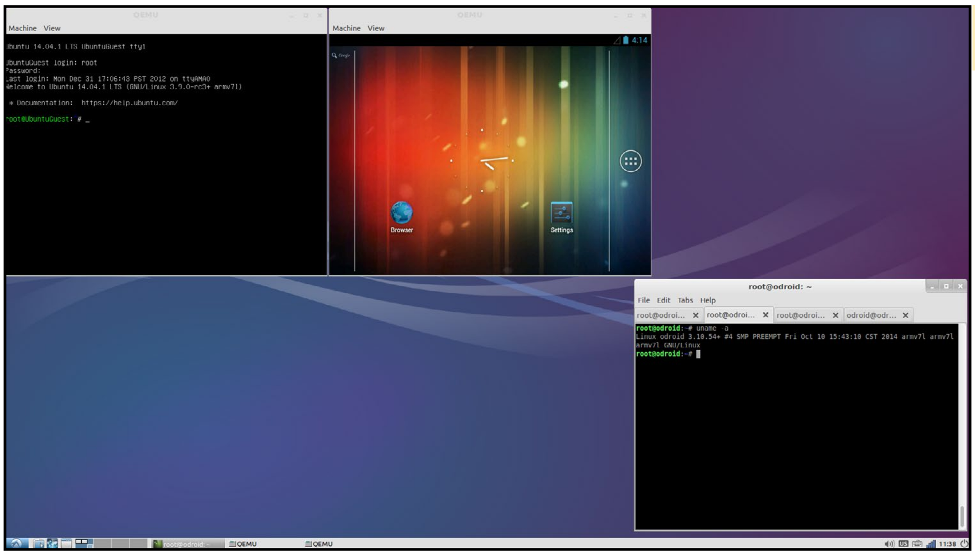
Backup your current kernel and dtb:

```
$ mkdir ~/backup_pre_kvm
$ cp /media/boot/* ~/backup_pre_kvm
```

Build the Kernel with KVM support:

```
$ git clone https://github.com/hardkernel/linux.git
-b odroidxu3-3.10.y
$ cd kernel
$ make odroidxu3_kvm_defconfig
$ make -j9
$ cp arch/arm/boot/zImage /media/boot/zImage
$ cp arch/arm/boot/dts/exynos5422-odroidxu3.dtb /me-
dia/boot/exynos5422-odroidxu3.dtb
$ make modules_install
```

At this point, you should have a board with a kernel capable of using KVM.



```
Ubuntu 14.04.1 LTS UbuntuGuest tty1

UbuntuGuest login: root
Password:
Last login: Mon Dec 31 17:06:43 PST 2012 on ttyAMA0
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.9.0-rc3+ armv7l)

 * Documentation:  https://help.ubuntu.com/

root@UbuntuGuest: # _
```

root@odroid: ~

```
root@odroid:~# uname -a
Linux odroid 3.10.54+ #4 SMP PREEMPT Fri Oct 10 15:43:10 CST 2014 armv7l armv7l
armv7l GNU/Linux
root@odroid:~#
```

KVM in action with Ubuntu 14.04

To learn more about KVM, or to contribute to the project, visit the KVM home page: `http://bitly/18isyvK`

Building the guest kernel

For the guest kernel, we used kernel 3.9 from Virtual Open Systems, which is specifically intended for ARM Foundation models. Any kernel version that supports KVM guest options and the foundation model hardware should work as well.

Please note that if you don't want to take the time to build the kernel, I've created some pre-built binaries, which can be downloaded by typing the following into a Terminal window:

```
$ cd ~/kvm
$ wget http://odroid.in/guides/kvm/kernel.tar.xz
$ tar -Jxf kernel.tar.xz
```

To build the kernel from source instead, type the following:

```
$ git clean -f -d -x
$ git remote add kvm_kernel https://github.com/vir-
tualopensystems/linux-kvm-arm.git
$ git fetch kvm_kernel
$ git checkout kvm_kernel/kvm-vexpress-3.9
$ wget -O .config http://odroid.in/guides/kvm/guest_
config
$ make -j9 zImage dtbs
$ mkdir ~/kvm
$ cp arch/arm/boot/zImage ~/kvm
$ cp arch/arm/boot/dts/rtsm_ve-cortex_a15x1.dtb ~/kvm
```

Building Qemu

Again, if you don't want to build the package from source, a pre-built version of Qemu is available by typing the following into a Terminal window:

```
$ wget -O ~/kvm/qemu.tar.xz http://odroid.in/guides/
kvm/qemu.tar.xz
$ cd /usr/local && tar -Jxf ~/kvm/qemu.tar.xz && cd -
```

Otherwise, type this to compile Qemu:

```
$ git clone git://github.com/virtualopensystems/qemu.
git -b kvm-arm-virtio-fb-hack
$ ./configure --target-list=arm-softmmu --audio-drv-
list=”” --audio-card-list=”” \
--enable-fdt --enable-kvm --enable-sdl -prefix=~/kvm
```

Testing

For testing, I created two pre-built images: one for Ubuntu 14.04 and another for Android 4.1.2. To download the Android version:

```
$ cd ~/kvm
$ wget http://odroid.in/guides/kvm/android.jb.img.gz
$ gzip -d android.jb.img.xz
$ wget http://odroid.in/guides/kvm/android.sh
$ chmod +x android.sh
$ ./android.sh
```

To download the Linux Lubuntu 14.04 version:

```
$ cd ~/kvm
$ wget http://odroid.in/guides/kvm/lubuntu.xu3.img.gz
$ gzip -d lubuntu.xu3.img.gz
$ wget http://odroid.in/guides/kvm/lubuntu.sh
$ chmod +x lubuntu.sh
$ ./lubuntu.sh
```

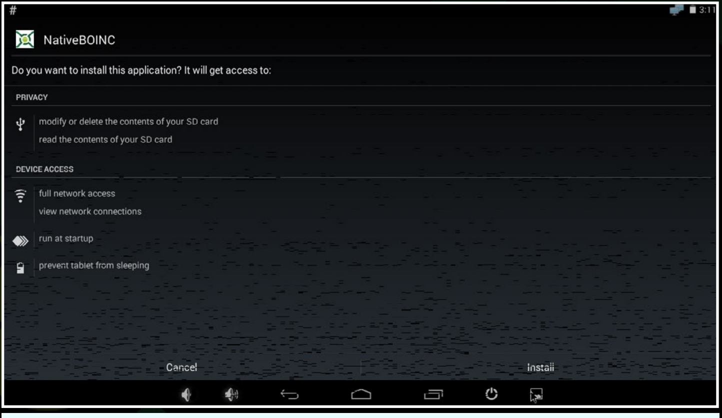
Happy hacking, and we look forward to seeing what everyone is able to do with the KVM images! Thanks to Suriyan for his XEN skills, Fanta for his version of u-boot with Hypervisor support, and Virtual Open Systems for creating the kernel, qemu and test Android images.

INSTALLING NATIVE BOINC

A PICTORIAL GUIDE

By Uli Abromeit

BOINC is a platform for distributed grid computing, widely used by scientists, universities and individuals to help explore the frontiers of human knowledge. It is a volunteer project, with a goal of improving the world by discovering new theorems, innovative medicine and other emerging discoveries. Volunteers can donate the normally unused power of their computers for chosen projects by performing scientific computations. You can get started by downloading the NativeBOINC application from the Google Play Store, or from the NativeBOINC home page at http://bit.ly/1o9rRxg.


NativeBOINC

Do you want to install this application? It will get access to:

PRIVACY

modify or delete the contents of your SD card

read the contents of your SD card

DEVICE ACCESS

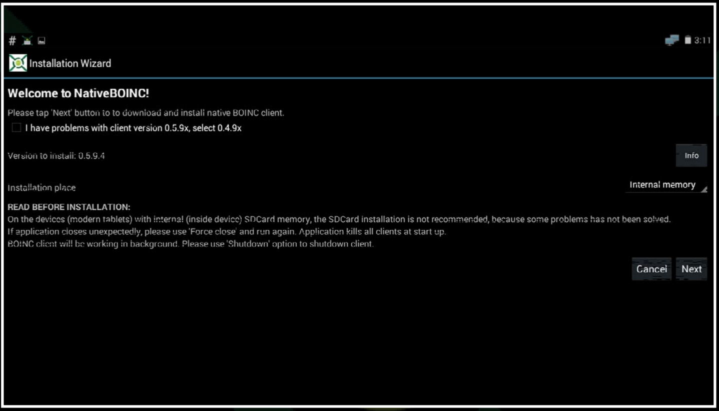
full network access

view network connections

run at startup

prevent tablet from sleeping

1 - After downloading and clicking the nativeboinc.apk file, click the Install button


Installation Wizard

Welcome to NativeBOINC!

Please tap 'Next' button to to download and install native BOINC client.

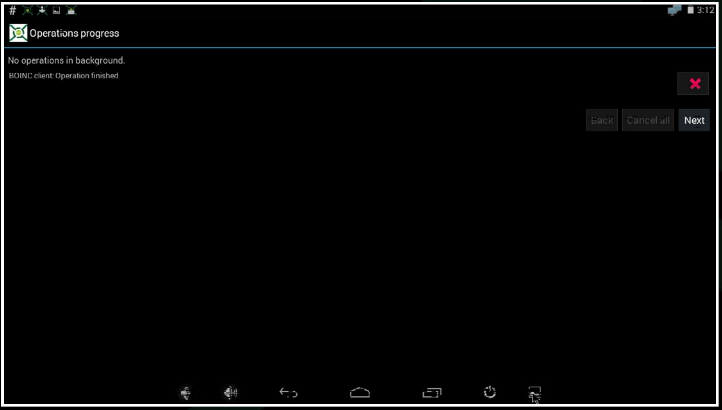
☐ I have problems with client version 0.5.9x, select 0.4.9x

Version to install: 0.5.9.4

Installation place: Internal memory

READ BEFORE INSTALLATION:
On the devices (modern tablets) with internal (inside device) SDCard memory, the SDCard installation is not recommended, because some problems has not been solved.
If application closes unexpectedly, please use 'Force close' and run again. Application kills all clients at start up.
BOINC client will be working in background. Please use 'Shutdown' option to shutdown client.

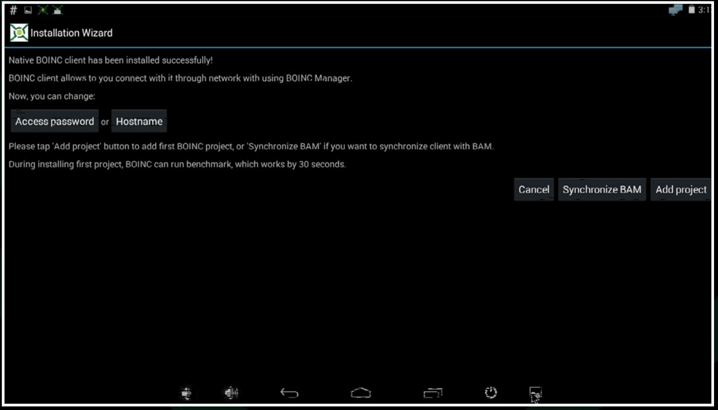
2 - Click "Next" to start the installation


Operations progress

No operations in background.

BOINC client: Operation finished

3 - Click "Next" to complete the installation


Installation Wizard

Native BOINC client has been installed successfully!

BOINC client allows to you connect with it through network with using BOINC Manager.

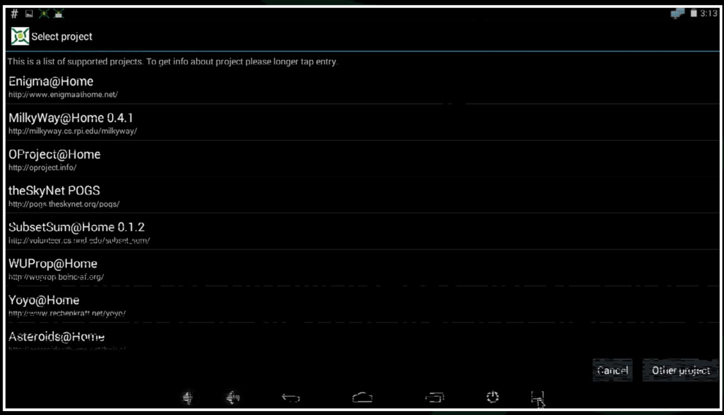
Now, you can change:

Access password or Hostname

Please tap 'Add project' button to add first BOINC project, or 'Synchronize BAM' if you want to synchronize client with BAM.

During installing first project, BOINC can run benchmark, which works by 30 seconds.

4 - Set the access password for the remote computer, as well as the hostname if you run more than one host

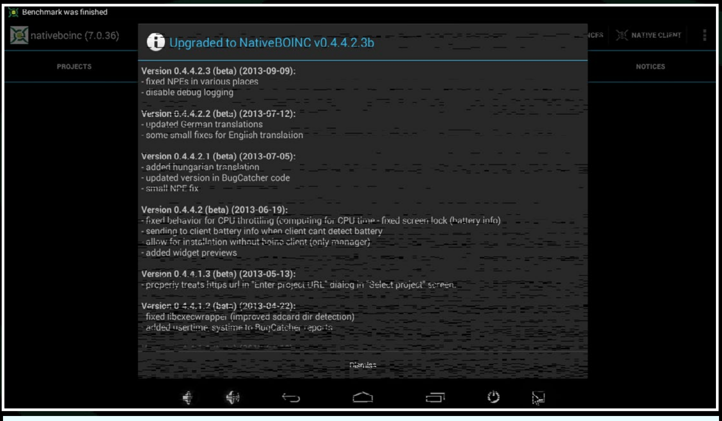


Select project

This is a list of supported projects. To get info about project please longer tap entry.

- Enigma@Home
  http://www.enigmaathome.net/
- MilkyWay@Home 0.4.1
  http://milkyway.cs.rpi.edu/milkyway/
- OProject@Home
  http://oproject.info/
- theSkyNet POGS
  http://pogs.theskynet.org/pogs/
- SubsetSum@Home 0.1.2
  http://volunteer.cs.und.edu/subset_sum/
- WUProp@Home
  http://wuprop.boinc-af.org/
- Yoyo@Home
  http://www.rechenkraft.net/yoyo/
- Asteroids@Home

5 - Select one of the projects from the list of supported projects, set the email and password, then press OK



Benchmark was finished

nativeboinc (7.0.36)

Upgraded to NativeBOINC v0.4.4.2.3b

Version 0.4.4.2.3 (beta) (2013-09-09):
- fixed NPEs in various places
- disable debug logging

Version 0.4.4.2.2 (beta) (2013-07-12):
- updated German translations
- some small fixes for English translation

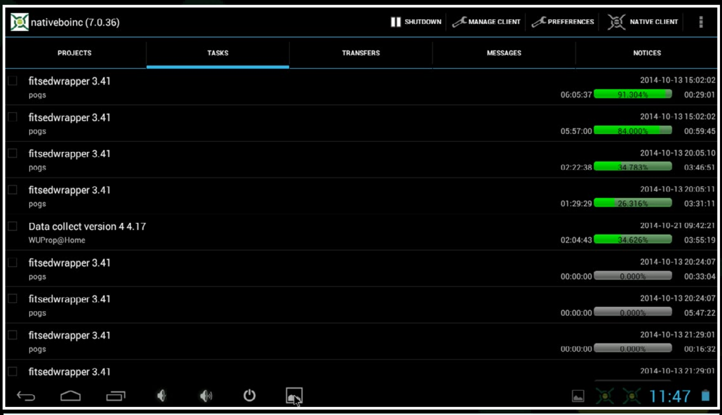
Version 0.4.4.2.1 (beta) (2013-07-05):
- added hungarian translation
- updated version in BugCatcher code
- small NPE fix

Version 0.4.4.2 (beta) (2013-06-19):
- fixed behavior for CPU throttling (computing for CPU time - fixed screen lock (battery info)
- sending to client battery info when client cant detect battery
- allow for installation without boinc client (only manager)
- added widget previews

Version 0.4.4.1.3 (beta) (2013-05-13):
- properly treats https url in "Enter project URL" dialog in "Select project" screen.

Version 0.4.4.1.2 (beta) (2013-04-22):
- fixed libcxecwrapper (improved sdcard dir detection)
- added usertime, systime to BugCatcher reports

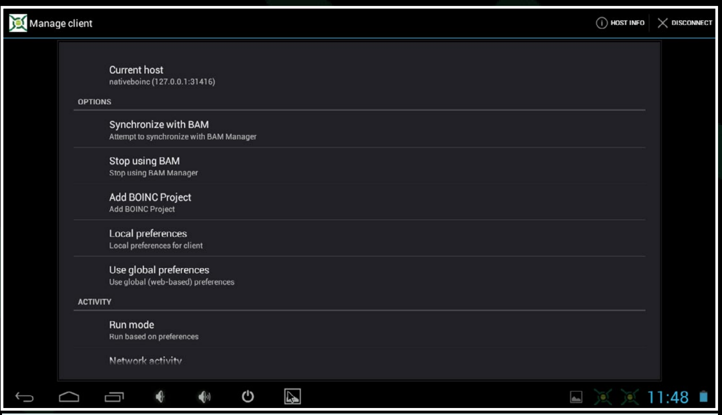
6 - NativeBOINC is running after clicking the “Dismiss” button



nativeboinc (7.0.36)

```
fitsedwrapper 3.41                                      2014-10-13 15:02:02
pogs                                  06:05:37  91.304%  00:29:01
fitsedwrapper 3.41                                      2014-10-13 15:02:02
pogs                                  05:57:00  84.000%  00:59:45
fitsedwrapper 3.41                                      2014-10-13 20:05:10
pogs                                  02:22:38  34.783%  03:46:51
fitsedwrapper 3.41                                      2014-10-13 20:05:11
pogs                                  01:29:29  26.316%  03:31:11
Data collect version 4 4.17                             2014-10-21 09:42:21
WUProp@Home                           02:04:43  34.626%  03:55:19
fitsedwrapper 3.41                                      2014-10-13 20:24:07
pogs                                  00:00:00  0.000%   00:33:04
fitsedwrapper 3.41                                      2014-10-13 20:24:07
pogs                                  00:00:00  0.000%   05:47:22
fitsedwrapper 3.41                                      2014-10-13 21:29:01
pogs                                  00:00:00  0.000%   00:16:32
fitsedwrapper 3.41                                      2014-10-13 21:29:01
```

7 - Task list and settings for the currently running BOINC project



Manage client

Current host
nativeboinc (127.0.0.1:31416)

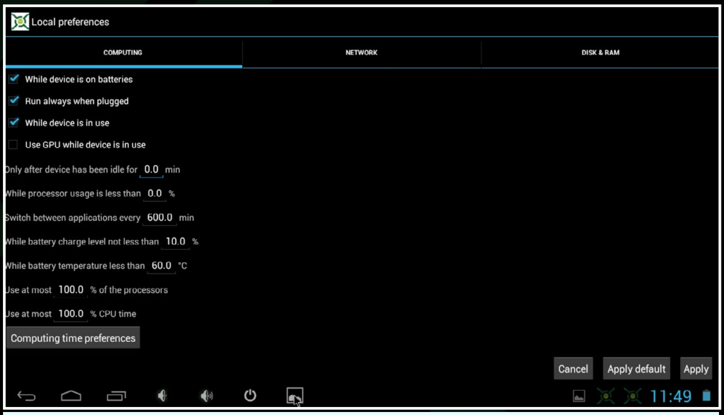
OPTIONS

- Synchronize with BAM
  Attempt to synchronize with BAM Manager
- Stop using BAM
  Stop using BAM Manager
- Add BOINC Project
  Add BOINC Project
- Local preferences
  Local preferences for client
- Use global preferences
  Use global (web-based) preferences

ACTIVITY

- Run mode
  Run based on preferences
- Network activity

8 - Click on “Manage Client” to add other BOINC projects and adjust settings

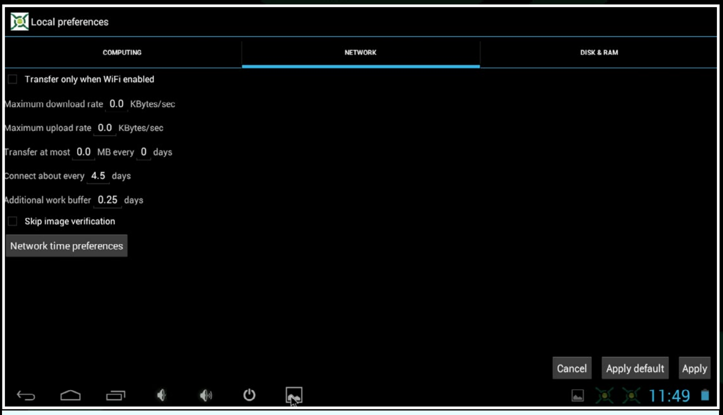


Local preferences

- [✓] While device is on batteries
- [✓] Run always when plugged
- [✓] While device is in use
- [ ] Use GPU while device is in use

Only after device has been idle for 0.0 min
While processor usage is less than 0.0 %
Switch between applications every 600.0 min
While battery charge level not less than 10.0 %
While battery temperature less than 60.0 °C
Use at most 100.0 % of the processors
Use at most 100.0 % CPU time

9 - Click on “Local Preferences”, and set the “Computing” settings to match those shown



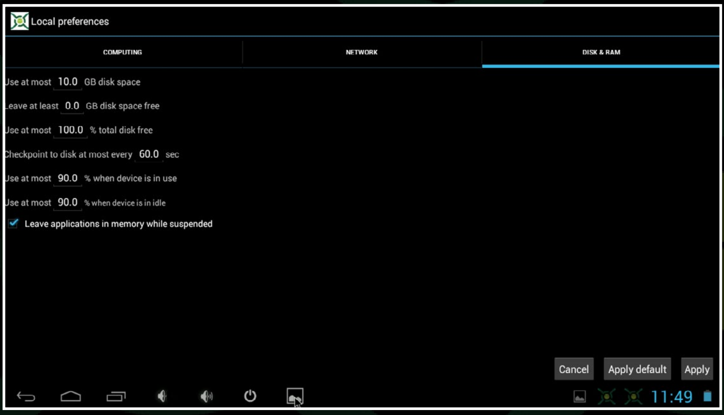
Local preferences

- [ ] Transfer only when WiFi enabled

Maximum download rate 0.0 KBytes/sec
Maximum upload rate 0.0 KBytes/sec
Transfer at most 0.0 MB every 0 days
Connect about every 4.5 days
Additional work buffer 0.25 days

- [ ] Skip image verification

10 - Click on “Network” and adjust the network settings to match your speed preferences



Local preferences

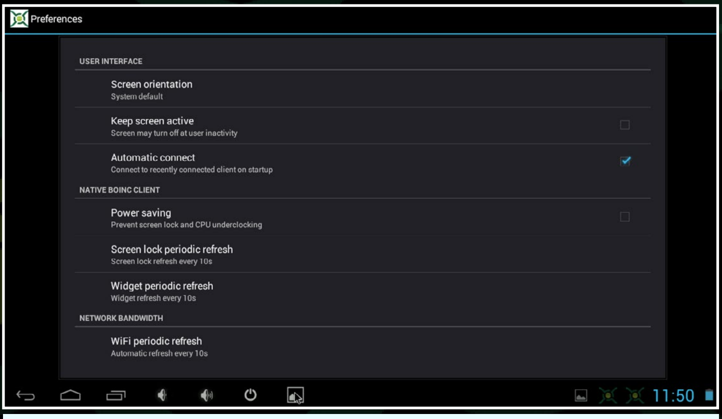
COMPUTING | NETWORK | DISK & RAM

Use at most 10.0 GB disk space
Leave at least 0.0 GB disk space free
Use at most 100.0 % total disk free
Checkpoint to disk at most every 60.0 sec
Use at most 90.0 % when device is in use
Use at most 90.0 % when device is in idle
✓ Leave applications in memory while suspended

Cancel | Apply default | Apply

11:49

11 - Click on "Disk & Ram" and set the options to match those shown

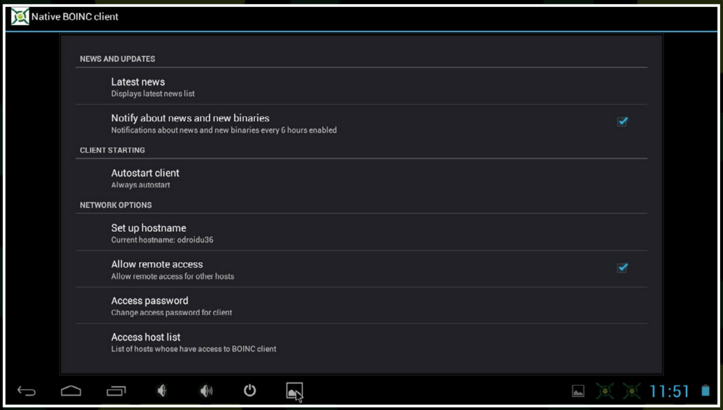


Preferences

USER INTERFACE
Screen orientation
System default
Keep screen active
Screen may turn off at user inactivity
Automatic connect ✓
Connect to recently connected client on startup
NATIVE BOINC CLIENT
Power saving
Prevent screen lock and CPU underclocking
Screen lock periodic refresh
Screen lock refresh every 10s
Widget periodic refresh
Widget refresh every 10s
NETWORK BANDWIDTH
WiFi periodic refresh
Automatic refresh every 10s

11:50

12 - Click on "Preferences" and adjust the user interface options to match those shown

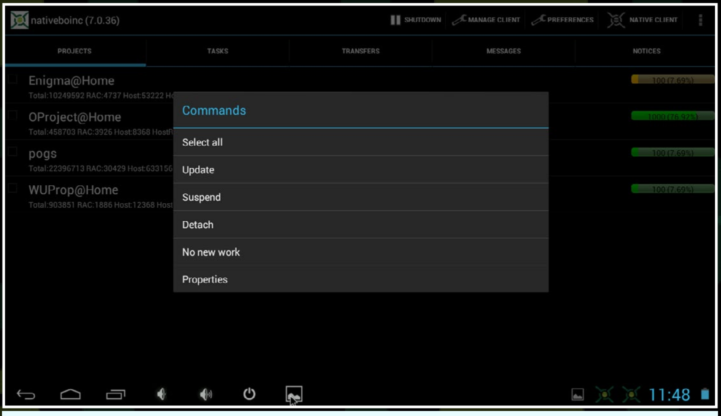


Native BOINC client

NEWS AND UPDATES
Latest news
Displays latest news list
Notify about news and new binaries ✓
Notifications about news and new binaries every 6 hours enabled
CLIENT STARTING
Autostart client
Always autostart
NETWORK OPTIONS
Set up hostname
Current hostname: odroidu36
Allow remote access ✓
Allow remote access for other hosts
Access password
Change access password for client
Access host list
List of hosts whose have access to BOINC client

11:51

13 - Click on "Native Client", then select and enable the Autostart Client option



nativeboinc (7.0.36)

SHUTDOWN | MANAGE CLIENT | PREFERENCES | NATIVE CLIENT

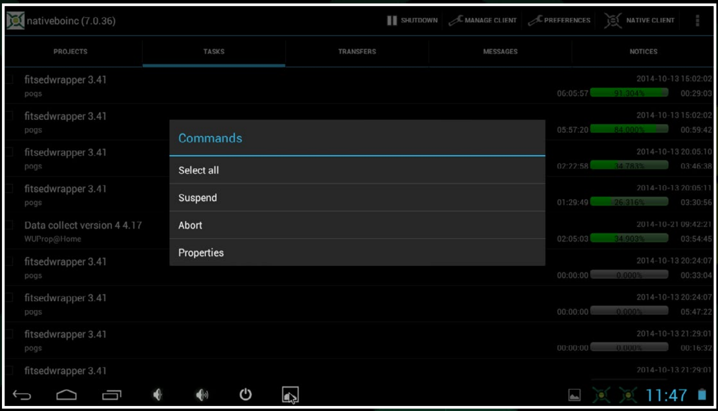
PROJECTS | TASKS | TRANSFERS | MESSAGES | NOTICES

Enigma@Home
Total:10249592 RAC:4737 Host:53222 H
100 (7.69%)
OProject@Home
Total:458703 RAC:3926 Host:8368 Host
1000 (76.92%)
pogs
Total:22396713 RAC:30429 Host:633156
100 (7.69%)
WUProp@Home
Total:903851 RAC:1886 Host:12368 Host
100 (7.69%)

Commands
Select all
Update
Suspend
Detach
No new work
Properties

11:48

14 - The "Projects" tab shows a list of all the projects, along with detailed information about each one. Manage a project by left long-clicking its name



nativeboinc (7.0.36)

SHUTDOWN | MANAGE CLIENT | PREFERENCES | NATIVE CLIENT

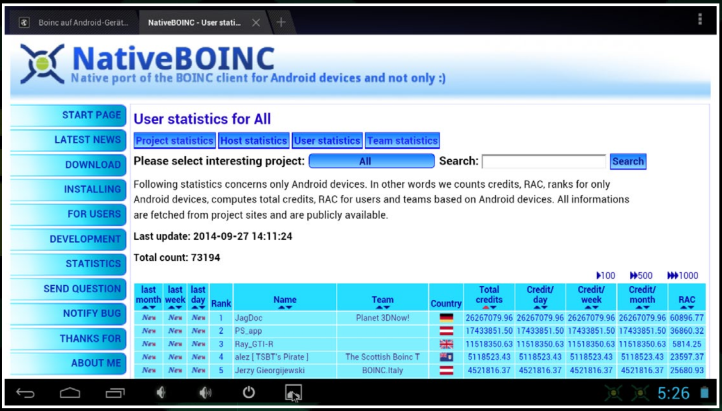
PROJECTS | TASKS | TRANSFERS | MESSAGES | NOTICES

fitsedwrapper 3.41 pogs — 2014-10-13 15:02:02 — 06:05:57 91.304% 00:29:03
fitsedwrapper 3.41 pogs — 2014-10-13 15:02:02 — 05:57:20 84.000% 00:59:42
fitsedwrapper 3.41 pogs — 2014-10-13 20:05:10 — 02:22:58 34.783% 03:46:38
fitsedwrapper 3.41 pogs — 2014-10-13 20:05:11 — 01:29:49 26.316% 03:30:56
Data collect version 4 4.17 WUProp@Home — 2014-10-21 09:42:21 — 02:05:03 34.903% 03:54:45
fitsedwrapper 3.41 pogs — 2014-10-13 20:24:07 — 00:00:00 0.000% 00:33:04
fitsedwrapper 3.41 pogs — 2014-10-13 20:24:07 — 00:00:00 0.000% 05:47:22
fitsedwrapper 3.41 pogs — 2014-10-13 21:29:01 — 00:00:00 0.000% 00:16:32
fitsedwrapper 3.41 — 2014-10-13 21:29:01

Commands
Select all
Suspend
Abort
Properties

11:47

15 - The "Tasks" tab shows a list of the current work units, which may be managed individually



Boinc auf Android-Gerät... | NativeBOINC - User stati...

NativeBOINC
Native port of the BOINC client for Android devices and not only :)

START PAGE
LATEST NEWS
DOWNLOAD
INSTALLING
FOR USERS
DEVELOPMENT
STATISTICS
SEND QUESTION
NOTIFY BUG
THANKS FOR
ABOUT ME

User statistics for All

Project statistics | Host statistics | User statistics | Team statistics

Please select interesting project: All Search: Search

Following statistics concerns only Android devices. In other words we counts credits, RAC, ranks for only Android devices, computes total credits, RAC for users and teams based on Android devices. All informations are fetched from project sites and are publicly available.

Last update: 2014-09-27 14:11:24

Total count: 73194

▶100 ▶▶500 ▶▶▶1000

last month	last week	last day	Rank	Name	Team	Country	Total credits	Credit/ day	Credit/ week	Credit/ month	RAC
New	New	New	1	JagDoc	Planet 3DNow!	🇩🇪	26267079.96	26267079.96	26267079.96	26267079.96	60896.77
New	New	New	2	PS_app		🇦🇹	17433851.50	17433851.50	17433851.50	17433851.50	36860.32
New	New	New	3	Ray_GTI-R		🇬🇧	11518350.63	11518350.63	11518350.63	11518350.63	5814.25
New	New	New	4	alez [TSBT's Pirate]	The Scottish Boinc T	🇦🇺	5118523.43	5118523.43	5118523.43	5118523.43	23597.37
New	New	New	5	Jerzy Gieorgijewski	BOINC.Italy	🇦🇹	4521816.37	4521816.37	4521816.37	4521816.37	25680.93

5:26

16 - List of some ARM-compatible NativeBOINC projects. Everything listed is available for ARM except for OProject, Primegrid, SubsetSum@Home and YAFU

The History Of ubuntu

(so far)

Launched in October 2004

9TH BIRTHDAY

Estimated 22 Million Users[1]

#1 Cloud Operating System[2]

Lenovo, Dell, Acer Laptops Available with Ubuntu

Growing Retail Presence In China & India[3]

One OS, Many Faces: PCs, Phones, Tablets & TVs

2004

4.10 'Warty Warthog'
First version of Ubuntu
GNOME 2.8 Desktop
Firefox 0.9
Free CD Copies Were Available

2005

5.04 'Hoary Hedgehog'
Update Manager Tool Introduced
Suspend & Hibernate Support
Dynamic CPU Frequency Scaling
Supported USB Device Install

5.10 'Breezy Badger'
First Release with Graphical Bootscreen
Simple Add/Remove Software Tool
Serpentine CD Burner App
Ubuntu Logo Added to Menu

Applications Places

2006

6.06 'Dapper Drake'
Network Manager for WiFi
Revamped Theme
GUI Ubuntu Installer
Merged Live & Install CDs
LTS

6.10 'Edgy Eft'
Added 'Tomboy' & 'F-Spot'
Debut of Ubuntu Login Sounds
Firefox 2.0 Included
Upstart Shipped by Default

2007

7.04 'Feisty Fawn'
Provided Windows Migration Assistant
Compiz Effects Support
Easy Install of Restricted Drivers/Codecs
WPA WiFi Network Support

7.10 'Gutsy Gibbon'
NTFS Support (read/write)
Fast User Switching
Compiz Made Default
Gaim (Finally) Updated to Pidgin

2008

8.04 'Hardy Heron'
Pulse Audio Introduced
Transmission & Brasero Apps Added
WUBI Added to Live CD
Netbook Remix Interface Introduced
LTS

8.10 'Intrepid Ibex'
Live USB Creation Tool
Dynamic Kernel Module Support
First Release to Ship With 'Guest Account'
Mobile Internet Connection Wizard

2009

9.04 'Jaunty Jackalope'
Much Improved Boot Time
Notify OSD Introduced
New Boot & Login Screens
Wacom Tablet Support

9.10 'Karmic Koala'
EXT4 Made Default Filesystem
Pidgin replaced with Empathy
Separate Netbook Edition
Ubuntu One Included by Default
Ubuntu Software Centre

2010

10.04 'Lucid Lynx'
Complete Visual Overhaul
Shipped with Plymouth
Social 'Me Menu' Debuts
Window Controls Moved To The Left
LTS

10.10 'Maverick Meerkat'
Unity Introduced in Netbook Edition
'Ubuntu' Font Made Default
Shotwell Replaced F-Spot
Codec Checkbox Added to Installer

2011

11.04 'Natty Narwhal'
Unity Made Default Desktop UI
Netbook Edition Discontinued
Banshee Made Default Music App
Introduced Overlay Scollbars

11.10 'Oneiric Ocelot'
Unity 2D Introduced
Thunderbird Replaced Evolution
Global Menu & Window Controls Set to Hide
LightDM Login Screen Debuts

2012

12.04 'Precise Pangolin'
Rhythmbox Re-Added
'HUD' Introduced To Desktop
Chameleonic Theming
Improved Multi-Monitor Support
LTS

12.10 'Quantal Quetzal'
Unity 2D Retired
Amazon Suggestions in Dash
Innovative Web App Integration
.ISO Image now larger than standard CD

2013

13.04 'Raring Ringtail'
Improved Privacy Features
New-Look Session Exit Menus
Ubuntu One Sync Menu
Improved Interface Animations

13.10 'Saucy Salamander'
Faster Unity Performance
'Smart Scopes'
Ubuntu for Phones 1.0
64bit Becomes Recommended Download

2014

14.04 'Trusty Tahr'
To be releaed April 2014
Will be Long-Term Support Release
LTS

[1]Source: Canonical
[2]Source: goo.gl/hkvve4
[3]Source: goo.gl/xxJrGo

By OMG! UBUNTU!
www.omgubuntu.co.uk

BUILDING A BOINC MONSTER

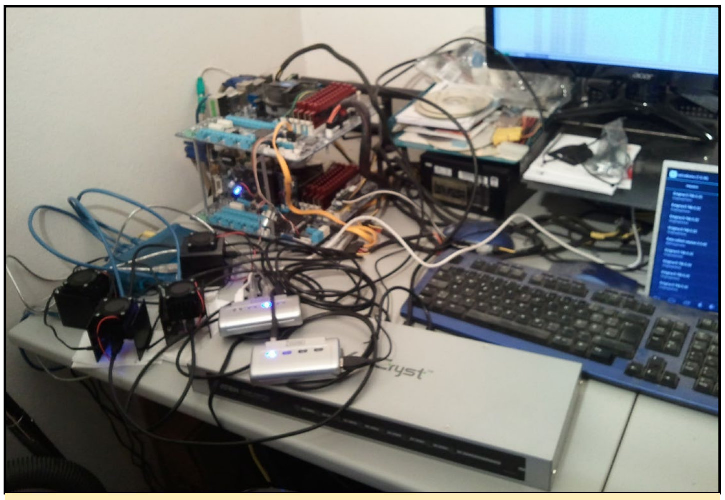
96 CORES FOR ONLY 135 WATTS

by Uli Abromeit

In January 2013, I decided to install the Android version of BOINC on an ODROID-X2, so that I could use it to help study global warming, discover pulsars, and do many other types of scientific research. BOINC was created at University of California Berkeley as a way to recycle unused CPU time in order to solve large mathematical and statistical problems efficiently. It distributes the workload to many computers, which can then process the calculations whenever the computer isn't in use.

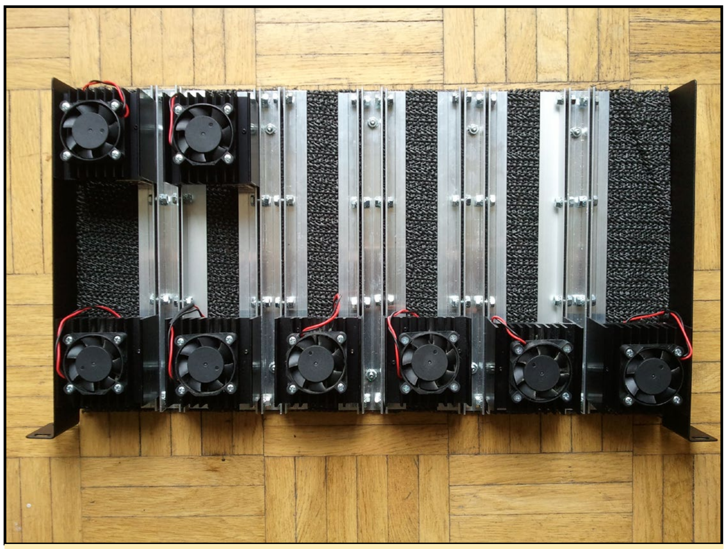
Many BOINC projects are available for the Android ARM platform, and can be found on the official list of supported projects at `http://bit.ly/1r4wpzu`. My BOINC cluster is designed to run BOINC constantly 24/7, using a total of about 135W. Over time, I have added 3 more X2s and an XU, along with some inexpensive U2s and U3s.

I decided that it was finally time to do something about the chaos on my workbench, so I designed a rack to hold all of the ODROIDs, which I call the "BOINC Monster". In order to organize the BOINC computers, I used a 19" rack with some modifications.



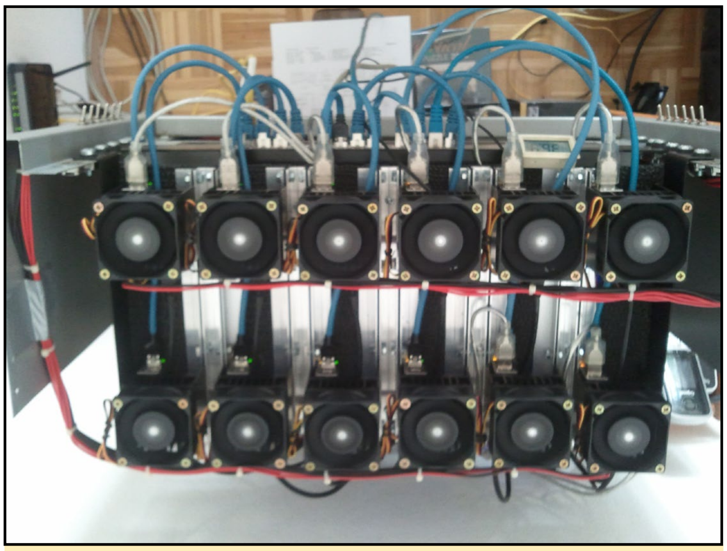
The BOINC cluster workbench prototype

ODROID-U2 x 12
24-port Network switch
8-port HDMI switch x 2 (Aten VS0801H)
5V 20A PSU x 2 (TDK-Lambda HWS100-5/A)



Preliminary design of the BOINC cluster, with the 40x40mm fans installed

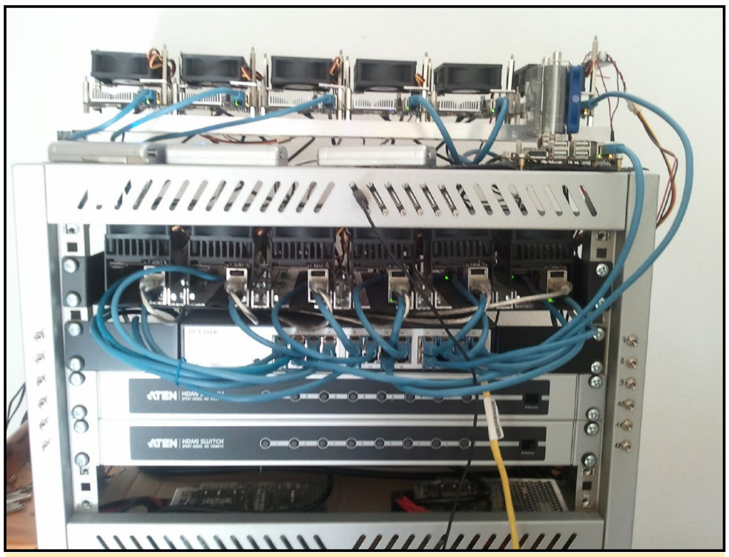
After experimenting with the arrangement, I found that the U2s ran a little bit hot and noisy with the stock fans, so I changed them to 60x60x25mm 12V Sharkoon Powerfans and connected them to the U2 fanports.

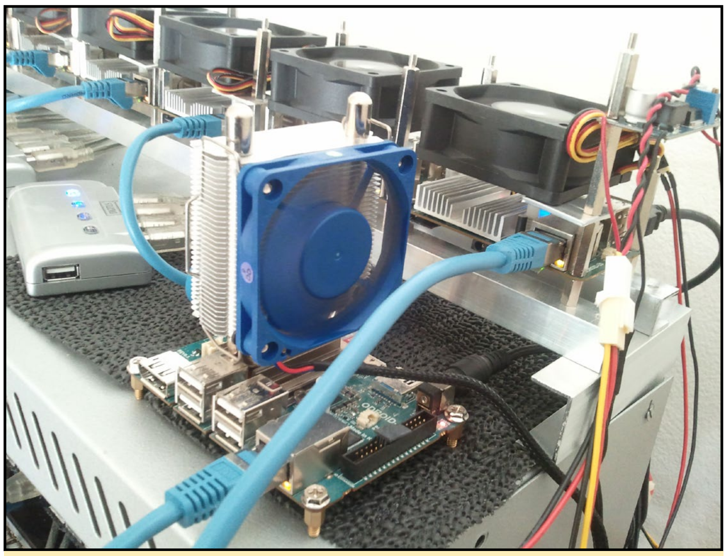


BOINC cluster with 12V fans attached

The U2s are connected to a Motorola Lapdock using the HDMI switches, and are controlled with a mouse connected over some 4-port USB-sharing devices. I prefer controlling them with USB peripherals rather than remotely.

Once I got the U2 rack arrangement working well, I added 6 ODROID-U3 computers to the 12 ODROID-U2s that were already mounted. Again, I found that with the smaller heatsink, the U3 ran too hot using the 12V 60x60x25mm fans at 5V, so I added an adjustable DC-DC stepup converter and set it to deliver 9V to the fans, which improved their effectiveness.


Combination U2, U3 and XU cluster


Closeup of the adjustable DC-DC step converter

The next step for the project is to add a third HDMI switch so that all of the new ODROIDs can use the Lapdock monitor, and of couse add some more computers to the BOINC monster!

MICROSOFT-FREE MINING WITH FREEMINER

NOW THAT MOJANG HAS SET SAIL FOR REDMOND, HOW ABOUT AN OPEN SOURCE GAME?

by Bruno Doiche

The FreeMiner world resembles Minecraft

Shock and awe rippled through the entire Internet when everyone's favorite sandbox game world was purchased by Microsoft for $2.5 billion! Now what? Well, you can celebrate with a full-featured open source version of Minecraft that you can run on your ODROID:

```
$ sudo apt-get install git subversion build-essential cmake
libbz2-dev libpng12-dev libjpeg8-dev libfreetype6-dev
libxxf86vm-dev libgl1-mesa-dev libsqlite3-dev libvorbis-dev
libopenal-dev libcurl4-openssl-dev libluajit-5.1-dev
libleveldb-dev libsnappy-dev libgettextpo0 libmsgpack-dev
libgles1-mesa-dev libgles2-mesa-dev
```

Download the irrlicht application using either Subversion or Github:

```
$ svn checkout svn://svn.code.sf.net/p\
/irrlicht/code/branches/ogl-es irrlicht

$ git clone -b ogl-es-svn --recursive \
https://github.com/freeminer/irrlicht.git \
irrlicht
```

Compile Irrlicht:

```
$ make -j4 -C irrlicht/source/Irrlicht
```

Download the Freeminer source code:

```
$ git clone --recursive \
https://github.com/freeminer/freeminer.git
```

Compile Freeminer and set the video driver to Open GLES:

```
$ cd freeminer
$ cmake . -DENABLE_GLES=1 \
-DIRRLICHT_INCLUDE_DIR=../irrlicht/include \
-DIRRLICHT_LIBRARY=../irrlicht/lib/Linux/\libIrrlicht.a
$ make -j4
$ echo "video_driver=ogles1" >> freeminer.conf
$ echo "enable_shaders=0" >> freeminer.conf
```

That's it! Freeminer, like Minecraft, also has servers available for enjoying multi-player games. To learn more about Freeminer and join the forums, visit the project home page at http://freeminer.org.

FAKE86

AN EXTREMELY FAST 8088/8086 VIRTUALIZER

by Jeremy (Cartridge) Kenney

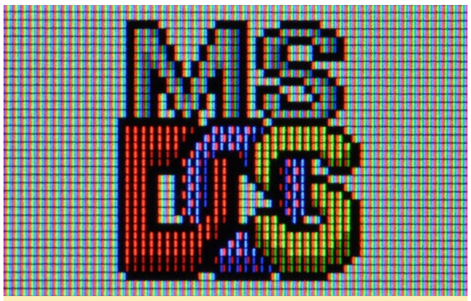
In the 1980s, this was considered a portable model, and weighed about the same as a suitcase full of bowling balls. With Fake86, you can now fit an 8086 in your front shirt pocket!

Have you ever felt that none of the IBM PC emulators and virtualizers available are up to the challenge of running intensive DOS programs? Although there are nearly unlimited choices for Intel emulation, it can be difficult for those without expert-level skills to get those 8088 programs working.

The popular Qemu (http://www.qemu.org) is a bit bulky, and requires adding lots of strings in order to get it properly set up. Bochs (http://bit.ly/1nuFeYj) has a similar learning curve to Qemu, but with even more strings to add. Dosbox (http://www.dosbox.com) does more or less a good job at making setup easy, but comes with the task of constantly configuring the cycles and other emulation options in the case that the game is glitching, or has vertical lines due to using a special engine to render the game playable.

What if there was an emulator/virtualizer that focuses on a certain CPU, a couple video cards and an era-appropriate Adlib/Soundblaster audio card? Luckily, Fake86 (http://bit.ly/1wcIpV6) specializes in virtualizing the best specifications of the time, with full support for the 8088/8086 and 16-bit 80186 instruction set. Here are some of its features:

- The CPU engine included with Fake86 is a pure interpreter that is still capable of reasonably fast execution
- Audio support includes emulation of the Sound Blaster, Adlib FM music card, Disney Sound Source, and standard PC speaker
- A standard Microsoft-compatible serial mouse can be emulated on COM1
- The hard disk and floppy emulation on interrupt 13h uses disk image files
- It's capable of running any 16-bit flavor of DOS (for example, MS-DOS, DR-DOS, PC-DOS, and FreeDOS)
- Hercules, CGA, MCGA, and VGA (except 640x480 16-color) video emulation is supported
- Ethernet emulation with libpcap and WinPcap is enabled
- Windows 1.0 and 2.0 work flawlessly, and some tweaks were done to support not all but some games requiring higher than a 286 (for example, Wolfenstein)


The original MS DOS logo looks like it was designed by a graffiti artist

After reading all those features that Fake86 focuses on, you would have never thought that DOS could be so massive! It's even better when you run it on your ODROID in high definition (HD). All you need is a couple of minutes of setup, rather than searching Google to fix for game glitches.

1. Read through the Fake86 in the forum post at http://bit.ly/1pgoyO8, and download the installation package from http://bit.ly/1Dx0Kzj. Unzip the file and read the "readme" file before continuing.

2. Open up a Terminal window, and type the following to install Fake86 (the filename may be different):

```
$ sudo dpkg -i <fake86_0.12.9.19-
1_armhf.deb>
```

3. Create a blank disk image for you to use as a hard drive, which will be explained further below. Open up a Terminal window and type:

```
$ sudo dd bs=1M count=100 if=dev/
zero of=./nameofyourimage.img
```

Note that the parameters "bs=1M count=100" will result in a 100MB unformatted disk image. If you wish to have more disk space in your image, increase the "count" integer.

4. To start Fake86, type the following:

```
$ sudo fake86 -hd0 nameofyourim-
age.img
```

Alternatively, if you want to boot from an external USB floppy drive or CD/DVD-ROM Drive, insert your floppy disk or DVD and type:

```
$ sudo fake86 -hd0 nameofyourim-
age.img \
-fd0 Path/To/Media/Floppy/Or/
CDROMDrive -boot 0
```

Upon booting the new disk image, a Fake86 window will appear and show the Turbo XT BIOS. From there, it will ask you to enable large disk support if you happen to have increased the size of your image. Press Yes, then press 1 to create a primary DOS partition, and press 1 again. A verification will occur so this may take a second or two.

Create your partition and leave the label empty. Once created, you can exit out of fdisk and close down Fake86 from the X button on the window. Your progress is saved automatically.

5. Format the drive by opening Terminal and typing:

```
$ sudo fake86 -hd0 nameofyourim-
age.img -fd0 \
/Path/To/Media/Floppy/Or/CDROM-
Drive -boot 0
```

At the DOS prompt, type:

```
> format C:
```

Confirm that the data may be overwritten, which is safe because it doesn't erase anything outside the image that you have created. After the format is complete, the image is ready for DOS and Windows. The setup of DOS 5.0.3 is very easy, which can be installed on your virtual machine once you have booted from the setup portion of DOS. Once DOS is running, you're ready to play any game or program that supports 8088/8086/80186.

Fake86 has the following options:

-fd0 -fd1: These are your floppy drives

-hd0 -hd1: These are your hard drives

-boot 0: Will boot from the first floppy drive

-boot 1: Will boot from the second floppy drive

-boot 128: Will boot from the first hard drive after your installation of your OS, unless you have put the installation files on the hard drive

-bios <filenameofbios>: Specifies a BIOS that you may have dumped or downloaded from the Internet

-nosound: This disables the use of sound, which is usually used when working only with DOS

-resw -resh: These set the screen resolution, which requires both parameters: "-resw 1280 -resh 720" is for 720p, and "-resw 1920 -resh 1080" creates a 1080p screen

-smooth: This will apply smooth rendering to your screen filter

-fullscreen: Use this command with resw and resh if you wish to have a scaled fullscreen. If used

alone, you will get a native unscaled screen in fullscreen

-ssource: This command enables Sound Source on LPT1

-latency and -samprate: These commands are best left alone, but if you encounter sound glitches

you can use these options to fix them by typing "-latency 150 -samprate 44100" (or any other number of your choice)

-console: This command enables your Terminal to act as an interactive console to change and eject floppies while the emulator is running.

Now you can use Fake86 to work, play games, use the Internet and invite your friends over for a game of Wolfenstein 3-D on a 1080p monitor!

Wolfenstein 3-D for MS-DOS was revolutionary for its first person perspective

LINUX GAMING: DOSBOX EMULATOR

PLAY YOUR ORIGINAL DOS GAMES IN HD

by Tobias Schaaf

DOSBox is an x86 DOS emulator that not only emulates the x86 architecture, but also offers a common 1990s-era DOS environment. With DOSBox, you can replay your old games on modern hardware, and there are many interesting and legendary DOS applications that aren't available for Windows or Linux.

DOSBox is very stressful on many computers, since you normally need a high-end PC to emulate a 486 at 33MHz. Since the ODROID uses a completely different architecture (ARM vs x86), it has even more work to do during emulation. Despite its complexity and multiple layers, DOSBox runs surprisingly well on the ODROID platform.

Some time ago, I compiled a custom ARMv7-optimized version of DOSBox, which appeared to be running faster than the stock DOSBox version that comes with the official distribution. I took some time to compare these versions and find out exactly what is improved by using an ARMv7-optimized build.

Below you will find a series of side-by-side tests that highlight the differences between the generic build of DOSBox, and a build that is specifically compiled for ARM. The custom build of DOSBox for ARMv7 may be downloaded from my repository at http://bit.ly/1DhCv6l.

Configuration

Configuring DOSBox can sometimes be difficult. While most games run fine with the basic settings, others only run with a very specific configuration, so I chose a set of values that worked best for the original version of the game Quake, since it's very demanding on the hardware.

What's remarkable about Quake is that the game itself is in 3D without requiring a graphical desktop environment. In contrast to games like "Duke Nukem 3D" ,which contains some 3D objects and use 2D sprites in many situations, Quake was already using 3D models, similar to the models used in later games on Windows, which was very impressive for that time.

There was no easy way for me to find the right settings, and after a period of experimentaiton, I ended up with the following results, with frameskip and aspect ratio turned off:

```
core=dynamic
cputype=pentium_slow
cycles=fixed 32000
cycleup=500
cycledown=300
memsize=32
scaler=normal3x
```

Dynamic cores should be used for any value of fixed cycles over 20,000. Pentium_slow is the CPU with the most features, and I set the cycles to 32,000, which is very high. Some test programs reported it to be a 1285 MHz fast Pentium CPU. I chose such a high number because of Quake, since at 32,000 cycles it gives the most fluid experience on both DOSBox versions.

Tests

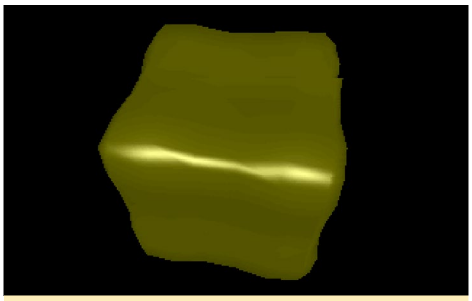
After performing a variety of tests, I discovered that it was actually hard to find some good benchmarks. I remembered some benchmarking applications from back when DOS was popular, but they were hard to find. However, I did find a test environment for performing different benchmarks under DOS called DOS Benchmark, which is available for download from http://bit.ly/1ttzaRR.

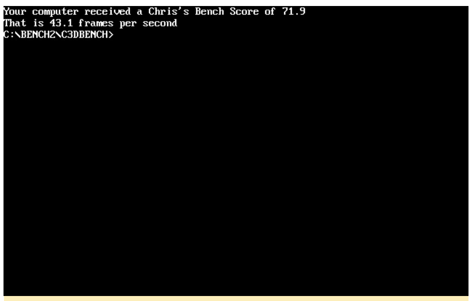
DOS Benchmark offers CPU, GPU, and memory tests, as well as demo versions of the games Doom and Quake which can be used for benchmarking the environment. I tried to run every test available, but not all of them were working. I did find a few that performed nicely.

For instance, I found a test with a spinning 3D cube running in DOS, which has some great visuals, and ran relatively fast on the ODROID, which can be seen in the screenshots on the following page.

3DBench test

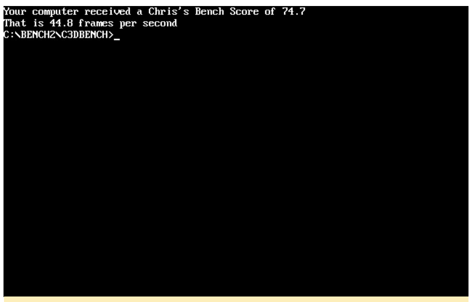
The ARMv7-a optimized version was nearly 17% faster in this test. Unfortunately, that test is not very reliable if you change the CPU cycles as I did. I was able to achieve results with over 200 FPS with values like 100,000 CPU cycles, but even with these high numbers, the


Spinning cube under DOS



```
Your computer received a Chris's Bench Score of 71.9
That is 43.1 frames per second
C:\BENCH2\C3DBENCH>
```

Stock Debian version of DOSBox



```
Your computer received a Chris's Bench Score of 74.7
That is 44.8 frames per second
C:\BENCH2\C3DBENCH>_
```

ARMv7-optimized version of DOSBox

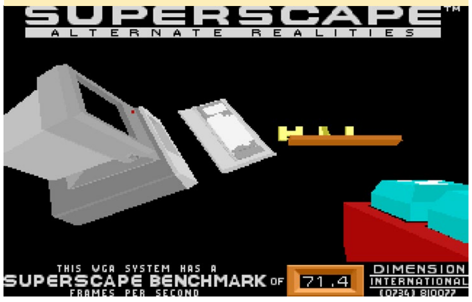
emulator was far from working better or even faster. I could see that the video output was lagging and skipping frames, but the test still got high scores.

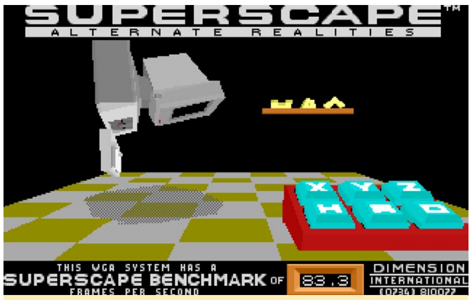
Benchmark

The CPU tests were showing that the ARMv7-a optimized version performs somewhat better. An improvement of around 30% was common when it came to CPU computing comparisons.

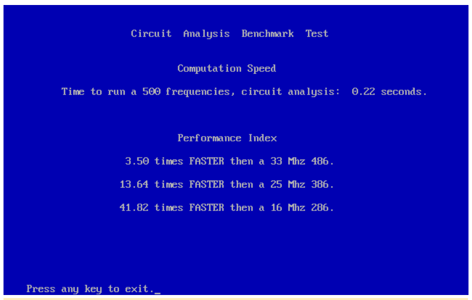
While some benchmarks worked better in the ARM version, I saw some major issues in some tests when it came to the ARMv7-a optimized version. Some tests didn't even run on the ARMv7-a optimized version of DOSBox, or caused strange behavior. Only the stock Debian version was running 100% of the tests correctly.

3D Bench test using stock build



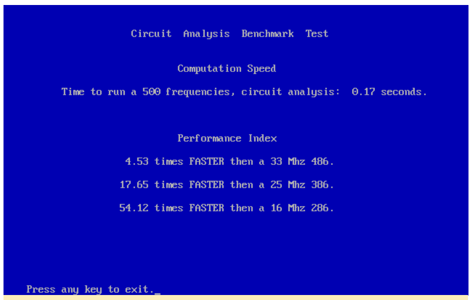


3D Bench test showing difference in results using ARM build



```
Circuit  Analysis  Benchmark  Test

Computation Speed

Time to run a 500 frequencies, circuit analysis:  0.22 seconds.


Performance Index

3.50 times FASTER then a 33 Mhz 486.

13.64 times FASTER then a 25 Mhz 386.

41.82 times FASTER then a 16 Mhz 286.


Press any key to exit._
```

CPU benchmark using the stock build

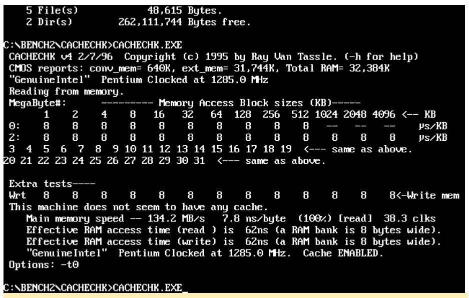


```
Circuit  Analysis  Benchmark  Test

Computation Speed

Time to run a 500 frequencies, circuit analysis:  0.17 seconds.


Performance Index

4.53 times FASTER then a 33 Mhz 486.

17.65 times FASTER then a 25 Mhz 386.

54.12 times FASTER then a 16 Mhz 286.


Press any key to exit._
```

CPU benchmark test using the custom ARM build

For example, there was a memory test that used blocks of different sizes, and did some operations on them, so that in the end the different blocks added up to 24MB in total. It operated on 384 x 64KB blocks and gave a result on how fast the memory did the computing.

The same test on the optimized version resulted in very different results. Not only did the ARM test take approximately 10 times as long to run, the values were completely inaccurate. Instead



```
   5 File(s)          48,615 Bytes.
   2 Dir(s)      262,111,744 Bytes free.

C:\BENCH2\CACHECHK>CACHECHK.EXE
 CACHECHK v4 2/7/96  Copyright (c) 1995 by Ray Van Tassle. (-h for help)
 CMOS reports: conv_mem= 640K, ext_mem= 31,744K, Total RAM= 32,384K
 "GenuineIntel"  Pentium Clocked at 1285.0 MHz
 Reading from memory.
 MegaByte#:      --------- Memory Access Block sizes (KB)-----
       1    2    4    8   16   32   64  128  256  512 1024 2048 4096 <-- KB
 0:    8    8    8    8    8    8    8    8    8    8   --   --   --    µs/KB
 2:    8    8    8    8    8    8    8    8    8    8    8    8    8    µs/KB
 3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19  <--- same as above.
20 21 22 23 24 25 26 27 28 29 30 31  <--- same as above.

 Extra tests----
 Wrt   8    8    8    8    8    8    8    8    8    8    8    8    8<-Write mem
 This machine does not seem to have any cache.
    Main memory speed -- 134.2 MB/s   7.8 ns/byte  (100%) [read]  38.3 clks
    Effective RAM access time (read ) is  62ns (a RAM bank is 8 bytes wide).
    Effective RAM access time (write) is  62ns (a RAM bank is 8 bytes wide).
    "GenuineIntel"  Pentium Clocked at 1285.0 MHz.  Cache ENABLED.
 Options: -t0

C:\BENCH2\CACHECHK>CACHECHK.EXE_
```

CACHECHK was only working on the stock Debian version of DOSBox, and properly identified the CPU

of 24MB in results, it added up blocks of 512 MB and more, at a ridiculous speed.

Some tests went so high, that they went out of scale and resulted in either a negative speed, or with high exponents calculating ten thousands of megabytes per second. Other tests didn't start at all, or caused the emulator to freeze.

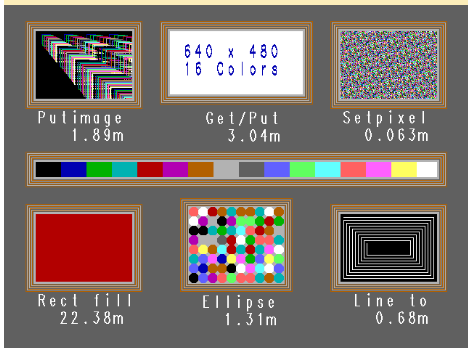
Testing tools

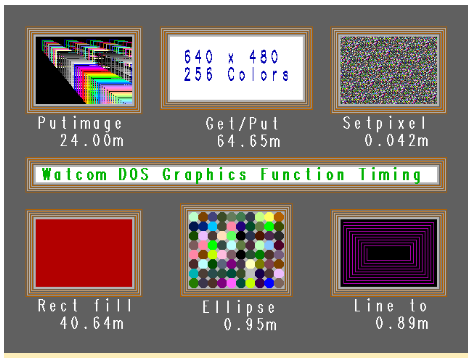
I tried some other testing tools for benchmarking the graphics performance of the system, like the spinning cube and VideoDOS, which sometimes had very odd results. Because the graphical tests are just benchmarks, and don't directly relate to gameplay responsiveness, I did some hands-on testing with some of my favorite games.

Games

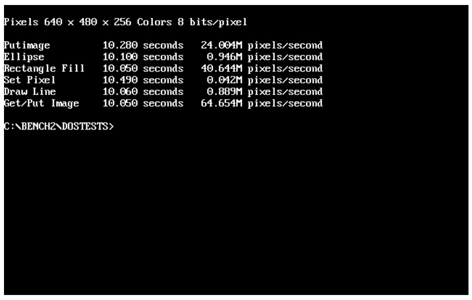
The benchmark package included two games, Doom and Quake, since both were very commonly played during the golden age of DOS, and offered some nice benchmark features in demo mode. However, the benchmark build into Doom did not work correctly, and

Graphical test on the stock Debian build of DOSBox

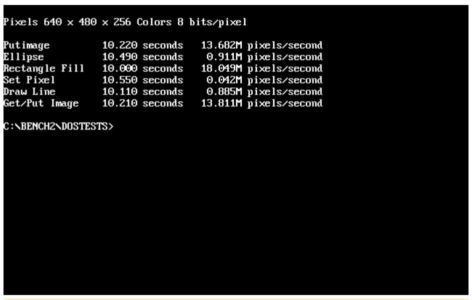




Graphical test on the ARM build of DOSBox. This graphic tests gave odd results: some tests seemed to run faster with more colors and in higher resolutions, while others seemed normal



```
Pixels 640 x 480 x 256 Colors 8 bits/pixel

Putimage         10.280 seconds   24.004M pixels/second
Ellipse          10.100 seconds    0.946M pixels/second
Rectangle Fill   10.050 seconds   40.644M pixels/second
Set Pixel        10.490 seconds    0.042M pixels/second
Draw Line        10.060 seconds    0.889M pixels/second
Get/Put Image    10.050 seconds   64.654M pixels/second

C:\BENCH2\DOSTESTS>
```



```
Pixels 640 x 480 x 256 Colors 8 bits/pixel

Putimage         10.220 seconds   13.682M pixels/second
Ellipse          10.490 seconds    0.911M pixels/second
Rectangle Fill   10.000 seconds   18.049M pixels/second
Set Pixel        10.550 seconds    0.042M pixels/second
Draw Line        10.110 seconds    0.885M pixels/second
Get/Put Image    10.210 seconds   13.811M pixels/second

C:\BENCH2\DOSTESTS>
```

Results of VideoDOS of the ARM optimized version (top) and the stock Debian version (bottom)

claimed to nearly always be running at nearly full speed, although it was far from it.

Instead of using the built-in benchmarks, I did my own testing and compared the time that it took the games to do a full demo run. The results were very surprising for me: Demo 3 took about 108 sec to complete a full demo run on the ARMv7-a optimized version of DOSBox, and on the stock Debian version of DOSBox it took 156 sec instead. That's a nearly 45% increase in speed for the ARM version.

Even more dramatically, you could

The Doom DOS version is playable but not very smooth on DOSBox, and performs much better as a native Linux port

see the difference when playing Quake. A demo run for Demo 3 took 147 sec on the optimized version and 248 sec on the stock Debian version, which is approximately 70% faster!

After all the benchmarking, I wanted to see how well the emulator performs in a real gaming experience, and soon found out that the settings that I had originally chosen did not work well for any other games, so I changed the settings again and ran a couple of test games. After I tuned down the cycles to 6,000 instead of 32,000, Dune 2 was running perfectly fine, with nice, smooth gameplay. The sounds, music and voices were all good, and I had no other issues.

I also tried a couple more demanding games, such as Prisoner of Ice, which is a very nice adventure game with some movie cut-scenes and an option to either run in 320x240 resolution or in 640x480. The last one even offered some other features such as better fonts. Both versions were running fine on DOSBox. I also found the same superior performance while playing Space Quest 6.

Results

The ARMv7-a optimized version runs significantly better than the stock Debian version of DOSBox. To estimate, the optimized version is, on average, 10 to 15% faster than the version from the Debian repository. Sometimes, it was even far faster than that, such as when running Quake.

The faster results seem to be related to some math optimizations inside the emulator, which may also create issues as a side effect, especially with memory operations. This, in turn, may cause glitches in some games, or prevent them from running properly. Besides that, the ARM-v7a optimized version is the better version when it comes to speed.

From my previous testing, I can say that it's even fast enough to handle Windows 3.11 or even Windows 95. Most games should run well on both emulators, but run just a little better on the ARM optimized version.

Game	Cycles	Infos	Comments
Sid Meier's Colonization	1,500-3,000	Game runs best with rather low cycles. Besides that, it's running very fine with no issues or sound drops. However, the intro on the first game start takes a long time to play through.	
Shadow Warrior	15,000-20,000	The game is laggy, and not playable	
Terry Pratchett's Discworld 1	3,000-6,000	Game ran fine without any issues	
Syndicate	6,000-10,000	Game ran fine without any issues	Does not run with glshim
Wing Commander I	2,000-4,000	Game ran fine without any issues. In my opinion, the Amiga version has a much better soundtrack	You should use a 3x scaler here
Prisoner of Ice (640x480)	2,000-8,000	Game ran fine with only a minor issue with the sound cracking occasionally	
Space Quest 6	~12,000	Game mostly runs at full speed, but has some slight stuttering in the music, and the text is too fast	
Dune 2	3,000	Game seems a little slow but generally good and without issues	
XCom Series	1,000-15,000	Works well with only slight speed issues	
Dark Legions	~20,000	Works well with only slight speed issues	

Additional configuration

When I was done with the testing, I played some games and changed my settings to the following options, which I found worked well with many games:

```
core=auto or dynamic
cputype=auto
cycles=fixed 3000
memsize=31
```

I also discovered that DOSBox is able to use glshim together with its opengl renderer using the "output" option:

```
output=opengl
```

Finally, I changed the sdl settings:

```
fullscreen=true
fulldouble=true
fullresolution=1280x720
windowresolution=original
output=opengl
```

These options start the game in fullscreen mode, and when used together with LD_LIBRARY_PATH=/usr/local/lib/, you can run the emulator with OpenGL support.

Other games

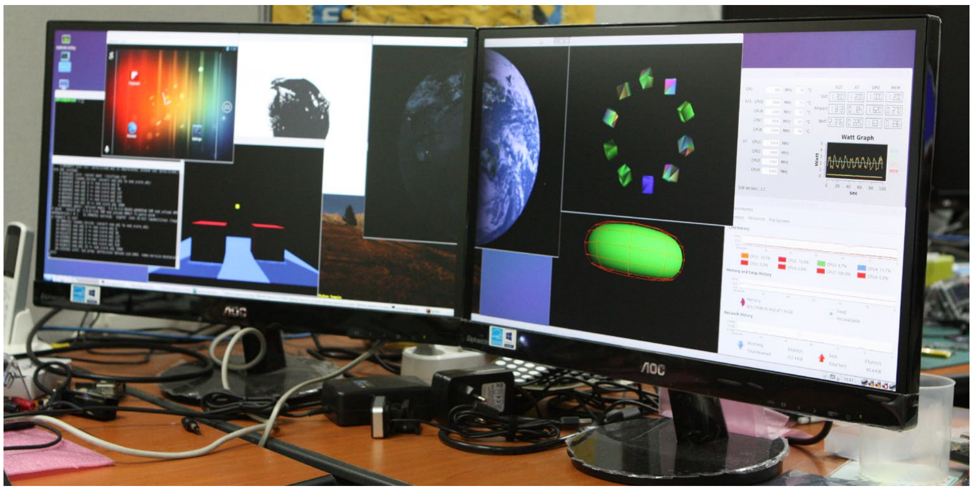
As you can see from the chart on the previous page, games vary a lot in playability, and there is no single setting file that works with all games. I suggest starting at a cycle value of 3,000, and increasing the value until the game runs slow again, then taking a few steps back. This should result in the optimum playability for your favorite DOS games.

MS-DOS is how the Microsoft empire began, when Bill Gates licensed it to IBM

```
Starting MS-DOS...

C:\>_
```

USING DUAL MONITORS WITH AN ODROID-XU3

by Justin Lee



In this article, I'll demonstrate how to use two monitors to extend the desktop of an ODROID-XU3, as seen at ARM TechCon 2014. Each monitor can display the same image, or be configured to display separate images, allowing the 1920 x 1080 desktop to be extended to 3840 x 1080.

Update the kernel

Open the ODROID Utility, which is included as a link on the desktop of all the official Ubuntu Hardkernel images. If the link has been removed, open a Terminal window and type the following to start the utility.

```
sudo odroid-utility.sh
```

Select Option 2, then Option 1 to update the kernel, which will take a few minutes to install.

Install build tools

```
apt-get install device-tree-compiler
```

Enable DisplayPort

```
fdtput -t s /media/boot/exynos5422-odroidxu3.dtb /dp-controller@145B0000 status "okay"
sudo reboot
```

Setup

After the reboot has completed, you'll notice that HDMI and DisplayPort outputs are the same. This is called "cloned" output, which produces the same image on both monitors. To allow each monitor to display a separate image, type the following into Terminal:

```
xrandr --output HDMI-1 --pos 2080x0
```

This enables the DisplayPort on XU3 by adding a default 1080p monitor. If, for some reason, you require a different resolution other than 1080p, please refer to the forum post at http://bit.ly/1zdGd4m, which explains how to properly configure the DisplayPort for alternative resolutions.

OS SPOTLIGHT: CODE MONKEY

HARDWARE AND SOFTWARE DEVELOPER'S ALL-IN-ONE OPERATING SYSTEM IMAGE

Code Monkey for the X and U series may be downloaded from http://bit.ly/IrhHymu

by Rob Roy

The community image Code Monkey is intended for developers of both hardware and software applications. Based on LXDE (Lubuntu) with Thunar as the default file explorer, it includes many of the popular Interactive Development Environments (IDE) and programming languages available from the Ubuntu Software Center:

Bluefish Web Editor
Code::Blocks IDE
CodeLite
Geany
Monkey Studio IDE
Netbeans IDE
Ninja IDE
JRuby
PHP 5
QT 5
Arduino IDE
Scratch
Squeak
Android Debug Bridge (ADB)
GCC
Oracle JDK 8
Python

Code Monkey also include standard desktop applications, including Firefox and Chromium, a video player (Xine), GNU Image Manipulation Program (GIMP), Transmission torrent client, PuTTY Telnet and SSH client, and an open source FTP application called Filezilla.

There are days that a coder considers himself special, but there are other days where he goes to the tattoo parlor and gets a tattoo just like this picture, to humble himself

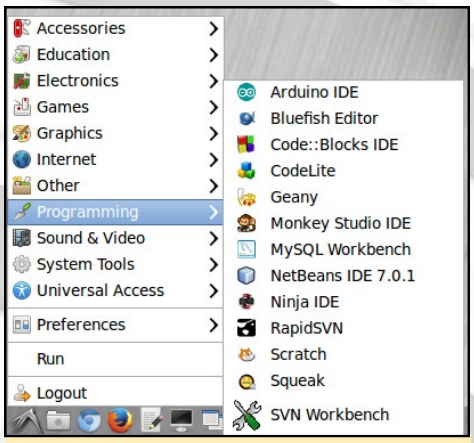
Servers

So that both backend and frontend coding may both be done on the host development ODROID, Code Monkey's development environments work together with several local servers including Tomcat, Samba, MySQL, VNC, and Apache 2. The popular code management tools Git and Subversion are also available via the right-click menu in File Explorer.

Although Code Monkey boots straight to desktop without requiring a password, the default username "odroid" has a password of "odroid". The root access password, required for "sudo" commands, is also "odroid".

Bluefish

For web development, Bluefish Editor (`http://bit.ly/1xAGg4Q`) supports many web languages such as JavaScript, Ruby and jQuery, while also serving as a great application environment for more traditional development languages like C++ and Python. I personally use it to write websites using jQuery, Angular JS, CSS3, and HTML5, since my prefer-

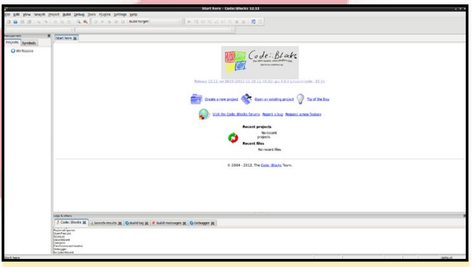


Code monkey comes fully packed to start programming

ence is for a fast, responsive and straightforward text editor.

Code::Blocks, CodeLite and Geany

The open source projects Code::Blocks (`http://www.codeblocks.org`) and CodeLite (http://codelite.org) offer excellent C++ tools, each with an extensible workspace that can also be modified and recompiled from source. Geany (`http://bit.ly/132QrpP`) also compiles C/C++ as well as many other languages, including C#, Go and Perl.



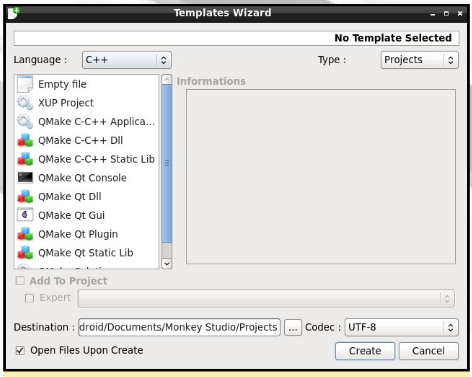
Code Blocks applications interface

Monkey Studio

Monkey Studio IDE is primarily intended for Qt development (`http://qt-project.org`) which is an evolving language, that focuses on cross-platform compatibility. The Razor desktop that was introduced with Ubuntu 13.10 was written entirely in the Qt language.

Ninja

Ninja IDE (`http://ninja-ide.org`) is an open source editor that can be used to



Monkey Studio IDE Application

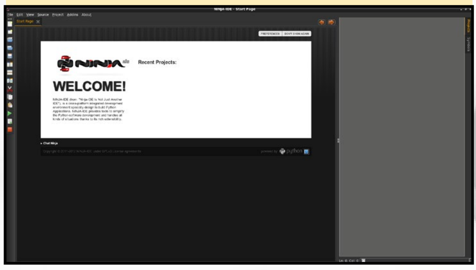
develop Python applications. It stands for Ninja Is Not Just Another IDE, and is maintained by a small team of dedicated, passionate developers. Python (`https://www.python.org`) is one of the primary languages used on the Linux platform, is very easy to learn, and has thousands of third-party modules available for free download from the Python Package Index.

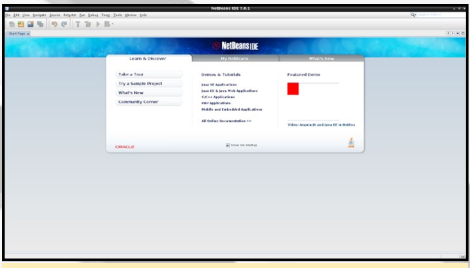
Netbeans

Netbeans IDE (`https://netbeans.org`) is one of the most full-featured development environments available, interpreting a wide range of modern languages including SASS, Java, HTML5, PHP, C/C++ and many more. It also has an enormous number of plugins available, and can be used to develop large professional-quality websites and desktop applications.

Before starting Netbeans for the first time, the Java virtual machine and libraries must be selected. To do so, update the file "`/etc/netbeans.conf`" to use java-7-openjdk-armhf. The "`openjdk-7-jre`" package may also need to be installed.

Ninja IDE Application.



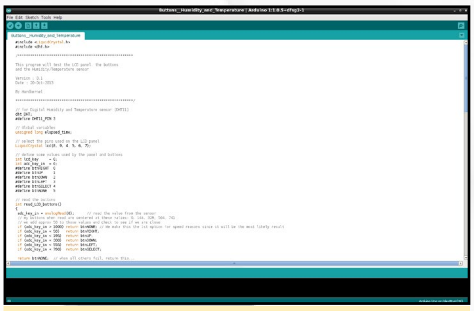


Netbeans IDE Application

```
netbeans_jdkhome="/usr/lib/jvm/
java-7-openjdk-armhf/"
```

Arduino

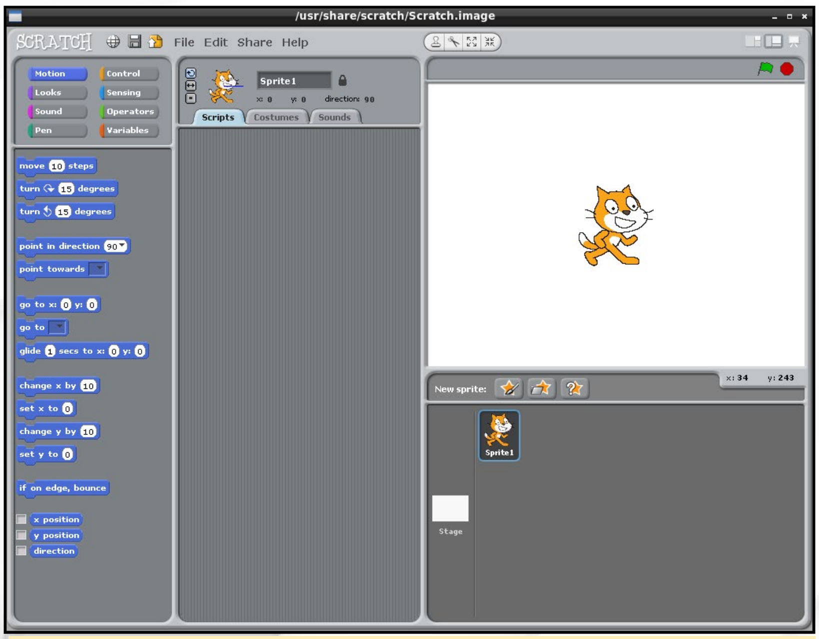
The Arduino IDE (`http://www.arduino.cc`) is the ideal development environment for home automation, robotics and control systems. It allows any of the Arduino products to be flashed with custom programs for use in interactive projects. It has its own programming language that is similar to C++. The Arduino IDE is especially suitable for ODROIDs, since it includes tools for writing, compiling and uploading programs for Hardkernel's I/O Shield and Oduino (Arduino Uno) peripherals.

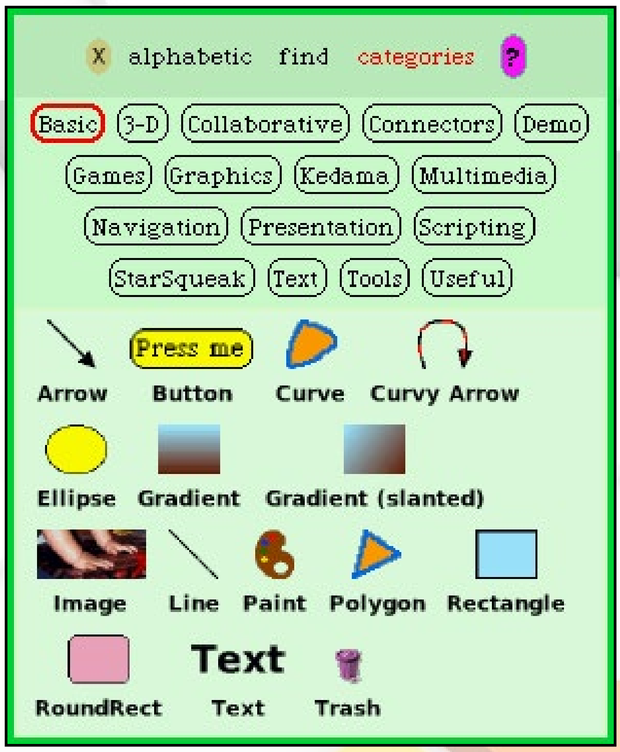


Arduino IDE Application

Scratch and Squeak

Scratch (`http://scratch.mit.edu`) is an educational programming language that teaches methodology and problem-solving approaches with an intuitive, easy-to-use interface. Aimed at making learning fun, Scratch can be used to create stories, games and animations that can then be shared with others. Squeak (`http://www.squeak.org`) is a customized version of Scratch that implements the Smalltalk language, which was one of the first object-oriented languages.





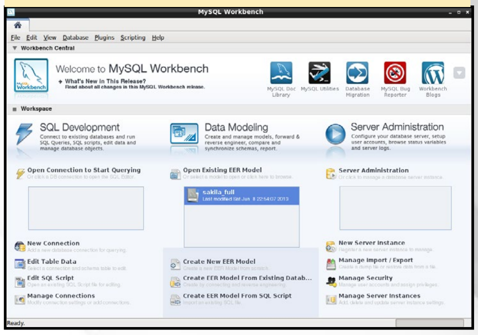
Scratch (above) and Squeak (right) are perhaps the most important tools available with the Code Monkey image. Why? Well, we always need to keep our young ODROIDians in mind, so we made sure that this distribution is as cool and educative as possible!

Game development

Code Monkey includes drivers for most joysticks and controllers, including Xbox 360, Wiimote and PS3. Use the "joytest" program, available in the Applications menu, to determine the serial port of the USB gamepad, which can then be accessed inside a loop to determine the controller's button and joystick movements. Refer to `http://bit.ly/1sAsmeP` for a complete guide on configuring and installing joysticks in Ubuntu.

Wiimotes are also compatible via the Bluetooth interface, and can be used as gyroscopic and infrared sensors. For more information on using the Wiimote with Linux, refer to page 8 of the June 2014 issue of ODROID Magazine.

MySQL Workbench



MySQL and PostgreSQL

Structured Query Language (SQL) is the software industry standard for reading and writing database information, and is well supported on the Linux platform. One of the more popular implementations, called MySQL, is pre-installed and may be managed by launching the "MySQL Workbench" application from the Applications menu. The MySQL root password is "odroid".

PostgreSQL is another popular implementation of SQL for the Linux platform, and is well known for its reliability. To install PostgreSQL on Code Monkey, please refer to the official Ubuntu guide at `http://bit.ly/1bsJQo4`.

Git and Subversion

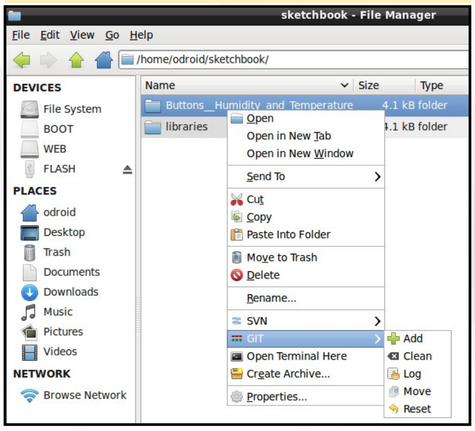
Both Git (`www.github.com`) and Subversion (`http://subversion.apache.org`) offer extensive code management and collaborative tools for managing large-scale projects. Hardkernel uses GitHub as its code repository (`http://bit.ly/ZX834P`), and all of their kernels and software sources may be downloaded from GitHub using the "git clone" Terminal command. For more information on accessing the Hardkernel repositories, visit `http://bit.ly/1wb9ity`.

Subversion is a similar software project management suite that allows the use of local repository servers instead of relying on an Internet cloud server to store code versions. Subversion functionality may be accessed by right-clicking on an empty folder using the Thunar File Explorer.

Apache Web Server and PHP

Apache 2 is the industry standard for web servers, although its popularity has been recently challenged by the more lightweight Nginx server. Websites are located at "`/var/www/html/<websiteName>/`", and

Git and Subversion



should also be added to the list of available websites at "`/etc/apache2/sites-available/`". To access the website, visit `http://127.0.0.1/<websiteName>`. Apache will automatically detect the language of the site depending on the filename extension used. For instance, index.html indicates that the page is HTML5-based, and index.php indicates that the web page is written in PHP.

PHP is a very well-known language that is easy to learn, and can be used to write powerful web applications. Because of its stability and wide-spread acceptance, some of the highest traffic websites on the Internet are written in PHP, including Facebook and Wikipedia. When developing with PHP, I recommend using the Netbeans or Bluefish development environments. To learn more about PHP, visit the official development site at http://bit.ly/1zeFFdp.

Tomcat Java Server

Tomcat (`http://bit.ly/1wzLbUz`), also maintained by the Apache Software Foundation, is another type of web server that handles Java-based applications. It also can be used to build solid web applications, and is often run in conjunction with Apache to query and save data via JSONP. To access Tomcat applications, visit http://127.0.0.1:8080/<websiteName> after copying the web application to the "/var/lib/tomcat7/webapps/ROOT/webSiteName" directory.

Samba

A Samba (SMB) share has been pre-configured at "`/home/odroid/Public`", which is accessible to both the root and odroid logins, with a default password of "odroid" for both users. To configure additional Samba directories or change the passwords, type the following in a Terminal window:

```
$ sudo system-config-samba
```

Vino and Remmina

Code Monkey includes Vino VNC Server which auto-starts on boot, allowing access to the desktop of the development machine remotely. Additional information on controlling the Vino server is available at `http://bit.ly/1wbbhOG`. To access other desktops from the Code Monkey development machine, use the Remmina application, which provides support for connecting to VNC and RDP servers.

Android Development

The ODROID family of single-board computers make ideal Android development machines, primarily because they can run both Linux and Android. This enables a developer to create Android apps using one ODROID running Code Monkey, then upload the app via Android Debug Bridge (ADB) to another ODROID running a rooted version of Android. To learn more about using ADB with Linux, visit the ODROID Wiki at `http://bit.ly/1u4L2uq`.

EASILY ROTATE YOUR SCREEN ON ANDROID

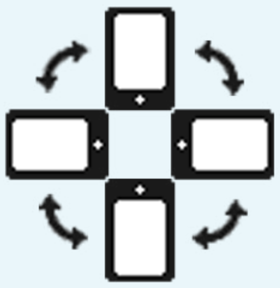
DEFEAT YOUR VIRTUAL ENEMIES WITHOUT ROTATING YOUR ENTIRE MONITOR

by Bruno Doiche

Every now and then we may want to play a fun Android game on our ODROID, but what do you do when the game requires you to use the screen in Android's portrait orientation?

iFighter 2: The Pacific 1942 is way more fun to play in portrait mode

To make life easier, just install the **App Screen Rotation Control**, or **Ultimate Rotation Control** from the Google Play store, and enjoy your vertical games!



BUILD AN ODROID-POWERED OFF-ROAD UNMANNED GROUND VEHICLE

PART 3: GPS NAVIGATION PROGRAMMING

by Christopher D. McMurrough

With this article, we conclude our series on building an Off-Road Unmanned Ground Vehicle (UGV) using the ODROID-XU platform by focusing on guiding the platform to a predefined GPS waypoint using navigation data provided by an external Android device. We will use the Ubuntu 12.04 Robotics Edition image for the ODROID-XU, which can be found on the Hardkernel forums at http://bit.ly/1vK6TWD.

In Part 1 and 2 of our series, we focused mostly on the hardware setup of our system. This included our mechanical chassis, power distribution, motor controllers, and mounting of our electronics. In Part 2, we discussed data acquisition and motor control using the Robotic Operating System (ROS) software. We developed ROS nodes for each of the necessary input and outputs of our system, and at the conclusion of our last article, we were ready to develop the control laws necessary to guide the platform to a target GPS coordinate. As before, we will continue to provide example code on the project repository at http://bit.ly/1jfykOU.

Waypoint navigation

In this demonstration, we will perform a simple navigation task that will continuously estimate the robot's current position and orientation, and compute the necessary motion control commands to move the platform toward a goal location. The "goal" will be defined as a constant in our navigation code. For testing purposes, the goal is the GPS location of a marked point in our test field, but could easily be extended to include other points or be configured dynamically at run-time. As our test area is an open field, we will not perform any obstacle detection or avoidance during these tests.

Our simple navigation strategy consists of two parts: first, the linear distance between our current and target GPS locations will be computed and used to calculate a maximum forward speed for the platform. This is intended to decelerate the platform as we approach the target such that we avoid overshooting our goal. Second, we compute the angle between the current robot's heading (provided by the compass subsystem on the Android device) and the direction of the goal location relative to the robot's location. This angle will be used to compute the differential wheel speeds necessary to correct the robot's heading as it attempts to move toward the goal.

In the previous article, we discussed motor control using the Teensy microcontroller and its ROS node. The ROS motor controller node expects a standard "Twist" type message, which encapsulates linear velocities (Vx, Vy, Vz) and rotation speeds about the robot axes (Rx Ry, Rz). Since our vehicle may only move along the ground, and can not slide left-to-right, we only need to consider a single forward/backward speed, Vx.

Similarly, since our robot cannot "roll" to either side or "pitch" up and down, we will only consider a single rotational speed, Rz (we will assume the x axis points forward, and the z axis to point up, thus we choose Vx and Rz). Our motor controller ROS node has already been designed to convert Vx and Rz values into individual left/right wheel speeds, so we will create a ROS node called "navigation" that will subscribe to the GPS and compass messages published by the "android_sensors_driver" node and publish the necessary "Twist" messages that the motor controller node expects.

Computations

Our navigation node will compute the forward speed, Vx, using the distance between the current and target GPS locations. Using the trigonometric solution presented at http://bit.ly/1FzriC6, we can compute the distance using:

```
dlat = lat2 - lat1;
dlon = lon2 - lon1;
distance = sqrt(dlat*dlat +
dlon*dlon);
```

Once we have the distance, we can compute the angle using:

```
y = sin(lon2-lon1)*cos(lat2);
x = cos(lat1)*sin(lat2)-
sin(lat1)*cos(lat2)*cos(lon2-
lon1);
if(y > 0)
{
      if (x > 0)
              angle =
arctan(y/x);
      else
              angle = 180 -
arctan(-y/x);
}
else
{
      if (x > 0)
              angle = -arctan(-
y/x);
      else
              angle =
arctan(y/x)-180;
}
```

Once we have computed the distance (meters) and angle (degrees), we can compute the Vx and Rz values that we need for our "Twist" message. As previously mentioned, we want our Vx value to slow down the platform when it approaches some distance to the target, otherwise, we will run at full speed. We will choose our approach distance arbitrarily to be 20 meters. We will set the limits of Vx to be (-1,1), where -1 is full speed in reverse and +1 is full speed forward. Our Vx value is then computed:

```
if (distance > 20.0)
{
      Vx = 1.0;
}
else
{
      Vx = distance/20.0;
}
```

This will cause the robot's forward speed to decrease as it moves closer to the target, once it is within the 20 meter goal radius. We can then compute the rotation speed Rz using:

```
if (angle < 180.0 && angle > 0)
{
      Rz = -angle/180.0;
}
else
{
      Rz = angle/180.0;
}
```

In this convention, a positive Rz will cause the robot to spin toward the right, while a negative will cause a spin toward the left. The rotation speed will decrease as the robot corrects its bearing angle similar to the way the forward speed will decrease as the robot approaches the goal. When these functions are combined when the robot is running, the platform should spin toward the goal location and apply small corrections as it makes forward progress. This is exactly the behavior we want when ignoring obstacles for a differentially steered robot. Once we combine the Vx and Rz functions into our navigation ROS node, we are ready for testing!

Testing

Our system was tested in an open field with no significant obstacles such as trees and large holes. Before we attempted a run, we selected a goal location in the middle of the field and measured its position with the Android tablet. This value was then added to our navigation node as the goal position. After we input this value, the robot's only purpose in life is to drive toward the goal location at all costs! Once the robot senses that its location is close to the goal, the speed decreases to the point where the robot no longer moves forward. This gave us a chance to flip the motor power switch to disable the robot when it reaches the goal.

One noticeable drawback of our relatively simple control laws is that they do not handle GPS inaccuracy very well. Consumer grade GPS data, such as that provided by the Android tablet, is only accurate to within 10 meters or so. This inaccuracy increases in the presence of trees, clouds, etc. When the robot is far away from the target, this is virtually unnoticeable.

The problem becomes obvious when the robot moves close to the target, since the GPS fix can "jump" by 10 meters or more between measurements. The result is that the robot can suddenly think that the goal position is behind, causing a sudden rotation similar to the initial bearing fix. In fact, once the robot is within a distance of 10 meters, its motion becomes quite chaotic. This could be addressed with some more advanced signal filtering or a high performance GPS, but getting within 10 meters was good enough for our initial tests!

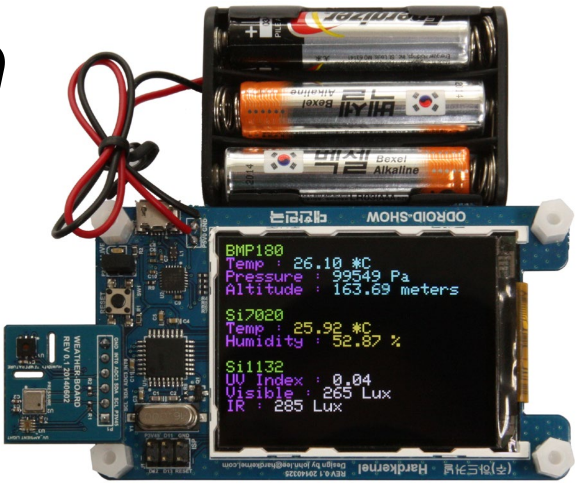
Another drawback to using the Android tablet as the only navigational sensor is that the compass data is susceptible to corruption by external magnetic fields. Since our platform consists of 6 magnetic motors, it is crucial that we mount the table as far away as possible. Sources of iron that may be present in the ground can cause interference as well, so we place the table on a pole that extends roughly 0.5 meters from the top chassis plate. While this does not completely eliminate magnetic interference, it seems to work well for our initial field testing.

Conclusion

In this series, we presented an overview of an Off-Road Unmanned Ground Vehicle (UGV) using the ODROID-XU platform. We have covered the mechanical, electrical, and software aspects of the system in order to provide aspiring ODROID robot enthusiasts with ideas on how to approach their own projects. As a continuing project, we will focus on obstacle avoidance using the RGB-D camera and the processing power of the ODROID-XU.

BUILDING A WEATHER BOARD APPLICATION

CREATE A MINIATURE WEATHER DATA COLLECTION SYSTEM



by Justin Lee

This article presents an example of an application that receives input from Hardkernel's Weather Board (http://bit.ly/1wtPdgP), and displays the data in realtime on an ODROID-VU touchscreen (http://bit.ly/UmZEod) using an Arduino-based ODROID-SHOW (http://bit.ly/1wyo5MZ) as an embedded controller.

For this example, an ODROID-U3 with Ubuntu 14.04 installed was used as the development host machine. The Android app may be programmed using Eclipse, and the ODROID-SHOW firmware can be updated using the Arduino IDE. To install Eclipse, please refer to http://bit.ly/1pMAAAJ. To install the Arduino IDE, type the following into a Terminal window:

```
$ sudo apt-get update && sudo
apt-get install arduino arduino-
core
$ sudo arduino
```

All of the software described below is available for free download from the Hardkernel Github repository at http://bit.ly/1snZCG0.

Download source

For reference, the original serial port source upon which this weather application is based is available from Google at http://bit.ly/1zubmQX. The full source code for the ODROID-SHOW version may be downloaded by typing the following:

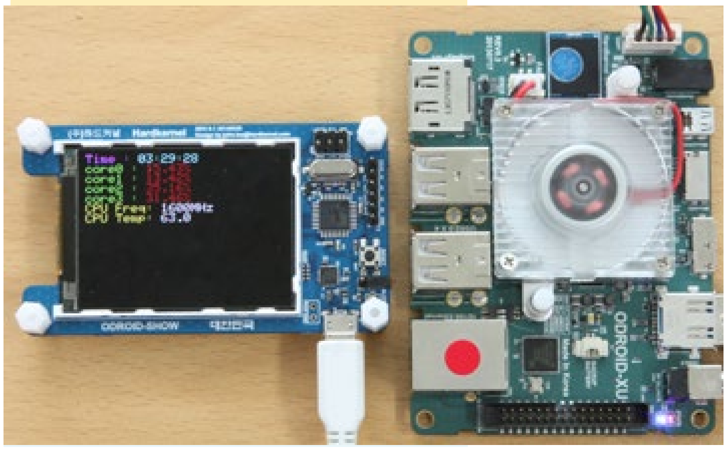
```
$ git clone git@github.
com:codewalkerster/Weather.git
```

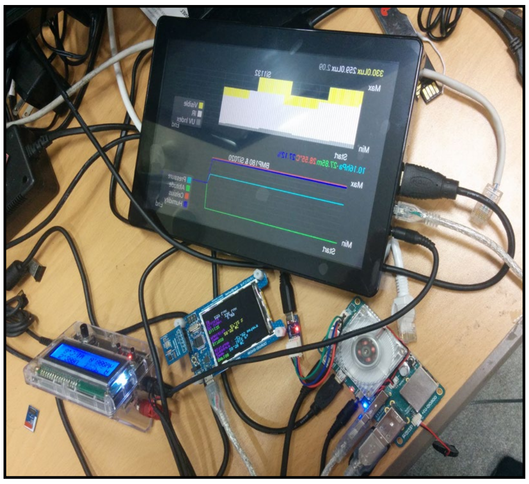
Create a new Android Application project and call it "Weather". Copy the "jni" directory and the file "SerialPort.java" from the downloaded source code into the new Weather project.

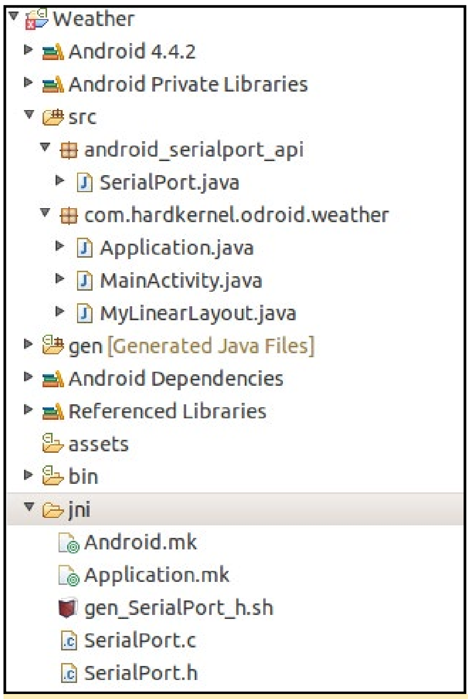
Next, copy the Application.java file into the project, then modify the port to "/dev/ttyUSB0" and the baud rate to 500000.

```
public SerialPort getSerialPort()
throws SecurityException, IOEx-
ception, InvalidParameterExcep-
tion {
      if (mSerialPort == null) {
      /* Open the serial port */
              mSerialPort = new
SerialPort(new File("/dev/tty-
USB0"), 500000, 0);
      }
      return mSerialPort;
}
```

Connecting the host computer to the ODROID-SHOW via USB







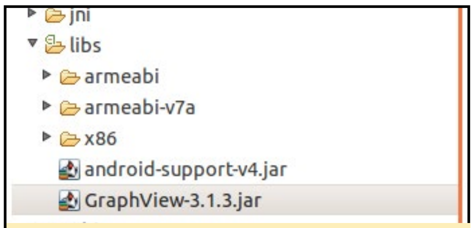
```
▼ Weather
  ▶ Android 4.4.2
  ▶ Android Private Libraries
  ▼ src
    ▼ android_serialport_api
      ▶ SerialPort.java
    ▼ com.hardkernel.odroid.weather
      ▶ Application.java
      ▶ MainActivity.java
      ▶ MyLinearLayout.java
  ▶ gen [Generated Java Files]
  ▶ Android Dependencies
  ▶ Referenced Libraries
    assets
  ▶ bin
  ▼ jni
      Android.mk
      Application.mk
      gen_SerialPort_h.sh
      SerialPort.c
      SerialPort.h
```

Files to be copied into the weather project

Download the GraphView-3.1.3.jar file from http://bit.ly/1wyuibY and copy it into the "libs" folder of your project.

Then, create the serial input stream:

```
mSerialPort = mApplication.getSe-
rialPort();
mInputStream = mSerialPort.getIn-
putStream();
```



```
  ▶ jni
  ▼ libs
    ▶ armeabi
    ▶ armeabi-v7a
    ▶ x86
      android-support-v4.jar
      GraphView-3.1.3.jar
```

Copying the GraphView .jar file

The raw data is in this format:

```
ESCw0[bmp180 Temperature] +
ESCw1[bmp180 Pressure] +
ESCw2[bmp180 Altitude] +
ESCw3[si7020 Temperature] +
ESCw4[si7020 Humidity] +
ESCw5[si1132 UV Index] +
ESCw6[si1132 Visible] +
ESCw7[si1132 IR]

public boolean updateData(byte[]
buffer) throws IOException {
        ...
        mInputStream.read(buffer,
0, 1);
        if (buffer[0] == 'w') {
            int i = 0;
            while (buffer[0] !=
0x1b) {
                mInputStream.
read(buffer, 0, 1);
            ...
            switch (index) {
            …
            case '1':   //bmp180
Pressure
                String str = new
String(buf).split("\0")[0];

mPressureData[mGraphX] = new
GraphViewData(mGraphX, Double.
parseDouble(str) / 100);
```

After the GraphViewData[] array has been filled with new data, call the GraphViewSeries::resetData() function.

```
mPressureSeries.
resetData(mPressureData);
mAltitudeSeries.
resetData(mAltitudeData);
mTemperatureSeries.
resetData(mTemperatueData);
mHumiditySeries.
resetData(mHumidityData);
mUVIndexSeries.
resetData(mUVIndexData);
mVisibleSeries.
resetData(mVisibleData);
mIRSeries.resetData(mIRData);
```

ODROID-SHOW Firmware

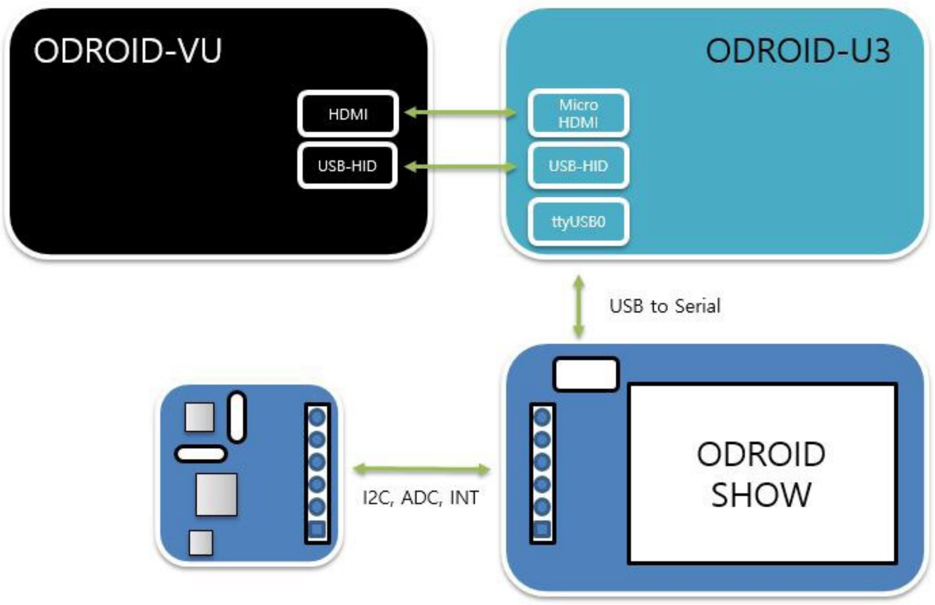
Download the ODROID-SHOW source code from Github, then add the libraries after installing the Arduino IDE. Please refer to our wiki for details at http://bit.ly/ZKD7UM.

```
$ sudo apt-get install git
$ git clone https://github.com/
hardkernel/ODROID-SHOW
```

After connecting the jumper, compile the project and upload it. Once running, the ODROID-SHOW will display the sensor values, and also send the values through the serial port in the background.

To fix the time latency of sensor values for serial transfer, edit the "weather_

Block Diagram for ODROID-VU + ODROID-U3 + ODROID-SHOW + Weather Board



board.ino" file. If you change the sensing time, make sure to also adjust the value of Timer1.initialize.

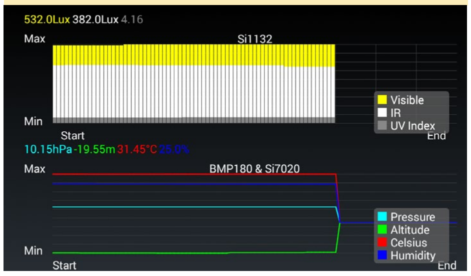
```
# File: ODROID-SHOW/weather_
board/weather_board.ino

void setup() {
…
…

// Timer one setting
Timer1.initialize(200000); //
200ms
Timer1.
attachInterrupt(timerCallback);
}
```

Further Reading

ODROID-SHOW Wiki
http://bit.ly/1toe7Pl

Weather Board Wiki
http://bit.ly/ZKD7UM

The output of the Weather Board application on an ODROID-VU

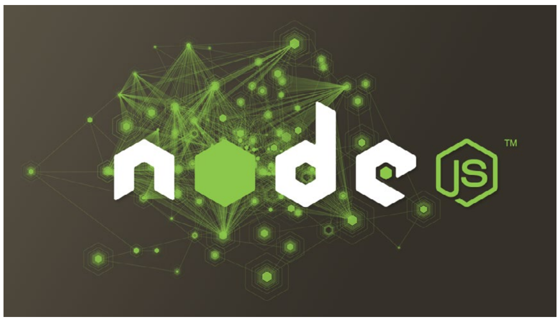


A WeatherBoard can help you predict humidity and prevent bad hair days

HOW TO INSTALL NODE.JS

A MODERN JAVASCRIPT-BASED WEB APPLICATION PLATFORM

edited by Venkat Bommakanti



Node.js is a lightweight and efficient platform, well-suited for data-intensive real-time applications. It's based on Chrome's JavaScript runtime language, and offers an event-driven, non-blocking I/O model for web applications. This article describes the installation process for Node.JS on the ODROID-U3.

Requirements

1. Any ODROID board, with an appropriate power adapter.
2. A bootable 8+ GB MicroSD card or eMMC module containing the latest U3 Lubuntu image available from the Hardkernel website.
3. Optional SSH access to the U3 via utilities like PuTTY (MS Windows 7+) or Terminal (Mac, Linux).

Update the system

To begin, install the essential build tools and environment using the single-line command apt-get. All commands shown in the example below should be typed onto a single line.

```
$ sudo apt-get install python \
g++ make auto-apt checkinstall \
fakeroot build-essential
```

Fetch the latest source code

As of November 2014, the latest version of node.js is 0.10.32, and the steps listed here may need to be adjusted for newer versions. To install an older version of node.js, simply type the following, then skip to the "Create Test Sample" section below:

```
sudo apt-get install nodejs
```

To install the latest version of node.js, create the following sub-directories in the home folder and navigate to the src directory:

```
$ mkdir nodejs && cd nodejs
$ mkdir src && cd src
```

Get the latest source tar-ball and unpack it, using the following commands:

```
$ wget -N http://nodejs.org/dist/
node-latest.tar.gz
$ tar xzvf node-latest.tar.gz
$ cd node-v0.10.32
```

Build the package

```
$ sudo auto-apt run ./configure \
--without-snapshot
```

```
Entering auto-apt mode: ./configure
--without-snapshot
Exit the command to leave auto-
apt mode.
{
  'target_defaults': {
   'cflags': [],
   'default_configuration':
     'Release',
   'defines': [],
   'include_dirs': [],
   'libraries': []
  },
  'variables': {
  'arm_fpu': 'vfpv3',
  'arm_neon': 0,
  'armv7': 1,
  'clang': 0,
  'gcc_version': 48,
  'host_arch': 'arm',
  'node_install_npm': 'true',
  'node_prefix': '',
  'node_shared_cares': 'false',
  'node_shared_http_parser':
'false',
  'node_shared_libuv': 'false',
  'node_shared_openssl': 'false',
  'node_shared_v8': 'false',
  'node_shared_zlib': 'false',
  'node_tag': '',
  'node_unsafe_optimizations': 0,
  'node_use_dtrace': 'false',
  'node_use_etw': 'false',
  'node_use_openssl': 'true',
  'node_use_perfctr': 'false',
  'node_use_systemtap': 'false',
  'openssl_no_asm': 0,
  'python': '/usr/bin/python',
  'target_arch': 'arm',
  'v8_enable_gdbjit': 0,
  'v8_no_strict_aliasing': 1,
  'v8_use_arm_eabi_hardfloat':
'true',
  'v8_use_snapshot': 'false',
  'want_separate_host_toolset':
0}}
creating  ./config.gypi
creating  ./config.mk

# the command below should be
typed on a single line
```

```
$ sudo fakeroot checkinstall -y
--install=no \
--pkgversion $(echo $(pwd) | sed
-n -re's/.+node-v(.+)$/\1/p')
make -j$(($(nproc)+1)) install

[...]

Done. The new package has
been saved to node-v0.10.32/
node_0.10.32-1_armhf.deb
 You can now install it in your
system anytime using:

dpkg -i node_0.10.29-1_armhf.deb
```

Install the package

First, check to ensure that the debian package was created properly by typing:

```
$ ls -ltr
[...]
-rw-r--r--  1 root
root  3688430 Jul 30 15:03
node_0.10.29-1_armhf.deb

$ sudo dpkg -i \
node_0.10.29-1_armhf.deb
```

If all is well so far, you can skip the rest of this section. In case of problems with the above process, the .deb file can be built manually by typing in the following commands:

```
$ sudo apt-get install python g++
make checkinstall fakeroot
$ src=$(mktemp -d) && cd $src
$ wget -N http://nodejs.org/dist/
node-latest.tar.gz
$ tar xzvf node-latest.tar.gz &&
cd node-v*
$ ./configure
# the command below should be
typed on a single line
$ sudo fakeroot checkinstall -y
--install=no \\
--pkgversion $(echo $(pwd) | \
sed -n -re's/.+node-v(.+)$/\1/p')
make -j$(($(nproc)+1)) install
$ sudo dpkg -i node_*
```

Then, verify the manual installation using the commands:

```
$ which node
/usr/local/bin/node
$ node --version
v0.10.32
```

Create a test sample

Create a subdirectory and the sample JavaScript file to be run:

```
$ mkdir sample
$ cd sample/
$ touch hello-world.js
$ chmod +x hello-world.js
$ medit hello-world.js
```

Add the following file contents to the sample JavaScript file:

```
var http = require('http');
http.createServer(function (req,
res) {
  res.writeHead(200, {'Content-
Type': 'text/plain'});
  res.end('Hello ODROID
World\n');
}).listen(8090,
'your-u3-ip-address');
console.log('Server run-
ning at http://your-u3-ip-ad-
dress:8090/');
```

Note the use of port 8090 in this case.

Test the sample

In a terminal window, start Node.js using the above JavaScript file:

```
$ node hello-world.js
```

To test that the installation is working properly, use another locally networked device such as a PC, tablet, or ODROID-U3, and direct a browser to http://<your-u3-ip-address>:8090.

For additional information or questions, please visit the Node.js website at http://nodejs.org.

OPEN MEDIA VAULT

OPEN SOURCE NETWORK ATTACHED STORAGE FOR DEBIAN GNU/LINUX

by Venkat Bommakanti

OpenMediaVault
The open network attached storage solution
The open network attached storage solution
Built to give your data a home

When creating a Network Attached Storage (NAS) solution, OpenMediaVault (OpenMediaVault) is a popular choice for small and home offices scenarios (SOHO). In addition to allowing shared files to be accessed by users on a common network, OpenMediaVault offers services and plugins such as:

- BitTorrent
- Secure Shell (SSH)
- File Transfer Protocol (T/FTP)
- Network File System (NFS)
- Samba (SMB)
- Common Internet File System (CIFS)
- Lightweight Directory Access Protocol (LDAP)
- Uninterruptible Power Supply (UPS)
- Digital Audio Access Protocol (DAAP) media server
- Remote synchronization (Rsync)
- Web-based administration utility

This article describes the installation process of OpenMediaVault on an ODROID-U3, and the steps may also be applied to the more capable ODROID-XU3.

Requirements

1. An ODROID-U3 board, with an appropriate power adapter
2. A Class 10 MicroSD (with an SD card USB adapter) installed with the latest U3-specific Debian Wheezy desktop image
3. A network where the device has access to the Internet and the ODROID forums
4. SSH access to the U3 via utilities like PuTTY (MS Windows 7+) or Terminal (Mac, Linux)

Install nginx

A web server is required to host the web-based GUI of OpenMediaVault. Install a basic version of nginx using the following commands. A more elaborate nginx installation may be performed using the steps outlined in the article on page 25 of the August 2014 issue of ODROID Magazine.

```
$ sudo apt-get install nginx
$ sudo service nginx stop
$ sudo service nginx start
```

Available images

Visit http://bit.ly/1zwjhxc to see a list of the available OpenMediaVault images:

```
Name                    Last modified
omnius/                 29-Apr-2012
                        23:08
ix/                     15-May-2012
                        21:04
omnius-proposed/        09-Sep-2012
                        00:45
fedaykin/               25-Feb-2013
                        21:05
sardaukar/              20-May-2013
                        16:34
kralizec/               11-Jan-2014
                        21:32
sardaukar-proposed/     16-May-2014
                        23:25
kralizec-proposed/      16-May-2014
                        23:25
```

For this example, we will use the latest kralizec image, which can be selected for installation by typing the following:

```
$ cd /etc/apt/sources.list.d/
$ sudo touch openmediavault.list
$ sudo medit openmediavault.list
```

Add the following line in the openmediavault.list file and save it:

```
deb http://packages.openmedi-
avault.org/public kralizec main
```

Install packages

Install the packages using the following commands, ignoring any errors and warnings:

```
$ sudo apt-get update
...
Get:1 http://packages.openmedi-
```

```
avault.org kralizec Release.gpg
[181 B]
Get:2 http://packages.openmedia-
vault.org kralizec Release [9,696
B]
Ign http://packages.openmedi-
avault.org kralizec Release
...
Get:3 http://packages.openmedi-
avault.org kralizec/main armhf
Packages [6,678 B]
Ign http://packages.openmedi-
avault.org kralizec/main Transla-
tion-en_GB
Ign http://packages.openmedi-
avault.org kralizec/main Transla-
tion-en
...
Reading package lists...  Done
W: GPG error: http://packages.
openmediavault.org kralizec Re-
lease: The following signatures
couldn't be verified because the
public key is not available: NO_
PUBKEY 7E7A6C592EF35D13
```

Install keyring

Install the OpenMediaVault keyring using the following command:

```
$ sudo apt-get install openmedia-
vault-keyring postfix
```

The installation process will ask for additional information:

Mail Server: Local Only
Name: <your-u3's-hostname>
Mail-recipient: odroid
Synchronous mail updates: No

Run the following command to update the installation:

```
$ sudo apt-get update
...
Get:1 http://packages.openmedi-
avault.org kralizec Release.gpg
[181 B]
...
Hit http://packages.openmedi-
avault.org kralizec Release
...
Hit http://packages.openmedi-
avault.org kralizec/main armhf
Packages
...
Ign http://packages.openmedi-
avault.org kralizec/main Transla-
tion-en_GB
Ign http://packages.openmedi-
avault.org kralizec/main Transla-
tion-en
...
```

Install OpenMediaVault

Install OpenMediaVault using the command:

```
$ sudo apt-get install openmedi-
avault
```

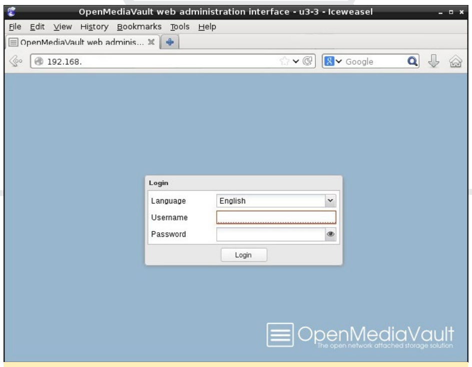
The installation process will prompt for additional information:

```
Run Samba as: Daemon
Install beep as: Usable for all
(users, not just root)
MD arrays: All
Monthly mdam checks: Yes
MD monitoring daemon: Yes
MD events notification: odroid
resolve.conf: No
Reboot: No
Run ProFTPD as: Standalone
Quota reminders: Yes
Over quota notification:
odroid@<your-u3's-hostname>
Phone #: <blank>
Watchdog module: None
Smart watchdog @bootup: Yes
Restart watchdog on updates: No
```

Initialize OpenMediaVault, then reboot using the following commands:

```
$ sudo OpenMediaVault-initsystem
$ sudo reboot
```

The installation may be updated with the Update Manager page.


OpenMediaVault web administration interface - u3-3 - Iceweasel
Login
Language
English
Username
Password
Login
OpenMediaVault

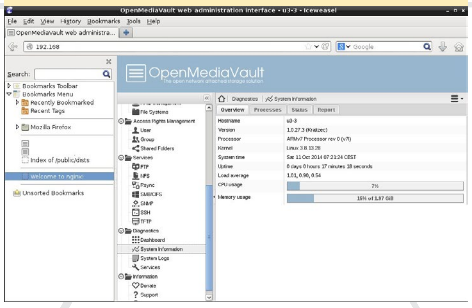
OpenMediaVault login page

Access OpenMediaVault

After the reboot has completed, access the OpenMediaVault installation by pointing the Iceweasel browser to http://<your-u3-ip-address>, which should show the OpenMediaVault login page. The default user id is "admin" with a password of "openmediavault".

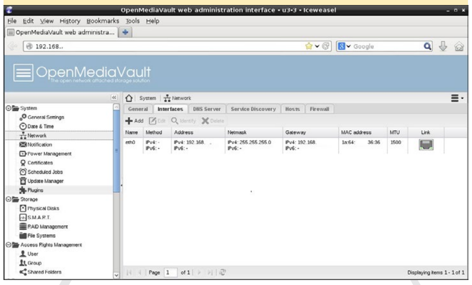
By default, the standard port 80 is used, but OpenMediaVault may also be configured to use a different port if another web server is already running on the machine. Upon a successful login, the System Information page is displayed, and the networking functionality may be verified by navigating to the Network Interfaces page, as shown in the screenshots below.

The OpenMediaVault system information web administration page



Hostname	u3-3
Version	1.0.27.3 (Kralizec)
Processor	ARMv7 Processor rev 0 (v7l)
Kernel	Linux 3.8.13.28
System time	Sat 11 Oct 2014 07:21:24 CEST
Uptime	0 days 0 hours 17 minutes 18 seconds
Load average	1.01, 0.90, 0.54
CPU usage	7%
Memory usage	15% of 1.97 GiB

The OpenMediaVault system information network interfaces page



Name	Method	Address	Netmask	Gateway	MAC address	MTU	Link
eth0	IPv4: - IPv6: -	IPv4: 192.168. IPv6: -	IPv4: 255.255.255.0 IPv6: -	IPv4: 192.168. IPv6: -	1a:64: 36:36	1500	

MEET AN ODROIDIAN

ALEXEY GUSEYNOV (@KIBERGUS): SOFTWARE ENGINEER EXTRAORDINAIRE

edited by Rob Roy

Please tell us a little about yourself.

My name is Alexey Guseynov. I'm a software engineer with the Russian search giant Yandex, where I work with the Yandex.Maps team. I've been using Linux for more than 10 years, so when I heard about Hardkernel's ODROID boards and realized that they are nearly as powerful as my old desktop, I couldn't avoid buying a few. From that moment, the computers in my home have outnumbered the human beings by an unattainable margin.

How did you get started with computers?

I should thank my grandfather, who insisted that his grandsons should have computers in the mid-1990s. For some time, I used it for games, but I was always interested in how things work, so my brother and I permanently disassembled many devices. As a result, I'd obtained some basic programming skills by 8th grade. After that, I transferred from regular school to a school of informational technology, where the teachers have given me a really good knowledge basis.

What is your favorite ODROID?

My favorite ODROID is still the U2. I understand that the U3 has a much more practical layout, is smaller, includes GPIOs, and is also less expensive. And, the XU series models are much more powerful. However, the U2 has a stylish, solid and complete look. Unlike all other boards I've seen, the bare PCB of the U2 gives the impression of innovative design elements instead of a "why waste money on a case" approach. I really like when pure technical design decisions result in things being pretty.

What types of contributions have you made to the XBMC development efforts?

The story is quite simple. My satellite receiver is located in a wardrobe in a back room. But I wanted to watch it on a TV which is on the opposite side of my apartment. While efforts of other developers were concentrated on playing progressive video, I always stuck with the fact that interlaced video needed some special processing. I was the guy who always groaned that MFC still does not work well.

There also were times when a bug in the Mali drivers prevented XBMC from playing smoothly, but it was believed that the root cause was in the MFC. At that time, @OverSun hadn't yet earned his great reputation as the MFC conjurer. So, I created stress tests and confirmed that @OverSun's code was working great, and that the problem was in the video subsystem.

The funny thing is that I never watch TV programs!

What hobbies and interests do you have apart from computers?

I don't have any particular hobby, but I do enjoy trying new activities. I have a bike which I ride all year round, which is even more fun in winter. When the snow is deep, I like to ski, both cross-country and downhill, and my wife is especially fond of skiing as well. I've also tried parachute jumping. After you jump, you're instructed to count "341", "342", "343", then pull a release ring.

I was astonished that I could forget how to count in three digit numbers and at the same time think very fast to find a solution! This summer, ten years after my father taught me how to dive with scuba gear, I passed my PADI (Professional Association of Diving Instructors) certification program.

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

I've been involved with Open Street

Alexy's uses his trusty U2 as an audio center. He has networked it for use as a universal sound card

Maps (OSM) and I've spent a lot of time walking around the areas near my house. But now, OSM data is so detailed, that it is hard to find an unmapped place. So I switched my focus to electronics, and as a result, the lights in my apartment are controlled by an Arduino board.

Two months ago I received a letter from a crowdfunding company ("Don't you want to buy a kit with a high temperature superconductor?") and I couldn't resist. So now, I'm building a type of Rube Goldberg machine which will use a levitating carriage as one of its steps.

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

I prefer silent devices, so I vote for a bigger and more effective heatsink on ODROIDs. I understand that it would increase the cost, but I nevertheless would spend money on a fan-less cooler.

What advice do you have for someone wanting to get started with programming?

During my first programming lesson, my teacher suggested to play a game called "Crazy Artist". You urgently need a very big drawing and your only option is to phone your friend (a crazy artist with an unbalanced mind and amnesia), and ask him to draw it. You always start with: "Hello, please read a drawing manual", then give him simple orders. If you say something wrong, he curses, hangs up the phone and forgets everything. It was fun and interesting.

We didn't know about loops, if statements, variables, but we started writing programs. And we wanted to know how to blink the lights on a car that we'd drawn. So we wanted to know about loops and other programming techniques.

My advice is to find an interesting problem that you want to solve, then learn the skills that you need to know to complete the task. This way, you will have a much greater motivation because you understand how to use the things you learn.

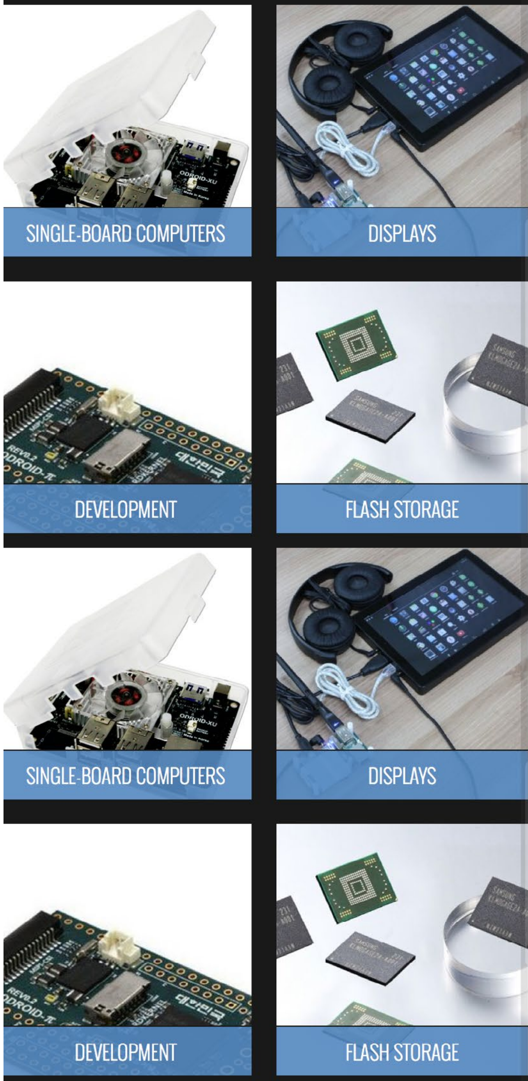
Alexey and his wife in the Mallorcian mountains

ADVERTISEMENT

OFFICIAL US DISTRIBUTOR OF HARDKERNEL PRODUCTS

ameriDroid

High-Performance Embedded Computers



Big excitement, small packages

Thrill your inner geek

ODROIDS ARE NOW AVAILABLE IN THE UNITED STATES

WWW.AMERIDROID.COM

AFFORDABLE SHIPPING