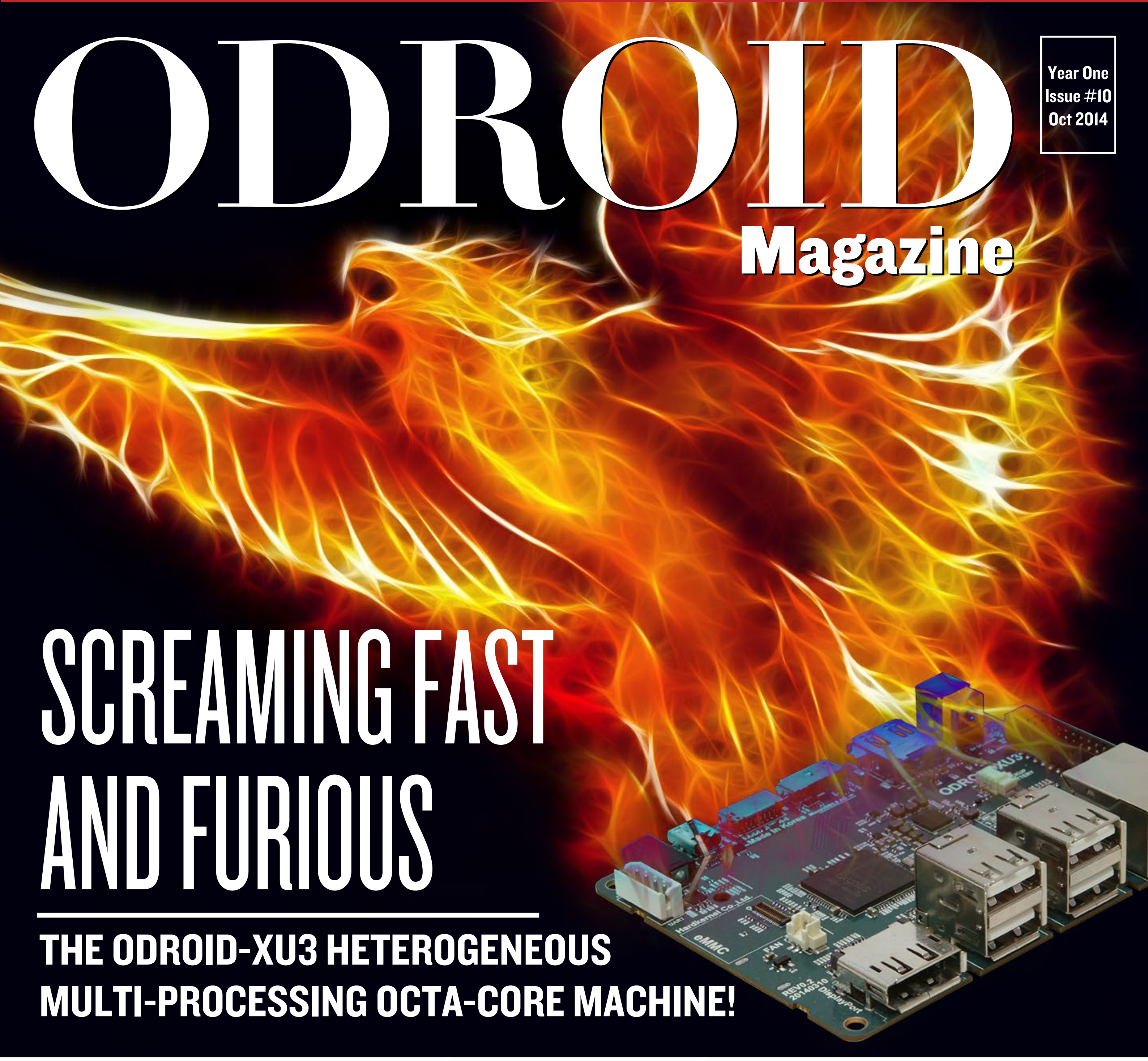
I2C communication between U3 and Arduino • Linux PSP emulation

# ODROID Magazine

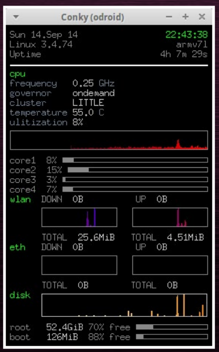
Year One
Issue #10
Oct 2014



## SCREAMING FAST AND FURIOUS

THE ODROID-XU3 HETEROGENEOUS MULTI-PROCESSING OCTA-CORE MACHINE!

USING THE CONKY HARMATTAN PERFORMANCE WEATHER MONITOR

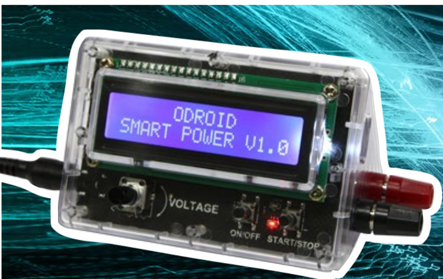


```
Sun 14.Sep 14                 22:43:38
Linux 3.4.74                    armv7l
Uptime                      4h 7m 29s
cpu
frequency   0.25 GHz
governor    ondemand
cluster     LITTLE
temperature 55.0 C
ulitization 8%
core1  8%
core2  15%
core3  3%
core4  7%
wlan   DOWN  0B           UP  0B
       TOTAL  25.6MiB    TOTAL  4.51MiB
eth    DOWN  0B           UP  0B
       TOTAL  0B          TOTAL  0B
disk
root   52.4GiB 70% free
boot   126MiB  88% free
```

• OS SPOTLIGHT: QUIET GIANT

• ANDROID DEVELOPMENT

CONFIGURING XBOX 360 CONTROLLERS ON RETROARCH



ODROID SMART POWER USE AND PROTOCOL ANALYSIS

# *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**

Elektronik, Technik und Sonderposten zu Superpreisen!

Sie sind hier: Startseite > Suche

Suchergebnis Ihre Suche nach "odroid" ergab 18 Treffer

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

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

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

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

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

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

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

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

ODROID-Linux
Supersch den OD Linux- berei
39,
Bes

Ihre Vorteile
✓ Innovative Produkte
✓ Optimales Preis-/Leistungsverhältnis
✓ Kompetente Beratung
✓ Schnelle Lieferung

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

Sicher Einkaufen

The Hardkernel team took a trip this month to Santa Clara, California for the ARM TechCon convention. At the booth were several demos of the current hardware, including the XU3, U3 and VU. Mauro and Justin created an amazing version of Ubuntu 14.04 running an Android virtual machine via KVM. You can see a picture of it in the ARM TechCon article on page 37.

We also had a fun contest going on the exhibition floor to see if anyone could beat us at Angry Birds. One participant walked away with a complete U3 kit for scoring 3 stars with a single bird!

Thanks to everyone who stopped by the booth to share their love of ODROIDs. Much of the show was focused on wearables such as smart watches and micro-controllers. Hardkernel's powerful combination of hardware innovation and free software support (including Ubuntu 14.04 and KitKat 4.4.4), along with a commitment to open-source whenever possible, make ODROIDs unique.

In this issue, we feature software tutorials for several of the Hardkernel products, including the ODUINO Arduino and the Smart Power supply. Nanik also continues his Android Development series with an article on building your first custom Android application, a TicTacToe game.

We also have a great comparison between PPSSPP (a PlayStation Portable emulator) running on both Android and Linux, step-by-step instructions on getting your Xbox 360 wireless controllers working with the multi-console emulator RetroArch, and tips on getting your favorite MSX games running again! Jussi once again finds a way to combine the ODROID with the great outdoors, and shows how to set up the Conky Harmattan desktop monitor in Ubuntu for displaying the local weather forecast.

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

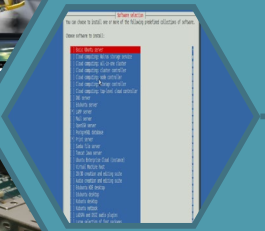
Secured his computing necromantic skills after bringing a fiber optics switch back to life, getting his Macintosh back from death, getting a PS3 back from death, getting his fiancee T400 back from death (that was a old style dd data transplant), and managing how to handle the cold innards of his steady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



ODROID
SMART POWER V1.0
VOLTAGE

ARM
2014 TechCon

# INTER-INTEGRATED CIRCUIT COMMUNICATION (I2C)

## ESTABLISHING A CONNECTION BETWEEN U3 AND ARDUINO

by Bennyamin Bergelson

This article describes a way to establish an I2C communication between an ODROID-U3 and an Arduino module. It shows what should be done at both the hardware and software level in order to successfully exchange data. I will also briefly compare the most popular communication protocols that exist today, and explain what data types can be transferred between the two modules.

## Communication Protocols

Several communication protocols exist that allow different electronic modules to communicate with each other. I will not go over all of them, but will concentrate on the UART, SPI and I2C protocols:

## UART

- suited for communications between only two devices (one bus per two devices)

- slow transfer speeds (9600 - 115200 bits per second, sometimes up to 230400 bps)

- the devices must "speak" at the same speed (baud rate)

- only one device should transmit, otherwise the devices receive gibberish

## SPI

- allows multiple "slave" devices.
- requires separate SS line for each slave device.
- operates at extremely high speeds (millions of bytes per second).
- master controls all communications (slaves can't talk directly to each other).
- communications must be well-defined in advance, and you can't send random amounts of data whenever you want

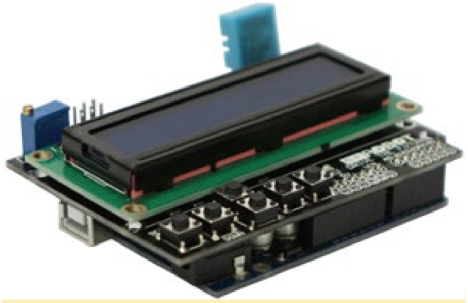
## I2C

- allows multiple "slave" devices, with support for up to 1008 slave devices.
- supports a multimaster system
- devices can communicate at 100kHz or 400kHz.
- requires only two wires

Because I need all my modules to communicate with each other using one mutual bus without adding extra wiring when adding new devices to the system, I decided to go with I2C. The following two sections will explain how to establish bidirectional I2C communication between my favorite modules.

## Hardware I2C connection

Both modules have dedicated pins for I2C communication: SDA, SCL, VCC, GND.


The awesome Arduino peripheral for U3

## Arduino pins

A4 (SDA), A5 (SCL), 5V, GND

## U3 pins

pin 1: SCL (gpio199)
pin 3: SDA (gpio200)
pin 2: 1.8v
pin 7: GND

In order to establish the connection between the Arduino and the U3, we must first connect the corresponding pins. For instance, the Arduino SDA must be connected to the U3 SDA, and so on. But, it is impossible to connect the pins directly because Arduino needs 5 volts and the U3 needs 1.8 volts. So, a logic level converter is required.

I tried both the Logic Level Converter BiDirectional (http://bit.ly/1puPKJl) and the PCA9306 Level Translator Breakout (http://bit.ly/1BdMyZf), and both worked great. I ultimately decided to use the PCA9306 model because it offers a dedicated I2C bus voltage level translator.

## Software I2C connection

Before going deep into the code examples, I will first explain what data types are valid for transferring between the modules. It is important to understand that if one module sends data and the other can't decode it, then that data is junk to the receiver.

By valid data types I mean that their binary representation and sizes are the same in both modules. With a bit of adaptation, you can run the code from http://bit.ly/YeO2VW on the Arduino and see the sizes of the variables on Arduino.

After comparing the output of both modules, you can see that variables like char, long and float have the same size. The "char" and "long" data types are simple integer variables that only take up a few bytes, so it's quite safe to say that they can be passed without any problem, but what about float?

Floating point numbers are divided into two parts: mantissa and exponent. If the sizes (in bits) of these parts are different, then it is impossible to send floating point numbers, in spite the fact that they have the same size. I couldn't find the amount of bits that are assigned to the mantissa on a ATmega328p, so I simply compared the hexadecimal representation of the variables on both modules using this method:

```
printf("%08lx",
(long)*((long*)&float_number);
```

Since the representations on both modules were similar, I concluded that it is safe to pass float numbers too.

There are two methods of sending several number. The first is to send them one by one, and the other is to arrange them into a struct, and send the whole struct as one piece. If you decide to use the second option, you must ensure that the padding in both modules is the same.

As an example, look at the following struct data definition:

```
struct S { char c, long l; };
```

Structs in arduino have no padding, so the size of this struct is 5 bytes (1 byte(char) + 4 bytes(long)). If you check the size of this struct on a U3 using sizeof(S); you will find that its size is 8 bytes because of the padding. So, you need wrap the definition in the following functions in order to compensate:

```
#pragma pack(push, 1)
:
#pragma pack(pop)
```

You can read more on data structure alignment at http://bit.ly/1pfjV8m. Below is an example of the software part of the code that I wrote. In the following example, the U3 acts as a master and the Arduino as a slave. The master sends a struct to slave, then the slave prints it on a LCD screen and sends the struct back to the master. I attached a small LCD screen in order to validate that the passed numbers were received correctly. Simply set the Arduino device as slave by passing its address to the "Wire.begin()" method using the following code.

## Arduino code

```
#include <Wire.h> #include <Liq-
uidCrystal_I2C.h>
#define I2C_ADDRESS 0x09
LiquidCrystal_I2C lcd(0x27,20,4);
// set the LCD address to 0x27
for a 16 chars and 2 line display
struct Numbers {
char c; long l; float d;
};
Numbers numbers = {0};
int isNewData = 0;
int bytes_to_read;
int buf_size;
char str[100];
char double_str[100];
void setup() {
lcd.init(); // initialize the lcd
lcd.backlight(); lcd.clear();
Wire.begin(I2C_ADDRESS); // Start
I2C Bus as a Slave (Device Number
9)
Wire.onReceive(receiveEvent); //
register event
Wire.onRequest(requestEvent); //
register event
}
void loop() {
if (isNewData) {
lcd.clear();
lcd.setCursor(0,0);
sprintf(str, "%02x : %d",
numbers.c, numbers.c);
lcd.printstr(str);
lcd.setCursor(0,1);
sprintf(str, "%08lx : %ld",
numbers.l, numbers.l);
lcd.printstr(str);
lcd.setCursor(0,2);
sprintf(str, "%08lx : %s",
(long)*((long*)&numbers.d),
dtostrf(numbers.d, 0, 3, double_
str));
lcd.printstr(str);
lcd.setCursor(0,3);
sprintf(str, "%d : %d", buf_size,
bytes_to_read);
lcd.printstr(str);
isNewData = 0;
}
delay(100);
}
// function that executes whenev-
er data is received from master
// this function is registered as
an event, see setup()
void receiveEvent(int howMany) {
byte * buf = (byte*)&numbers;
buf_size = sizeof(numbers);
for (int i = 0; i < buf_size;
++i) {
buf[i] = Wire.read();
}
bytes_to_read = howMany;
isNewData = 1;
}
// function that executes when-
ever data is requested by master
// this function is registered as
an event, see setup()
void requestEvent() {
```

```
Wire.write((byte*)&numbers,
sizeof(numbers));
}
```

Next, update your U3 kernel to the latest version (see http://bit.ly/1rhz52C). Then, open terminal in superuser mode and type:

```
modprobe gpiopca953x
modprobe i2cgpiocustom
bus0=4,200,199
```

Now, the U3 can use pins (200,199) in its 8 pin header for I2C communication. Run the compiled code with root privileges, otherwise the system blocks the access to I2C device.

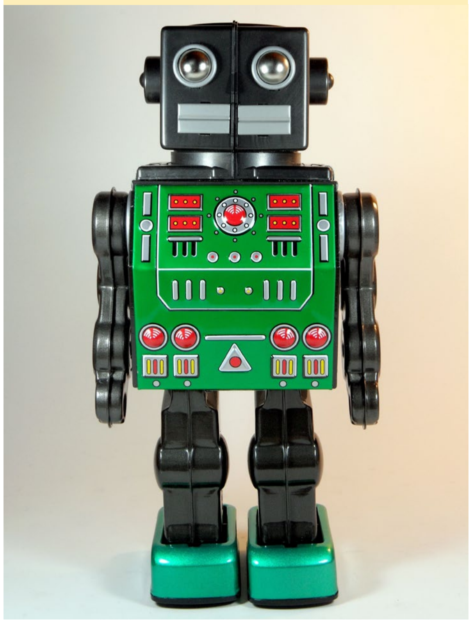
## Code Example

```
#include <unistd.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <linux/i2cdev.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <string.h>
#define SLAVE_ADDRESS 0x09
#pragma pack(push, 1) // exact fit
no padding
struct Numbers {
char c; long l; float d;
};
#pragma pack(pop) // back to
whatever the previous packing
mode was
const char * i2cDevName = "/dev/
i2c4";
int main() {
// Open up the I2C bus
int file = open(i2cDevName, O_
RDWR);
if (file == 1) {
fprintf(stderr, "Bad device name
%s\r\n", i2cDevName); exit(1);
} else {
printf("*** Device '%s' opened
successfully.\r\n", i2cDevName);
}
// Specify the address of the
slave device
if (ioctl(file, I2C_SLAVE, SLAVE_
ADDRESS) < 0) {
fprintf(stderr, "Failed to ac-
quire bus access '%x' and/or talk
to slave\r\n", SLAVE_ADDRESS);
exit(1);
} else {
printf("*** Acquired bus access
to a slave device adr: %x.\r\n",
SLAVE_ADDRESS);
}
Numbers numbers;
numbers.c = 117;
numbers.l = 876543210;
numbers.d = 1234.567;
printf("*** Send to the i2c
bus.\r\n");
printf("numbers.c = %d.\r\n",
numbers.c);
printf("numbers.l = %ld.\r\n",
numbers.l);
printf("numbers.d = %lf.\r\n",
numbers.d);
// Write a byte to the slave
if (write(file, &num-
bers, sizeof(numbers)) !=
sizeof(numbers)) {
fprintf(stderr, "Failed to write
to the i2c bus adr: %x.\r\n",
SLAVE_ADDRESS);
exit(1);
} else {
printf("*** Wrote to the i2c bus
adr: %x.\r\n", SLAVE_ADDRESS);
}
// Read a byte from the slave
Numbers n = {0};
if (read(file, &n, sizeof(n)) !=
sizeof(n)) {
fprintf(stderr, "Failed to read
from the i2c bus.\r\n");
exit(1);
} else {
printf("*** Read from the i2c
bus.\r\n"); printf("n.c = %d.\
r\n", n.c);
printf("n.l = %ld.\r\n", n.l);
printf("n.d = %lf.\r\n", n.d);
}
close(file);
return 0;
}
```

## Additional Reading

**I2C**
http://bit.ly/1rnjxaX
**Serial Communication**
http://bit.ly/1v6gHtp
**SPI**
http://bit.ly/1v6gJ4w
**Data type size**
http://bit.ly/1DAyPzs
**Data structure alignment**:
http://bit.ly/1ytfn7T

This robot just learned about the new XU3



# ANDROID DEVELOPMENT

## CREATING A CUSTOM ANDROID APPLICATION

By Nanik Tolaram

In this article, I discuss the process of creating your own Android app, as well as how to set up an Android development environment. I recommend starting with a fresh installation of Ubuntu 14.04 64bit, but feel free to use any Linux distro that you are comfortable with, as long as you have all the relevant development tools available. If you want to test it out first, you can also run Ubuntu as a virtual machine. First, you must have a basic understanding of the Java programming language, or at least have done some coding in Java, which is the de-facto programming language in the Android world.

This article will explain the different parts of an Android application using the Tic-Tac-Toe example applications that are bundled inside the Android source code. You can view many different sample applications from Google's Android source code repository at http://bit.ly/1vkVLNE.

The sample app can be checked out from http://bit.ly/1ytcbsR. There are plenty of resources on the Internet dedicated to teaching about Android apps, and the best place to start is by visiting Google's Android Development training website at http://bit.ly/1cB6RmA.

## Development setup

For this article, I will be using Ubuntu 14.04 (64bit), and will walk you through installing the different tools. There are 2 main Interactive Developement Environments (IDEs) that you can use: Eclipse or Android Studio. For this article I will be focusing on Eclipse.

**Download JDK 1.7 from the Oracle website at http://bit.ly/196ebsY. In my case, I downloaded the file "jdk-7u67-linux-x64.tar.gz".**
**Extract the .gz file into a separate directory. My file was extracted to "/home/nanik/Downloads/jdk1.7.0_67".**

Run the following command from Terminal, which instructs Ubuntu as to which version of the Java tools is installed:

```
$ sudo update-alternatives --install \
"/usr/bin/java" "java"
"/home/nanik/Downloads/jdk1.7.0_67/bin/java" 1071
$ sudo update-alternatives --install \
"/usr/bin/javac" "javac"
"/home/nanik/Downloads/jdk1.7.0_67/bin/javac" 1071
$ sudo update-alternatives --install \
"/usr/bin/javaws" "javaws" \
"/home/nanik/Downloads/jdk1.7.0_67/bin/javaws" 1071
$ sudo update-alternatives --install \
"/usr/bin/javap" "javap" \
"/home/nanik/Downloads/jdk1.7.0_67/bin/javap" 1071
$ sudo update-alternatives --install \
"/usr/bin/javadoc" "javadoc" \
"/home/nanik/Downloads/jdk1.7.0_67/bin/javadoc" 1071
```

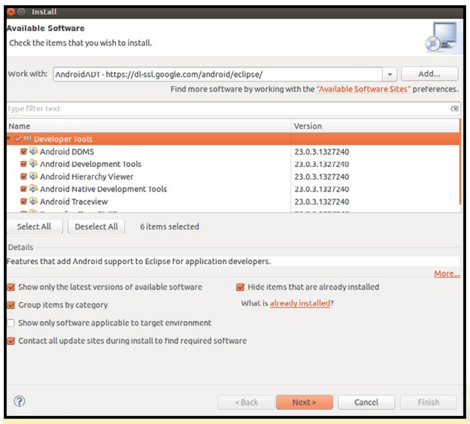
If you run the command `java -version` from Terminal, you will get the following output:

```
$ java -version
java version "1.7.0_67"
Java(TM) SE Runtime Environment (build 1.7.0_67-b01)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04,
mixed mode)
```

Upon completing the Java installation, you will also need to download the Eclipse Kepler IDE from http://bit.ly/1v5GssU.

**Extract the .gz file into a directory, such as /home/nanik/Downloads/eclipse.**
**Download the Android Development Toolkit (ADT) for Eclipse. Follow the excellent steps at http://bit.ly/1vcZCMD.**

Once you have entered the URL into the "Help > Install New Software" section, and selected the Development Tool to



Install

**Available Software**

Check the items that you wish to install.

Work with: AndroidADT - https://dl-ssl.google.com/android/eclipse/

Find more software by working with the "Available Software Sites" preferences.

| Name | Version |
|---|---|
| ✓ Developer Tools | |
| ✓ Android DDMS | 23.0.3.1327240 |
| ✓ Android Development Tools | 23.0.3.1327240 |
| ✓ Android Hierarchy Viewer | 23.0.3.1327240 |
| ✓ Android Native Development Tools | 23.0.3.1327240 |
| ✓ Android Traceview | 23.0.3.1327240 |

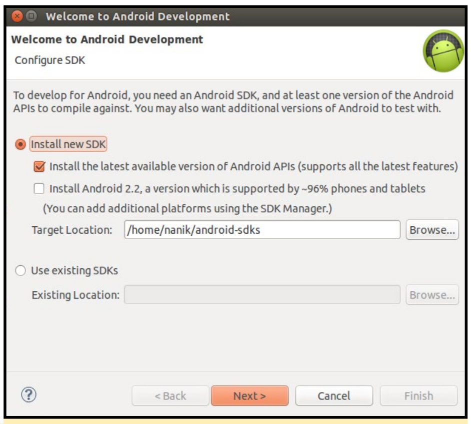
6 items selected

Details

Features that add Android support to Eclipse for application developers.

- [✓] Show only the latest versions of available software
- [✓] Hide items that are already installed
- [✓] Group items by category
- What is already installed?
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software

**Installing Developer Tools in Eclipse**



Welcome to Android Development

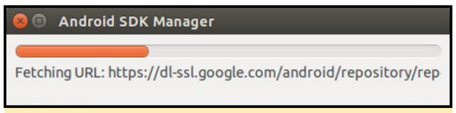
**Welcome to Android Development**

Configure SDK

To develop for Android, you need an Android SDK, and at least one version of the Android APIs to compile against. You may also want additional versions of Android to test with.

- [✓] Install new SDK
  - [✓] Install the latest available version of Android APIs (supports all the latest features)
  - [ ] Install Android 2.2, a version which is supported by ~96% phones and tablets
  - (You can add additional platforms using the SDK Manager.)
  - Target Location: /home/nanik/android-sdks
- [ ] Use existing SDKs
  - Existing Location:

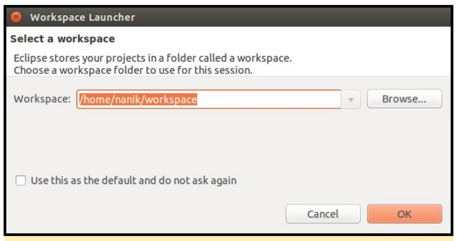
**Installing the Android SDK**



Android SDK Manager

Fetching URL: https://dl-ssl.google.com/android/repository/rep

**Downloading the Android SDK**



Workspace Launcher

**Select a workspace**

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /home/nanik/workspace

[ ] Use this as the default and do not ask again

**Selecting a workspace after installation**

show the "Available Software" screen. Click on Next, follow the instructions, and it will complete the installation.

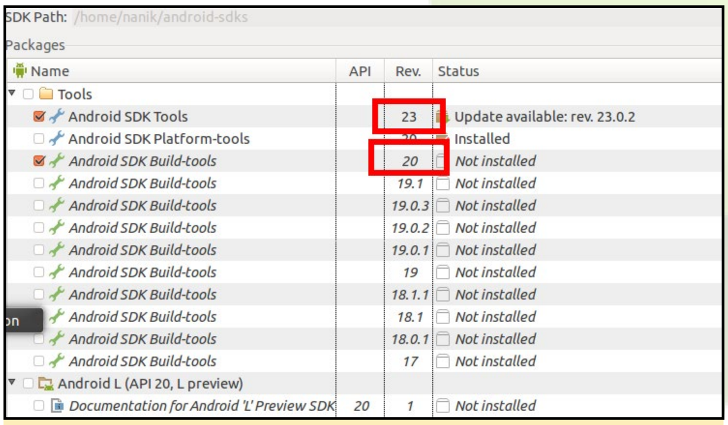
After restarting Eclipse, it will ask you to download and install the Android SDK. Upon completing the download, accept the license to continue the installation.

Besides the SDK, the installer will download the Build Tool as shown in the screenshot.

Click on "Open SDK Manager", then select the one shown highlighted in the screenshot while deselecting all the other selections.

There are 2 libraries that need to be installed in order to complete the installation inside Ubuntu. Open your terminal and execute the following command:

```
sudo apt-get install lib32stdc++6

sudo apt-get install lib32z1
```



SDK Path: /home/nanik/android-sdks

Packages

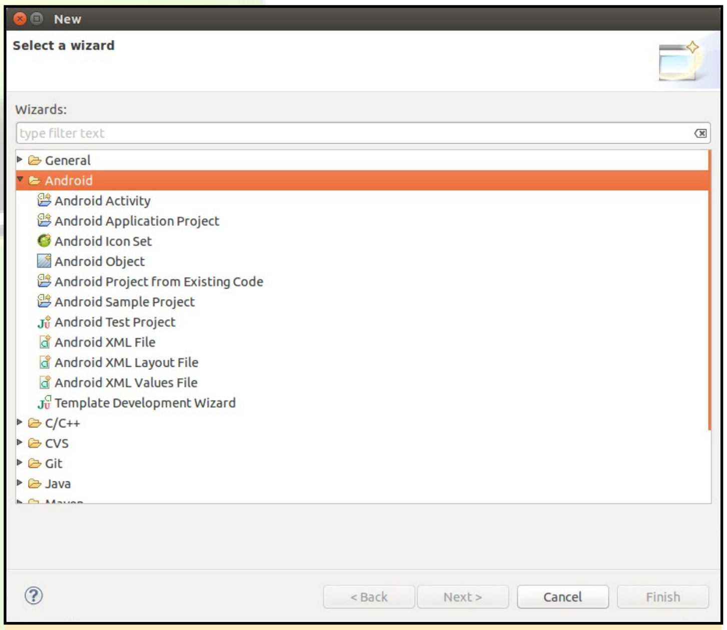
| Name | API | Rev. | Status |
|---|---|---|---|
| Tools | | | |
| ✓ Android SDK Tools | | 23 | Update available: rev. 23.0.2 |
| Android SDK Platform-tools | | 20 | Installed |
| ✓ Android SDK Build-tools | | 20 | Not installed |
| Android SDK Build-tools | | 19.1 | Not installed |
| Android SDK Build-tools | | 19.0.3 | Not installed |
| Android SDK Build-tools | | 19.0.2 | Not installed |
| Android SDK Build-tools | | 19.0.1 | Not installed |
| Android SDK Build-tools | | 19 | Not installed |
| Android SDK Build-tools | | 18.1.1 | Not installed |
| Android SDK Build-tools | | 18.1 | Not installed |
| Android SDK Build-tools | | 18.0.1 | Not installed |
| Android SDK Build-tools | | 17 | Not installed |
| Android L (API 20, L preview) | | | |
| Documentation for Android 'L' Preview SDK | 20 | 1 | Not installed |

**Android package selection of SDK tools**

Upon completing the build tool installation restart Eclipse. To check if the installation was successful, right click the "Package Explorer" tab, and you will see an Android selection screen as shown in the image above.

## Sample app setup

After completing the Eclipse installation, you are now ready



New

**Select a wizard**

Wizards:

- General
- Android
  - Android Activity
  - Android Application Project
  - Android Icon Set
  - Android Object
  - Android Project from Existing Code
  - Android Sample Project
  - Android Test Project
  - Android XML File
  - Android XML Layout File
  - Android XML Values File
  - Template Development Wizard
- C/C++
- CVS
- Git
- Java

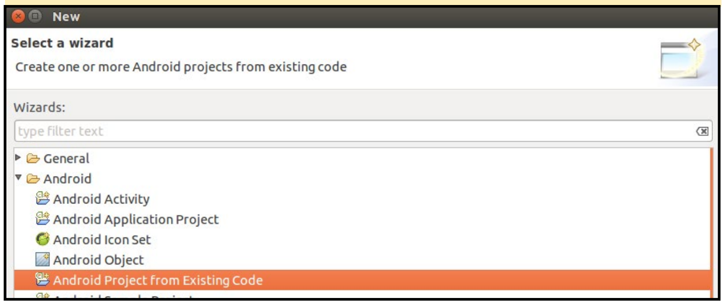
**Creating a new Android project**

to work with the sample app. Checkout the sample from Github, then follow the steps below to import it into Eclipse.

**Select File -> New -> Other and select "Android Project from Existing Code", then click "Next".**

Select the root directory of the sample app that you checked out from Github, and select "Copy projects into workspace", then click "Finish".

**Android project creation wizard**



New

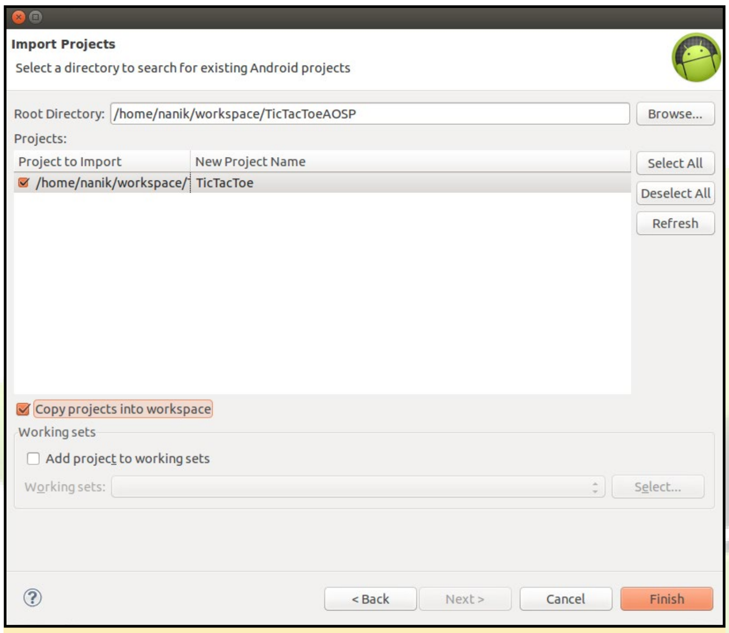
**Select a wizard**

Create one or more Android projects from existing code

Wizards:

- General
- Android
  - Android Activity
  - Android Application Project
  - Android Icon Set
  - Android Object
  - Android Project from Existing Code

You will see the project in Eclipse as shown in the screenshots below:



Import Projects
Select a directory to search for existing Android projects

Root Directory: /home/nanik/workspace/TicTacToeAOSP

Projects:

| Project to Import | New Project Name |
|---|---|
| ✓ /home/nanik/workspace/ | TicTacToe |

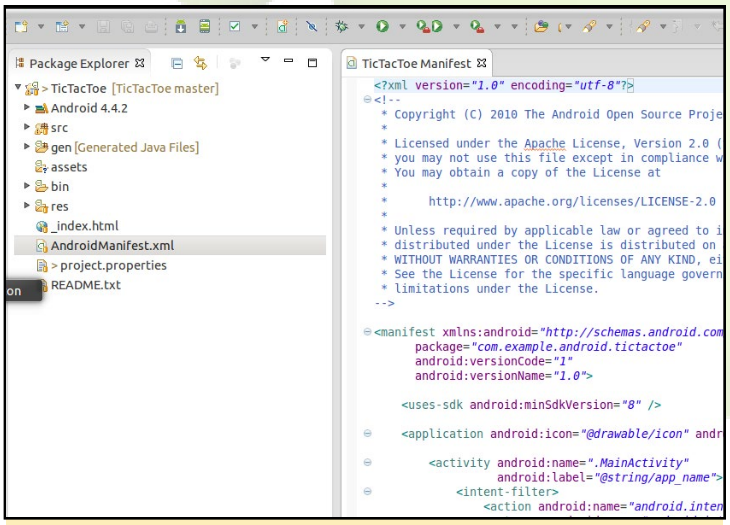
✓ Copy projects into workspace

Working sets

☐ Add project to working sets

Working sets:

The Eclipse Import Project window



```
> TicTacToe [TicTacToe master]
    Android 4.4.2
    src
    gen [Generated Java Files]
    assets
    bin
    res
    _index.html
    AndroidManifest.xml
    > project.properties
    README.txt
```

```
<?xml version="1.0" encoding="utf-8"?>
<!--
 * Copyright (C) 2010 The Android Open Source Proje
 *
 * Licensed under the Apache License, Version 2.0 (
 * you may not use this file except in compliance w
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to i
 * distributed under the License is distributed on
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, ei
 * See the License for the specific language govern
 * limitations under the License.
-->

<manifest xmlns:android="http://schemas.android.com
      package="com.example.android.tictactoe"
      android:versionCode="1"
      android:versionName="1.0">

    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" andr

        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.inten
```

The TicTacToe project code in the Eclipse Project Explorer

Don't forget that in the movie Wargames, the Global Thermonuclear War simulation was ended with a simple game of TicTacToe

To run the app on the ODROID, plug the micro USB to your computer USB, then right click on the project and select "Run As > Android Application". Eclipse will automatically detect the ODROID (make sure there are no other Android devices connected) and will run the app on your board.

## AndroidManifest.xml

An Android application does not have a "main" entry point, but does have an XML file that describes the application content. This file is the first file that is read by Android in order to know what is the content of an application, and how it can start the application. This file is called AndroidManifest.xml, and looks similar to this:

```
<manifest xmlns:android="http://schemas.android.com/
apk/res/android"
      package="com.example.android.tictactoe"
      android:versionCode="1"
      android:versionName="1.0">

    <uses-sdk android:minSdkVersion="8" />
    <application android:icon="@drawable/icon"
android:label="@string/app_name">

        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.
action.MAIN" />
                <category android:name="android.in-
tent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.example.android.
tictactoe.GameActivity" />
    </application>
</manifest>
```

You can read the following document to get an in depth understanding of the different element inside the file at http://bit.ly/1msJ804. The main elements that we need to understand are detailed below:

### <uses-sdk..>

This element lets you decide which version of Android to target. In our sample, we are targeting a minimum SDK of version 8 (KitKat is version 20).

### <activity..>

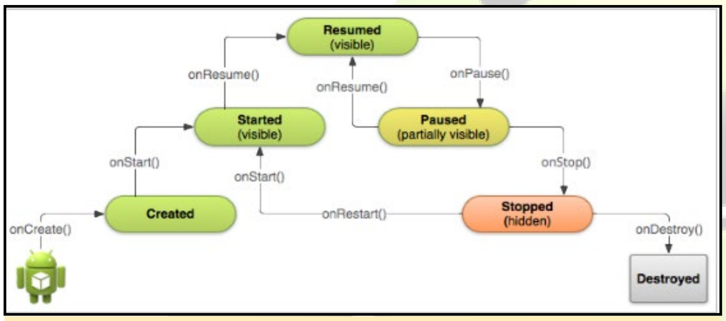
This element specifies the Activity class name that we have declared for our application.

## <intent-filter..>

In Android, everything the process communicates with one another is through Intent, which is like a message queue. This element indicates to Android which action the Activity is linked to. For our sample app, the android.intent.action.MAIN is linked to our Activity, which means that this Activity class will be the first class to be executed by the framework when the application launches.

## Activity

Android app runs by subclassing the Activity class, then following the Activity Lifecycle in order to run an application. In



**Android application activity lifecycle**

our app, we have 2 different classes (MainActivity and GameActivity), and as you can see in the code, both of these classes extend the Activity class and override onCreate(..) and onResume(..).

When an app runs for the first time, it will call the onCreate(..) method, and subsequently call the onStart(..) method. When our app goes to the background when the user switches to different app, the onPause(..) method is called, and when we switch back to our app by bringing it to the foreground, the onResume(..) method is called. The flow is simple and easy to remember since the app has only 2 running states: foreground or background.

Notice that we have 2 different elements – MainActivity and GameActivity. The way Android decides which class to run when the app starts up is by looking at the <intent-filter> element. The MainActivity has an <action..> element android.intent.action.MAIN, which tells Android that this is the main class that will have to be run for the app.

The MainActivity class is the main entry point of the whole app, which in turns call the GameActivity when the app launches:

```
private void startGame(boolean startWithHuman) {
    Intent i = new Intent(this, GameActivity.class);
    i.putExtra(GameActivity.EXTRA_START_PLAYER,
            startWithHuman ? State.PLAYER1.getValue()
: State.PLAYER2.getValue());
    startActivity(i); }
```

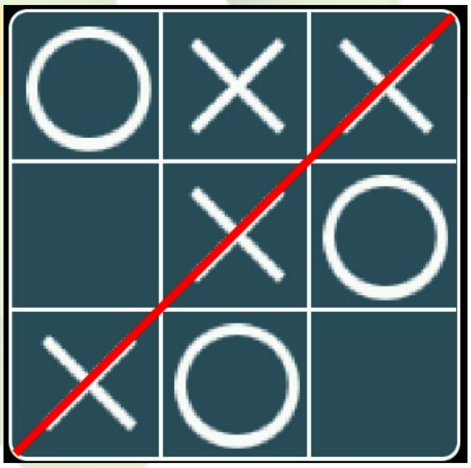
As this code snippets shows, the GameActivity class is called by calling the startActivity(..) method that is part of the Activity class. The startActivity(..) method is a method that instructs Android to execute the specified Activity class that has been declared. In our example, we have declared the Intent using the variable "i".

Android uses interprocess communication between same or different apps extensively. This allows applications to 'reuse' other parts of an application as if it were part of its own internal libraries using the Binder infrastructure. To use this facility, any app that wants to do interprocess communication uses an Intent object. This can be seen in our code snippet above where we defined the variable "i" to create a new Intent class with GameActivity as one of the parameters.

## View

When you run an Android app, what is drawn on your screen is based on a class called View. This class is the building block of your user interface, since everything you need to put on the screen must be put on the View. To explore further into the UI world of Android, you can take a look at the documentation available at http://bit.ly/1ss7o8p.

The standard view is not suitable for our sample app on Android, since it only provides a view for a widget like a textbox



**A TicTacToe game in progress**

or checkbox. Controlling the view gives the app flexibility with respect to how the UI will look.

In our sample app we have a class called GameView that extends the View class:

```
public class GameView extends View {

    public static final long FPS_MS = 1000/2;
}
```

| | |
|---|---|
| onDraw(..) | This is the main method that contains most of the code, and will be executed every time Android needs to draw on the screen. You can think of this function as the 'paint' method. |
| onMeasure(..) | This method is called to set the view measurement. |
| onSizeChanged(..) | This method is called when view size has changed, or when it is beginning to initialize the view hierarchy. |
| onTouchEvent(..) | This method is called when there has been a touch or mouse click event initiated by the user. |
| onSaveInstanceState(..)<br>onRestoreInstanceState(..) | These 2 methods are used to save and restore the state of the game. |

The logic to build the cubes, along with the state of the player, is inside the onDraw(..) function. Everytime Android needs to refresh the view, it will call this method, so it is important that code runs in this method are as quick and efficient as possible, in order to avoid lag when users are interacting with the app.

## Handler

Android is designed to rely on asynchronous processing, where messages flow back and forth easily between apps. Because of this design, apps utilize a Handler class, which acts like a "callback", where it processes incoming messages. The sample app makes use of this class for blinking the cell containing the user selection.

```
@Override
    public boolean onTouchEvent(MotionEvent event) {
                ...
                ...
                if (state != State.EMPTY) {
                    // Start the blinker
                    mHandler.
sendEmptyMessageDelayed(MSG_BLINK, FPS_MS);
                }
```

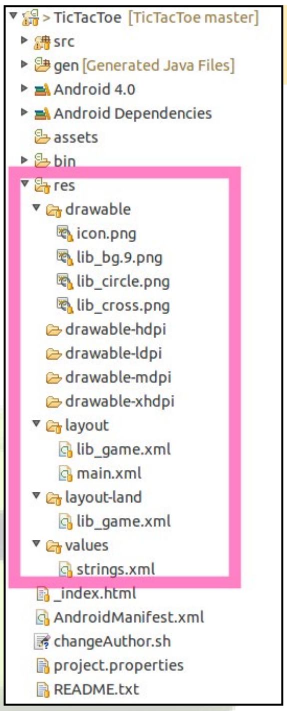
The handler sends a message (MSG_BLINK) at a particular interval defined by the variable FPS_MS. The Callback class that the app defines to receive the MSG_BLINK messages are defined like this:

```
    private class MyHandler implements Callback {
         public boolean handleMessage(Message msg) {
             if (msg.what == MSG_BLINK) {
                 ...                    }
                 }
                 return true;
             }
             return false;
         }
     }
```

## Resources (Graphics and Strings)

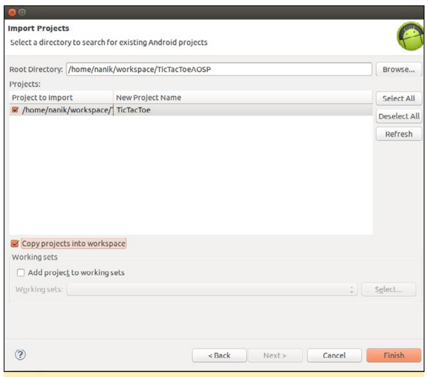
Android apps store strings, images and other binary resources inside a separate folder that can be referred to by the app. All of the resources are stored inside the res/ folder. During the compiling and packaging process, the file inside this folder will be stored as shown in the screenshot.

The generated R.java file contains an ID for each of the defined resources in our app. The way in which the app accesses resources is by using the built-in Android API:



```
▼ > TicTacToe [TicTacToe master]
  ▶ src
  ▶ gen [Generated Java Files]
  ▶ Android 4.0
  ▶ Android Dependencies
    assets
  ▶ bin
  ▼ res
    ▼ drawable
        icon.png
        lib_bg.9.png
        lib_circle.png
        lib_cross.png
      drawable-hdpi
      drawable-ldpi
      drawable-mdpi
      drawable-xhdpi
    ▼ layout
        lib_game.xml
        main.xml
    ▼ layout-land
        lib_game.xml
    ▼ values
        strings.xml
    _index.html
    AndroidManifest.xml
    changeAuthor.sh
    project.properties
    README.txt
```

Contents of the res/ folder



Import Projects
Select a directory to search for existing Android projects

Root Directory: /home/nanik/workspace/TicTacToeAOSP

Projects:

| Project to Import | New Project Name |
|---|---|
| ☑ /home/nanik/workspace/ | TicTacToe |

☑ Copy projects into workspace

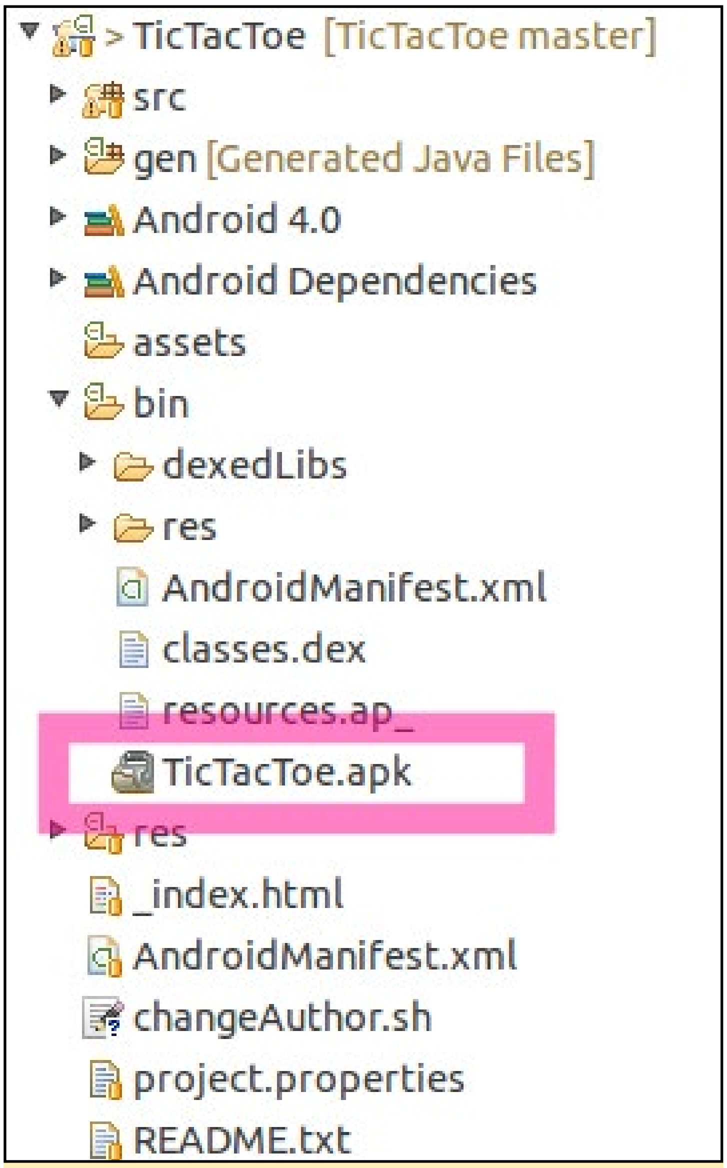
Working sets
☐ Add project to working sets
Working sets:

After generating the res/ folder in Eclipse

```
 public GameView(Context context, AttributeSet attrs)
{
       ...

        mDrawableBg = getResources().getDrawable(R.
drawable.lib_bg);
        setBackgroundDrawable(mDrawableBg);

        mBmpPlayer1 = getResBitmap(R.drawable.lib_
cross);
        mBmpPlayer2 = getResBitmap(R.drawable.lib_
circle);
       ...
}
       ...
       ...

    private Bitmap getResBitmap(int bmpResId) {
       ...

        Resources res = getResources();
        Bitmap bmp = BitmapFactory.
decodeResource(res, bmpResId, opts);

        if (bmp == null && isInEditMode()) {
            ...
            Drawable d = res.getDrawable(bmpResId);
            ...
    }
}
```

## APK

Your app will be packaged into a single APK file. If you would like to know more about the internal workings of an .APK, please refer to my article in the September 2014 issue.



```
▼ > TicTacToe [TicTacToe master]
  ▶ src
  ▶ gen [Generated Java Files]
  ▶ Android 4.0
  ▶ Android Dependencies
    assets
  ▼ bin
    ▶ dexedLibs
    ▶ res
      AndroidManifest.xml
      classes.dex
      resources.ap_
      TicTacToe.apk
  ▶ res
    _index.html
    AndroidManifest.xml
    changeAuthor.sh
    project.properties
    README.txt
```

**The resulting TicTacToe.apk file after building**

## Installation

You can run your application from Eclipse by right clicking on the project name, then selecting Run As -> Android Application. If you are encountering issues running the app from Eclipse, as I sometime experience when running inside a virtual machine, you can also run it from command line by using adb:

**1.Type adb devices to make sure you have a connection to the ODROID.**
**2.Use the command adb install TicTacToe.apk to install the app once the connection has been established. The location of the apk is normally inside the /bin folder on your Eclipse workspace.**

# *8-BIT COMPUTING BONANZA*

## *HAVE FUN PLAYING YOUR FAVORITE MSX GAMES*

**by Bruno Doiche**

Enjoy all the rage of modern computing ranging from 1983 to 1995 emulating the one and only MSX. Before the appearance and great success of Nintendo's Family Computer, MSX was the platform for which major Japanese game studios, such as Konami and Hudson Soft, produced video game titles. The Metal Gear series, for example, was originally written for MSX hardware, so game on!

```
$ wget http://sourceforge.net/projects/openmsx/\
  files/openmsx/0.10.1/openmsx-0.10.1.tar.gz
$ tar -zxvf openmsx-0.101.tar.gz && \
  cd openmsx-0.10.1
$ sudo -s ./configure && make -j4 && make install
```

You may need to install the following dependencies:

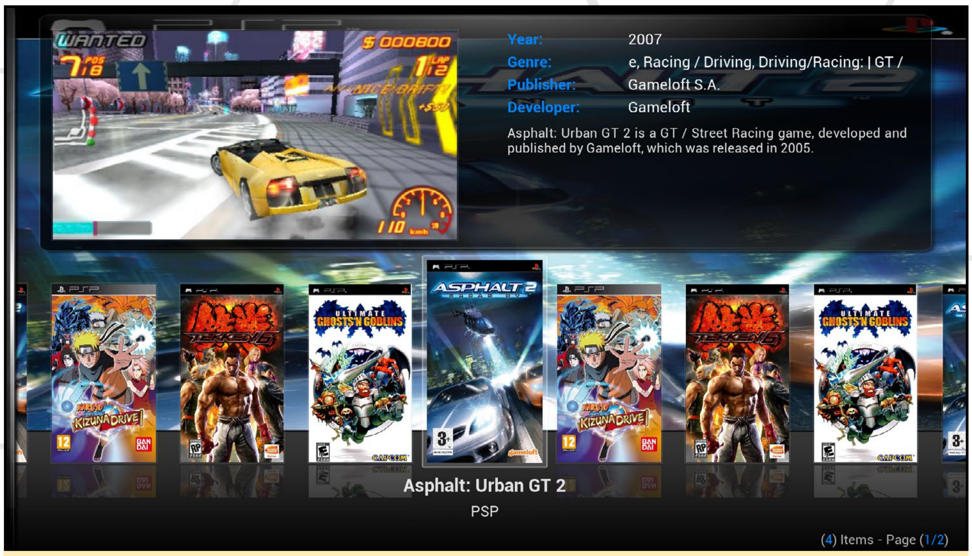
**GLEW,**
**libao,**
**libogg,**
**libpng,**
**libtheora,**
**libvorbis,**
**libxml2,**
**OpenGL,**
**SDL,**
**SDL_ttf,**
**Tcl,**
**zlib**

Copy your favorite MSX .rom file to `~/.openMSX/share/software/`, then type `openmsx <gamename.rom>`.

# LINUX GAMING

## PSP EMULATION COMPARISON BETWEEN LINUX AND ANDROID

by Tobias Schaaf



Year: 2007
Genre: e, Racing / Driving, Driving/Racing: | GT /
Publisher: Gameloft S.A.
Developer: Gameloft

Asphalt: Urban GT 2 is a GT / Street Racing game, developed and published by Gameloft, which was released in 2005.

**Asphalt: Urban GT2 (PSP) is one of many great games available using a PPSSPP emulator**

There are many benefits to playing games on Linux over Android, and for this I'll compare the performance of the well-known PlayStation Portable (PSP) emulator called PPSSPP, available for both Andoid and on Linux. Since I'm not experienced at using Android, all the screenshots used in this article are captured from the Linux version of PPSSPP..

## PPSSPP settings

To establish a meaningful comparison between the two systems, I used the same settings on both systems at 1080p resolution:

| | |
|---|---|
| **Frameskipping:** | **3** |
| **Auto frameskip:** | **ON** |
| **Rendering Resolution:** | **2x PSP** |
| **Mipmapping:** | **OFF** |
| **Hardware transform:** | **ON** |
| **Software skinning:** | **ON** |
| **Vertex Cache:** | **ON** |
| **Lazy texture caching:** | **ON** |
| **Retain changed textures:** | **ON** |
| **Disable slower effects:** | **ON** |
| **Spline/Bezier curves quality:** | **Medium** |

Everything else is defaulted to the standard settings, with no hacks activated. As a benchmark, I set the "Show FPS counter" option to BOTH, which gives a detailed view of the performance of the emulator as a ratio and percentage: 40/60 (100%). The first number is the current Frames Per Second (FPS), which shows how fast the emulator is currently rendering. The second number is the expected FPS, which varies from game to game, as well as from scene to scene. Some games only want to run at 30 or 40 FPS in a certain scenes (for example, during video playback), while others might always want to run at 60 FPS.

The last number in % gives the current speed of the emulator, compared to the required speed. 100% means the game is running in full speed. This is possible even if the emulator is only able to render 40 out of the 60 FPS which it really wants to have, because the frameskip option is enabled. If the value drops significantly (80% or less) you will experience lagging in the game.

## Overview

I used my ODROID GameStation Turbo image to test the games on Linux and Android, running with the latest 4.4.4 KitKat version provided by Hardkernel. The settings are not very different from the default settings, except with some speedup options activated, which are probably not even needed, and frameskip, which mostly increases performance.

I noticed that the resolution at which games are rendered is primarily responsible for how well the emulator performs. The default settings only use 1x PSP resolution for rendering, which is the lowest setting. This means that the games will be rendered in 480x272, which as you can imagine, looks blocky when stretched to a 1920x1080 screen resolution. With the 2x setting, you can get 920x544 resolution, which looks nice at 1080p. Some games are even able to run in 3x PSP resolution under Linux at a decent speed.

Just to see the difference, try running a game in 1x PSP, followed by changing the settings to 2x and 3x resolution, and you'll understand why this is also responsible for the performance impact. If you choose "1:1 Auto" as a resolution, PSP will be rendered in a resolution closest to the native screen resolution,which is very resource intensive.

## Games

I chose 4 games to compare Linux

and Android performance on PSP:

**Tekken 6 – a representative of a fighting game, since many people seem to like it**
**Ultimate Ghosts 'n Goblins – a platformer / jump and run, which was one of the harder Arcade games**
**Asphalt Urban GT2 – a racing game, because of its high resource usage**
**Naruto Shippuuden: Kizuna Drive – another fighting game with missions instead of personal combat**

I decided to try out Tekken 6 first, since it's very popular, and is constantly mentioned in the ODROID forums. Although the 3D graphics are not outstanding, it tends to be laggy and use a lot of resources, which makes it a good test for comparison between Android and Linux.

**Tekken 6 (PSP) gameplay is very fast, and the graphics show off the power of the ODROID**

**You've earned it if you can win against Ling Xiaoyu in Tekken 6 - she is not easy to beat!**

## Tekken 6 - Android

The Android version of Tekken 6 works without any major issues, and the game can be fuly enjoyed at a playable speed. The introduction is slightly laggy, and jumps between 20 and 40 FPS (out of 60 FPS), since it's using frameskip. There are some horizontal lines during the movie playback which means the game is a little out of sync during playback.

The menu is running at 30 out of 60 FPS, which is noticeable when moving around quickly in the menu, and sound sometimes stutters. Gameplay varies between 12 and 20 FPS, with the overall speed between 95 and 100%, with rare cases of lag. The overall experience on Android is acceptable, and with some tweaking of the emulator, it should be a good enough to play normally.

## Tekken 6 - Linux

Tekken runs noticeably better on Linux than on Android. The intro runs at a steady 60 FPS, with no horizontal lines or other issues during movie playback. Menu speed remains at 60 FPS as well, with no sound issues and very fluent reactions. The graphics look great, the shadow and light effects are there, and it runs at full speed. Framerates during gameplay remain between 15 and 25 FPS. The action is very fluent, and gives a feeling of a fast fighting style. Overall the experience of Tekken 6 on Linux is better than on Android.

Tekken 6 is a nice fighting game for PSP, although I prefer games like Soul Calibur over Tekken or Street Fighter. Both the Android and Linux versions are fully playable, with only minor graphical issues. Both run fast enough to eliminate any slowdown and lag in gameplay, although the Linux version is a little faster than the Android version, particularly during video playback and menu navigation.

Overall I would give compatibility on ODROID for Tekken an 8 out of 10 for Android and a 9 out of 10 for Linux. This game is definitely a WIN.

## Ultimate Ghosts 'n Goblins

As a platformer, this game is more of a 2D than a 3D game without using a lot of fancy graphics. As a result, the game makes very efficient use of the CPU and GPU, and is a good example of a lightweight game running on PPSSPP.

Ultimate Ghosts 'n Goblins is a remake of the old arcade game Ghosts 'n

60/60 (100.0%)
ULTIMATE GHOSTS'N GOBLINS
PRESS START BUTTON
© CAPCOM CO., LTD. 2006 ALL RIGHTS RESERVED.

Ultimate Ghosts 'N Goblins (PSP) is one of the hardest arcade games of all time

The G'n'G sequel includes graphical and sound improvements, and is very fun to play!

Goblins, which was one of the hardest games of its time. The new version is not much easier than its predecessor, but is a rather fun game to play. It's worth noting that the original game had an issue with a bloom effect in the first level which made the screen glitch, and the performance dropped dramatically until level 2 was reached. This issue was actually fixed within the PPSSPP emulator code, and since version 0.9.8, the game can be played without experiencing bloom issues.

## UG'nG - Android

Similar to Tekken 6, the intro runs at 30/60 FPS, but at least it doesn't have horizontal lines or other graphical abnormalities. This game is hard, and you should use a gamepad with Android, since playing it with a keyboard is nearly impossible. Menu control works great, and so do the sound and music.

However, the actual gaming experience on Android is nearly unplayable. PPSSPP for Android is only able to render about 8 to 13 FPS, which makes the game very laggy. This is odd, since the game is not that demanding, and was even running on older versions of PPSSPP on Linux very well, even when it still had the bloom issue, even though it wasn't optimized for hardfloat images.

## UG'nG - Linux

The Linux experience with Ultimate Ghosts 'n Goblins is perfect, with absolutely no problems. The intro, menu, and music are all in full speed (60/60 FPS), and the game runs without any noticeable speed drop. There isn't much else to say about the Linux version, except that it's very fun to play!

I don't understand why the Android experience is so bad with this game. Still, the intro, music and menu seems to work well, and with some tweaking, the Android version can probably run UG'nG at 15/30 FPS. On Android, I give UG'nG 5 out of 10 points, while the Linux version gets 10 out of 10 for compatibility and user experience.

## Asphalt Urban GT2

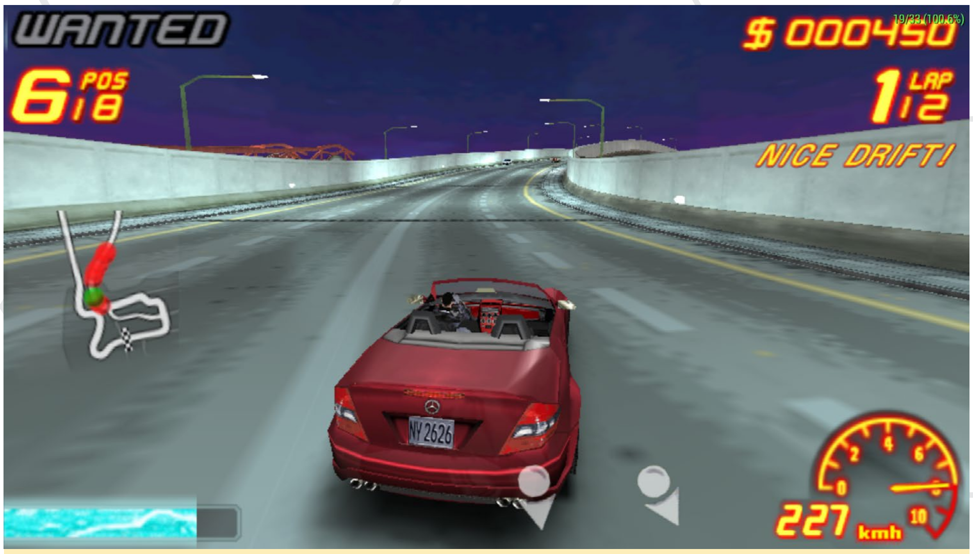
This was one of the first racing games that I played on the PSP. It's not that hard as a racing game, but you also don't have many options compared to Need for Speed or Midnight Club. However, you still have plenty of different cars and tracks to play. You start with a very high amount of nitro, and it's really fun to kick your opponents (and the cops) out of the way for some extra cash.

Although this game is rather small (a 400 MB .cso rom file), it's one of the hardest games to get running well on the ODROID. Ever since I got my first version of PPSSPP working, this game has been far from running at full speed. It was always a performance test for me, since sometimes not even my laptop, which has a modern NVIDIA gaming graphics card, is able to play it at full speed.

Asphalt Urban GT2 is a very nice racing game with a wide variety of tracks and cars. However, the gaming experience can vary a lot depending on your settings and the version of the emulator that you're using. A wrong setting can cause a previously working version to run at less than 10 FPS.

It also seems to use a lot of special effects that aren't completely supported on PPSSPP yet, so it's a really useful game to check for improvements and bug fixes as developers release new versions of PPSSPP.

Asphalt Urban GT (PSP) features lots of cars and challenging tracks



Asphalt Urban GT lets you drift cars around corners, and acts like a real car

## Asphalt Urban GT2 - Android

As expected because of the graphical intensity, Android has performance issues when running this game. Even the 2D logos during the introduction are not rendered in full speed (30/60 FPS). The menu runs surprisingly well on the start screen (between 25 and 40 FPS), and if you go into a submenu (for example, "Arcade - Single Race") or preview a car, it drops to a steady 15 FPS but is still running at full speed.

However, the gameplay on Android is not as good as the menu. Although the benchmark varies between 20 FPS down to 7 FPS, it doesn't even feel like it's running at full speed when it hits 20 FPS and 100%. It's playable, but Asphalt Urban GT2 on Android is not really a nice experience.

## Asphalt Urban GT2 - Linux

The overall experience with Asphalt Urban GT2 is slightly better on Linux: logos are at 60 FPS, the menu is about 40-50 FPS, and submenus are at 15-20 FPS. During gameplay, the framerate drops to 10 to 25 FPS as well, but feels slightly faster than Android. Unless you use your nitro, it actually performs like an authentic PSP console.

The game has some odd behavior sometimes: for example, when you turn on multi-threading, it actually gets slower, and frameskip can also have a negative effect on the performance of the game. Light effects also often seem as if they are misplaced.

Although this game is rather small in size, it offers some nice features, has a very good soundtrack and the game is fun to play. However, it runs slowly, and has a lot of glitches. It will be nice to see how the emulator evolves, and I'm looking forward to playing this game at full speed on the ODROID soon.

## Naruto Shippuuden: Kizuna Drive

Naruto Shippuuden: Kizuna Drive is another fighting style game where you complete missions rather than fighting against single enemies, like Tekken. You often fight multiple enemies at the same time, and you can fight in teams instead of personal combat. It has attractive comic-style graphics, and since the game is rendered in 30 FPS (rather than 60 FPS), it is rather easy on the hardware.

Boss fights are very interesting, and you often need your entire team to beat an enemy. In so-called "free missions", you can fight as any character you like, and are not restricted to only using the Naruto character.

## Kizuna Drive - Android

The game, menu and opening movie run well on Android at 30FPS. Gameplay varies between 20 to 30 FPS, but is most of the time above 25 FPS with no slowdowns.

## Kizuna Drive - Linux

Kizuna Drive runs at full speed without any issues. The Linux version is, in fact, so well-performing that it looks amazing when played in 3x PSP resolution or even in 1:1 Auto modes.

Naruto Shippuuden – Kizuna Drive is one of the best running games on the PPSSPP emulator. It works perfectly on either Android or Linux, while on Linux, it performs extraordinarily well, and allows you to really push the graphics to make the game look beautiful in 1080p resolution.

## Summary

General speaking, PPSSPP runs faster and with less issues on Linux than with Android, even though PPSSPP was never intended to run on hardfloat systems. However, when using PPSSPP on Debian Wheezy (such as my ODROID GameStation Turbo image), games will stop for a few seconds every now and then during play.

I'm not quite sure what causes the pausing behavior, but I think it is an I/O issue that happens when accessing the .cso images.

The lag only seems to happen with Debian Wheezy, and doesn't happen when using Ubuntu. Some games do it more often. Overall, PPSSPP is a well-written emulator which runs very well on ODROID devices using either Linux or Android. It has the advantage of using OpenGL ES 2.0, which enables the full graphics power of the ODROID.

Naruto Shippuuden: Kizuna Drive(PSP) is like Mortal Kombat with combat missions and a 3D world to explore

The boss fights in Naruto Shippuuden: Kizuna Drive will challenge you for hours, and take lots of practice

He found my copy of GameStation Turbo, and has been up for three days trying to beat the last boss in Kizuna Drive

The winner between Linux and Android versions of PPSSPP is clearly Linux, which boasts high frame rates and fluid movements

# OS SPOTLIGHT: QUIET GIANT

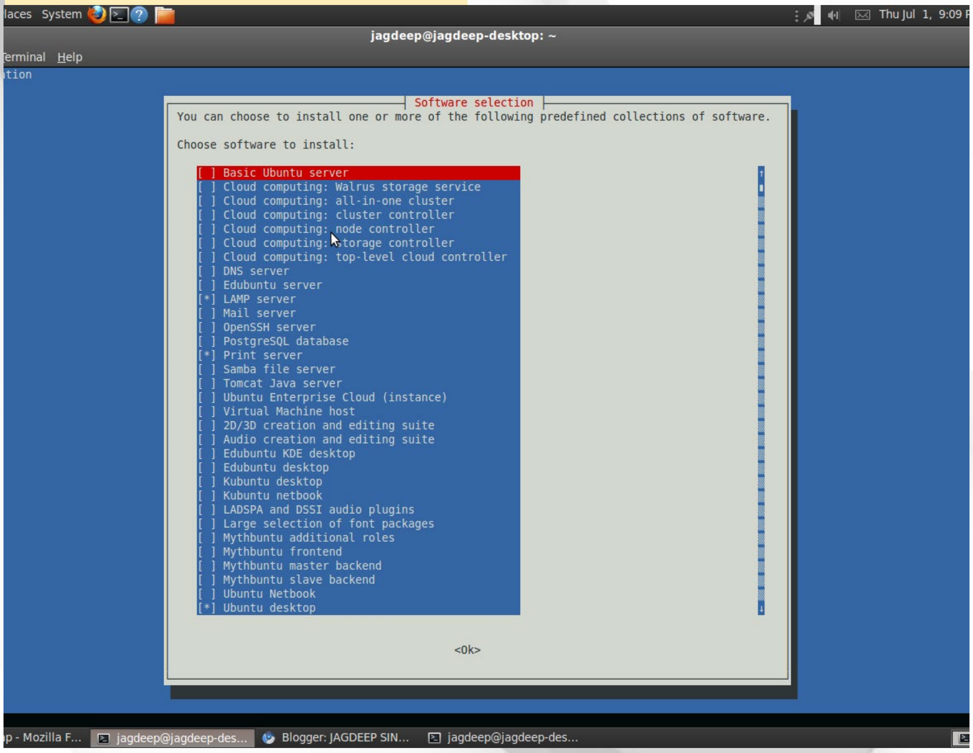
## A LIGHTWEIGHT LAMP, SAMBA, AND MINECRAFT SERVER

**Quiet Giant for the X, U and XU series may be downloaded from http://bit.ly/1rhHymu**

by Rob Roy

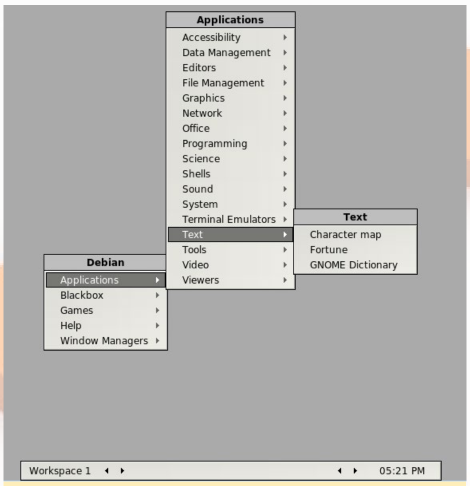
Quiet Giant, a downloadable image for the X, U and XU series based on Ubuntu Server, offers several different servers including Apache, Tomcat, MySQL, FTP, Samba and Minecraft. It's intended as an easy-to-use development sandbox LAMP server, but can also serve as a lightweight platform for an embedded system requiring long-term stability. LAMP, which stands for Linux, Apache, MySQL and PHP/Perl, is a popular choice for Internet applications, and Quiet Giant works great as an affordable learning platform for web developers.

**Tasksel Command**



```
laces  System                                         Thu Jul  1,  9:09
                    jagdeep@jagdeep-desktop: ~
erminal  Help
tion

              ┤ Software selection ├
 You can choose to install one or more of the following predefined collections of software.

 Choose software to install:

   [ ] Basic Ubuntu server
   [ ] Cloud computing: Walrus storage service
   [ ] Cloud computing: all-in-one cluster
   [ ] Cloud computing: cluster controller
   [ ] Cloud computing: node controller
   [ ] Cloud computing: storage controller
   [ ] Cloud computing: top-level cloud controller
   [ ] DNS server
   [ ] Edubuntu server
   [*] LAMP server
   [ ] Mail server
   [ ] OpenSSH server
   [ ] PostgreSQL database
   [*] Print server
   [ ] Samba file server
   [ ] Tomcat Java server
   [ ] Ubuntu Enterprise Cloud (instance)
   [ ] Virtual Machine host
   [ ] 2D/3D creation and editing suite
   [ ] Audio creation and editing suite
   [ ] Edubuntu KDE desktop
   [ ] Edubuntu desktop
   [ ] Kubuntu desktop
   [ ] Kubuntu netbook
   [ ] LADSPA and DSSI audio plugins
   [ ] Large selection of font packages
   [ ] Mythbuntu additional roles
   [ ] Mythbuntu frontend
   [ ] Mythbuntu master backend
   [ ] Mythbuntu slave backend
   [ ] Ubuntu Netbook
   [*] Ubuntu desktop

                          <Ok>

p - Mozilla F...  jagdeep@jagdeep-des...  Blogger: JAGDEEP SIN...  jagdeep@jagdeep-des...
```

Once booted, the username and password are both "odroid", which opens onto the Blackbox desktop. Right-click on the desktop and select "xterm" from the menu to open a Terminal window. Most Linux server maintenance is done from the command line, and any servers that aren't needed may be removed with the `tasksel` command. There are many other application bundles available using tasksel, including desktop packages such as Kubuntu and Unity. However, the image is pre-tuned for high performance by including the minimal Blackbox desktop, which has a sparse interface and low memory usage. Approximately 350MB of RAM is used when all 8 servers are running, leaving 1.65GB free for server use.



**Blackbox is set as the default desktop for Quiet Giant**

Although Quiet Giant includes the Blackbox desktop, it's more common to access the machine to perform upgrades in headless mode using an Secure SHell (SSH) client. File transfers are done via File Transfer Protocol (FTP) to specified directories, where they may be picked up by scheduled processes running on the server. Alternatively, the Samba protocol is also available for sharing files with other computers on the network, and can be used to create a simple media server.

## SSH

Secure Shell (SSH) is the most basic communication service available on

Quiet Giant. SSH projects a Terminal command window over Ethernet so that commands may be launched remotely.

## HTTP/Apache

I use Quiet Giant as a web server on my local home network, which allows me to develop websites without exposing them to the Internet. However, it's also possible to use Quiet Giant as a public server, with some modifications.

When choosing to make the server available publicly, it's critical to install a router between the server and the Internet, so that the router's firewall can protect the ODROID server from random hacking. It's only safe to use a computer as public Internet server if important private data is not stored on the same network as the server. Also, make sure that the passwords for each service are secure before configuring the router.

## Internet web server

To create a publicly available web server, first uninstall all other services except for Apache and MySQL, for security. Then, configure the local router to forward incoming web requests to the Quiet Giant server, as detailed in the following steps.

- Note the web server's local (private) IP address by typing the following into the server's Terminal window or SSH:

```
$ ifconfig | grep inet | grep Bcast

inet addr:192.168.1.10
Bcast:192.168.1.255
Mask:255.255.255.0
```

- Assign a permanent IP address, which is 192.168.1.10 in this example, to the web server using the router's administration panel.

- Once the IP address has been reserved, port 80, which is standard for HTTP, should be forwarded to that IP address, again using the router's Port Forwarding adminsitrative panel.

- After the router has been properly configured, type the following into a Terminal window on the server in order to discover its public IP address:

```
$ curl -s checkip.dyndns.org|sed
-e 's/.*Current IP Address: //'
-e 's/<.*$//'
79.211.83.113
```

In this example, the server's address is 79.211.83.113, which may be accessed from any browser worldwide using "http://79.211.83.113" whenever the ODROID and router are connected to the Internet. Similar services, including File Transfer Protocol (FTP) and SSH, may also be publicized with the same port-forwarding technique, using the corresponding port for those services, such as Port 21 for FTP.

## MySQL

MySQL uses Port 3306 by default, and enables websites and other applications to access information from a database using a special programming language called Structured Query Language (SQL). When paired with another programming language such as PHP, user input, log data, and other information may be recorded, retrieved and archived.

The version of MySQL installed on Quiet Giant comes with a blank admin password. For security, the default password should be changed immediately. The procedure for updating the password is described on http://help.ubuntu.com:

- First, stop the mysql process by typing the following into Terminal:

```
sudo /etc/init.d/mysql stop
```

- Then, type this to restart the mysqld daemon:

```
sudo /usr/sbin/mysqld --skip-
grant-tables --skip-networking &
```

- Next, restart the mysql client process:

```
mysql -u root
```

- From the MySQL prompt, execute this command in order to reset the root password:

```
FLUSH PRIVILEGES;
SET PASSWORD FOR root@'localhost'
= PASSWORD('password');
FLUSH PRIVILEGES;
exit;
```

- Finally, stop the mysql process and relaunch it:

```
sudo /etc/init.d/mysql stop
sudo /etc/init.d/mysql start
```

## Samba

Samba is a Windows-based file sharing protocol, and Quiet Giant has pre-configured Samba shares located at /var/www/ (Apache), /var/lib/tomcat7/webapps/ROOT (Tomcat), and /home/odroid/Documents (general use). Connect to Samba by typing the local IP address of the Quiet Giant server into a file explorer from any other computer on the network, and supplying the default username and password of "odroid". This will gives access to the shared directory.

To configure Samba, edit the file /etc/samba/smb.conf in a Terminal window. An intuitive GUI for managing users and folders is also available when using the Blackbox desktop:

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

## Tomcat

Tomcat is a Java-based open-source web server that can be used to build nearly any type of web application. An installed application may be accessed by visiting http://127.0.0.1:8080 from the server, or by typing the internal (private) IP address of the server, followed by ":8080", from any computer on the local network. If setting up a public Internet server, Port 8080 should be forwarded to the ODROID as described above. For more information on programming and contributing to the Tomcat project, visit http://tomcat.apache.org/.

## Spigot (Minecraft)

Everyone loves Minecraft! Quiet Giant comes with an optimized version of the Minecraft server called Spigot version 1.6.4 is installed in /home/odroid/Public/, and can be started by typing the following commands in Terminal:

```
cd ~/Public/spigot
./spigot.sh
```

## Virtual memory

Swap is enabled in the kernel, which extends memory beyond 2GB by writing blocks of memory to disk, either as a single file on the root file system, or on a separate dedicated partition. To learn more about setting up a swap file, refer to http://bit.ly/1pYfWSY. For information on creating a swap partition, visit http://bit.ly/1rdONWu.

## Clock synchronization

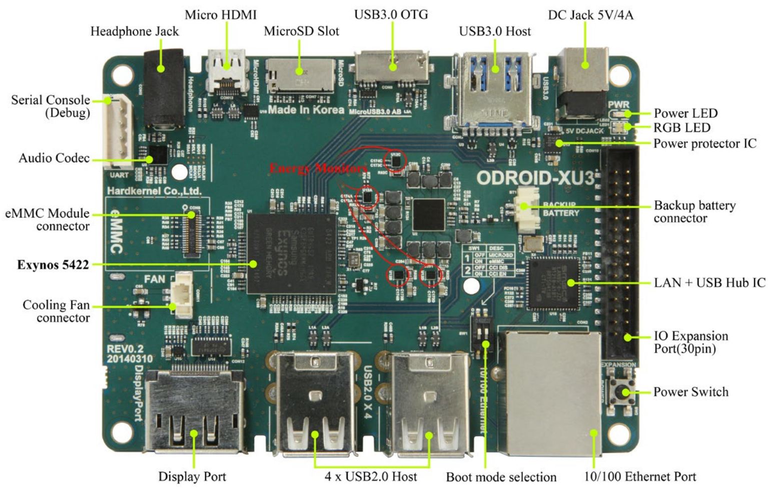
When the image is booted without a wired LAN plugged in, the clock time may become out-of-sync, unless you are using a clock battery. With Linaro, this time difference can cause the root file system to be mounted as read-only. If this happens, type “fsck /” while logged in as root, then reboot with the LAN properly attached. This will unlock the file system and re-synchronize the clock with Internet time via NTP.

**Make sure to keep your server up-to-date with the latest patches from Ubuntu**

# ODROID-XU3

## THE FASTEST COMPUTER MADE BY HARDKERNEL SO FAR!

**by Justin Lee**



**The ODROID-XU3 is an 8-core ARM big.LITTLE Single Board Computer**

The ODROID-XU3 is a new 8-core micro Single Board Computer (SBC) powered by ARM® big.LITTLE™ technology and utilizing a Heterogeneous Multi-Processing (HMP) solution. It’s a member of a new generation of computing devices with more powerful, energy-efficient hardware and smaller form factor. Offering open source support, the board can run various flavours of Linux, including the latest Ubuntu 14.04 and the Android 4.4. By adopting eMMC 5.0 and USB 3.0 interface, it boasts fast data transfer speed, a feature that is increasingly required to support advanced processing power on ARM devices that allows users to fully experience an upgrade in computing such as faster booting, web browsing and 3D game experience.

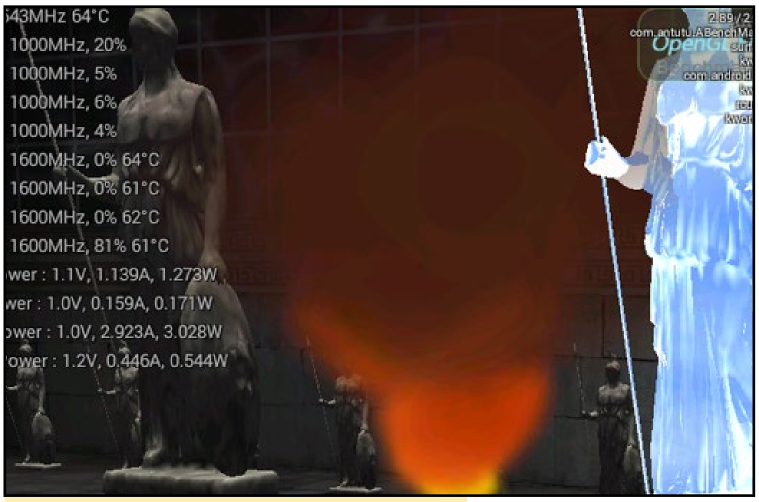
- Samsung Exynos 5422 Cortex™-A15 2.0Ghz quad core and Cortex™-A7 quad core CPUs
- Mali-T628 MP6 (OpenGL ES 3.0/2.0/1.1 and OpenCL 1.1 Full profile)
- 2Gbyte LPDDR3 RAM at 933MHz (14.9GB/s memory bandwidth) PoP stacked
- eMMC5.0 HS400 Flash Storage
- USB 3.0 Host x 1, USB 3.0 OTG x 1, USB 2.0 Host x 4
- HDMI 1.4a and DisplayPort1.1 for display
- Integrated power consumption monitoring tool

## Integrated power monitoring

The ODROID-XU3 has an integrated power analysis tool, with 4 current/voltage sensors measuring the power consumption of the Big A15 cores, Little A7 cores, GPUs and DRAMs individually. Professional developers can monitor CPU, GPU and DRAM power consumption using the included on-board power measurement circuit.

With the integrated power analysis tool, the XU3 can reduce the need for repeated trials when debugging with relation to power consumption, and users get the opportunity to enhance and optimize the performance of their CPU/GPU compute applications by keeping power consumption as low as possible.

Using the power analysis tool, fre-



```
543MHz 64°C
1000MHz, 20%
1000MHz, 5%
1000MHz, 6%
1000MHz, 4%
1600MHz, 0% 64°C
1600MHz, 0% 61°C
1600MHz, 0% 62°C
1600MHz, 81% 61°C
wer : 1.1V, 1.139A, 1.273W
wer : 1.0V, 0.159A, 0.171W
ower : 1.0V, 2.923A, 3.028W
ower : 1.2V, 0.446A, 0.544W
```

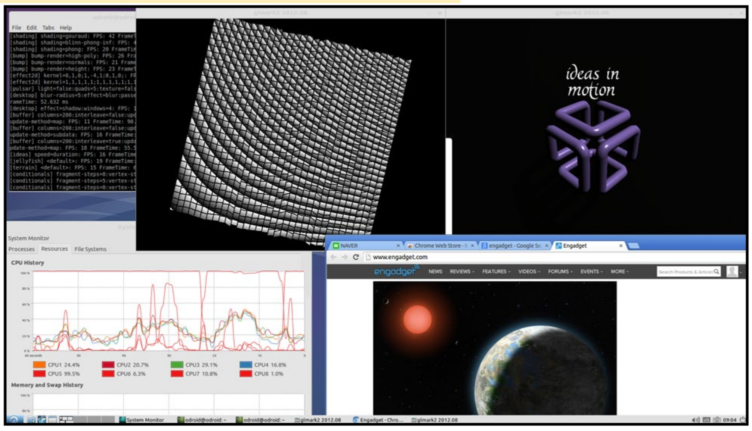
**The ODROID-XU3 running the Power Monitor application**

quency, voltage, amperage and power information shows as an on-screen overlay in the Android platform. You can monitor 4 big cores and GPU temperature as well, as shown in the screenshot.

## Heterogeneous Multi-Processing (HMP)

The ODROID-XU3 is equipped with four big cores (ARM® Cortex® -A15™ up to 2.0GHz) and four small cores (ARM® Cortex® -A7™ up to 1.4 GHz), providing improved processing capabilities while maintaining the most efficient power consumption imaginable. With the big.LITTLE™ HMP solution, Exynos-5422 can utilize a maximum of all eight cores to manage computationally intensive tasks.

**An ODROID-XU3 running Ubuntu 14.04 LTS and the latest version of Kernel 3.10. It's so fast!**



## OpenGL ES 3.0 and OpenCL 1.1

The ARM® Mali™-T628 MP6 GPU offers key API support OpenGL ES 1.1, OpenGL ES 2.0 and OpenGL ES 3.0, OpenCL 1.1 Full Profile and Google RenderScript. The Mali-T628 chip is the GPU of choice for use in the next generation of market-leading devices, optimized to bring breathtaking graphical displays to consumer applications such as 3D graphics, visual computing, augmented reality, procedural texture generation and voice recognition. You can download the full featured OpenGL ES and OpenCL SDK from the ARM Mali developer website at no charge.

## eMMC 5.0

eMMC uses intelligent flash memory technology that not only offers the capacity to store digital content, but also meets even stricter high sequential and random performance requirements to ensure a strong user experience. This enables fast OS booting, quick application launching, seamless multi-tasking, and quick access to the cloud.

In October 2013, JEDEC published the latest version of its popular eMMC standard called JESD84-B50: Embedded MultiMedia-Card, Electrical Standard (5.0). eMMC v5.0 defines several new functionalities and enhancements for embedded mass-storage flash memory widely used in smartphones and other mobile devices; and matches the challenging performance targets required by the next generation of mobile systems by introducing an HS400 mode that offers additional improvement in terms of interface speed (up to 400 MB/s vs 200 MB/s in the prior version). JESD84-B50 is available for free download from the JEDEC website at http://bit.ly/1uQKfZC.

For a demonstration of the XU3's capabilities, please watch the video at http://bit.ly/1CvJBWv.

## Specifications

### Processor

Samsung Exynos5422 ARM® Cortex™-A15 Quad 2.0GHz/ Cortex™-A7 Quad 1.4GHz

### Memory

2Gbyte LPDDR3 RAM PoP (933Mhz, 14.9GB/s memory bandwidth, 2x32bit bus)

### 3D Accelerator

Mali™-T628 MP6 OpenGL ES 3.0 / 2.0 / 1.1 and OpenCL 1.1 Full profile

### Energy Monitor

Measure the power consumption of big.LITTLE cores, GPU and DRAM

### Audio

On-board Audio codec / Standard 3.5mm headphone jack with HDMI Digital audio output
SPDIF optional USB optical output

### USB

USB 3.0 Host SuperSpeed USB standard A type connector x 1 port
USB 3.0 OTG SuperSpeed USB Micro A-B type connector x 1 port
USB 2.0 Host High Speed standard A type connector x 4 ports

**Display**
HDMI, DisplayPort

**Storage**
eMMC 5.0 Flash Storage (up to 64GB)
MicroSD Card Slot (up to 64GB)

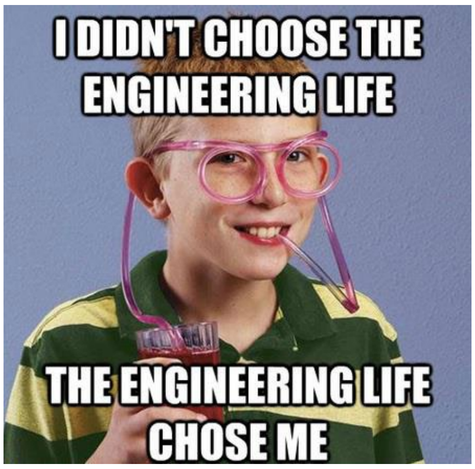
**LAN**
Fast Ethernet LAN 10/100Mbps Ethernet with RJ-45 Jack (Auto-MDIX support)
Gigabit Ethernet LAN (Option)
USB 3.0 to Gigabit Ethernet adapter (optional USB module)
WiFi USB IEEE 802.11b/g/n 1T1R WLAN with Antenna (optional USB module)

**Storage**
HDD/SSD optional SATA interface
SuperSpeed USB (USB 3.0) to Serial ATA3 adapter for 2.5"/3.5" HDD and SSD storage

**Power Supply (included)**
5V 4A Power

**System Software**
Ubuntu 14.04 + OpenGL ES + OpenCL on Kernel LTS 3.10
Android 4.4.2 on Kernel LTS 3.10
Full source code is accessible via our Github

**PCB**
Size: about 94 x 70 x 18 mm


I DIDN'T CHOOSE THE ENGINEERING LIFE
THE ENGINEERING LIFE CHOSE ME

# RETROARCH

## CONFIGURING XBOX 360 CONTROLLERS WITH RETROARCH V1.0.0.2+

by Rob Roy

In the most recent development versions of RetroArch for Android (1.0.0.2r34 and above), the way in which Xbox 360 controllers are configured has changed. Instead of using the options menu on the first screen, the setup is done inside the game itself using the RGUI interface. The Xbox 360 controller (wireless and wired) works natively with the Android operating system without additional drivers, but other USB controllers can also be connected with the same method.

To begin, download the latest development version of RetroArch from http://bit.ly/1uP6ejM. The APK works with any recent version of Android, including KitKat. Make sure that an Xbox 360 controller is connected and able to control the Android desktop, then launch RetroArch. If using the wireless version of the controller, use the button on the USB receiver to connect the joysticks first.

At the initial RetroArch options screen, select "Settings", click the "Input" tab, and make sure that the On-screen Overlay is enabled. Click the right mouse button and use the "Load Content (Detect Core)" option to start your favorite emulator.

Once inside the game, click the RetroArch symbol with the left mouse button, then press the "A" button on the dpad overlay. Use the dpad on the overlay to highlight "Settings", then press "A" again. Select "Input Options" and press "A" one more time, which displays the controller configuration menu.

Use the on-screen dpad to match the options shown in the screenshot, making sure to set "Analog D-pad Mode" to "Left Analog" so that the joystick movements are recognized. Finally, click on "Configure All" and press the requested buttons on the Xbox 360 controller. Repeat this process for each player's controller.

After the controllers are configured, press the right button (B) on the Xbox 360 controller. Select "Resume Content", use the mouse button to minimize the RetroArch overlay, and you're ready to play!

# ODROID SMART POWER

## USE AND PROTOCOL ANALYSIS

Edited by Venkat Bommakanti

One of the principal areas of embedded system development is System Power Requirement Analysis. Along with many validation tests, it is essential to ensure that the overall system is performing within the design parameters, power wise. The ODROID Smart Power solution is an ideal tool for this purpose. It is essentially an adjustable smart power supply that can periodically collect/display/forward voltage, current and power load of the system, for analysis and energy consumption optimization.

This article walks you through, the use of this solution (Hardkernel-developed hardware & software), both under Windows (7+) and Lubuntu (3.8.13 kernel), and

the use of the popular open source protocol analyzer Wireshark (and the command line equivalent Tshark) software with USB protocol analysis module, specifically Lubuntu. Sniffing the USB data traffic is useful to study the communication protocol, enhance the firmware and debug issues, if need be. Several protocol analysis tools are also available for Windows.

### Requirements

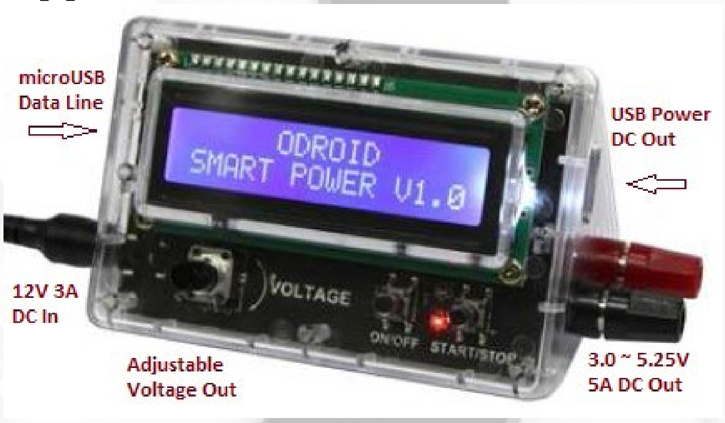
**1. The entire Smart Power v1.0 package.**
**2. An ODROID single board computer, such as a U3, whose power requirements are to be analyzed. The power supply and cables provided with the Smart Power package should be sufficient to drive the U3.**
**3. An adapter may be required to use the device in your specific region.**
**4. A bootable 8+ GB MicroSD card or eMMC module containing the latest Lubuntu image available from the Hardkernel website at http://www.hardkernel.com.**
**5. A microUSB to USB cable for data transfer between the U3 and the host computer.**
**6. A host computer to gather the power data transmitted by the Smart Power, such as a Windows 7+ PC. The host computer can also be an ODROID-U3 (possibly even the one being analyzed) running Lubuntu.**
**7. A compatible HDMI monitor to be used with U3 or VNC access to the U3 via utilities like the TightVNC vnc-viewer from the host PC.**
**8. Wireshark and tshark Ver. 1.10+ software, available for both Ubuntu and Debian operating systems**
**9. SmartPower Ver. 1.1.0 monitoring application source code from Hardkernel**
**10. MinGW 0.6., QT 4.8.6, and Qwt 6.1.0 library for Windows (if using a Windows host machine)**

### Device setup

Ensure that the main power is stabilized and surge-protected, and that all devices are grounded properly. Connect the host PC or U3 to a functional wired network that can access the Internet. Attach the provided 12V 3A power supply to the Smart Power and turn on the device. Wait for the display to show the power parameters such as the voltage. Adjust the output voltage regulator to ~5.01V (slightly higher than 5V), since we will be examining the power requirements of a U3.

Attach the microUSB data cable to the host PC or the U3 itself, then attach the HDMI display to the U3, if available. Connect the exposed red and black terminals of the DC plug cable (2.5mm/0.8mm for U3) to the Smart Power device and the power jack to the barrel on the U3. Turn on the AC power and ensure the U3 goes through its boot-up process properly. If a dedicated HDMI monitor is unavailable, access the U3 via the vnc-viewer or SSH to examine its progress.

### Monitoring Application for Windows

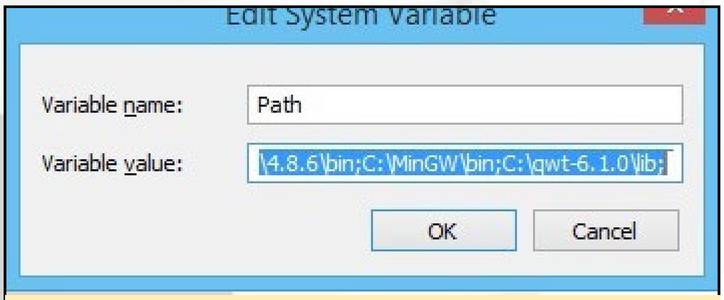


Hardkernel has developed an open source monitoring application called SmartPower that works with the Smart Power device. Although they have provided a pre-built executable binary, I'll describe the process of building the application from its source code. Follow the list of steps to create the build environment and build the monitoring application.

MinGW is a minimalist (GNU) development environment for native Windows applications. Access the http://www.mingw.org/ website and click on the Download Installer button on the top right of the webpage to download and run the mingw-get-setup.exe utility. Select the default options where applicable, and MinGW will be installed to C:\MinGW. Finally, add C:\MinGW\bin to the environment PATH variable, which is the location of the mingw32-make.exe tool.

Qt is a cross-platform application and UI framework for developing C++ applications. Download the latest 4.8.x version from http://bit.ly/1ru4Jsk. Install it to C:\Qt and add C:\Qt\4.8.4\bin to the environment PATH variable so that the qmake.exe tool may be found.

The Qwt library contains GUI Components and utility classes which are primarily useful for programs presenting technical data. The latest 6.1.0 Version can be downloaded from http://bit.ly/1quAoaY and extracted to C:\qwt-6.1.0. Add C:\qwt-6.1.0\lib to the environment PATH variable, which should match the PATH variables shown in the screenshot.


Edit System Variable

Variable name: Path

Variable value: \4.8.6\bin;C:\MinGW\bin;C:\qwt-6.1.0\lib;

**Win 8.1 environment PATH settings**

Build qwt-6.1.0 from within a new cmd instance, using the following commands:

```
> cd C:\qwt-6.1.0
> qmake
> make
> make install
> qmake -set QMAKEFEATURES C:\qwt-6.1.0\features
```

Download the SmartPower monitoring PC Application source code from http://bit.ly/1DKSGw0. Extract it to C:\smartpower_source, then build it from within a new cmd instance, using the following commands:

```
> cd smartpower_source\HIDAPI
> qmake
> make -f MakeFile.Release
> cd ..\smartpower_source\smartpower
> qmake
> make -f MakeFile.Release
```

After a successful build, the monitoring application that was just created can be found at the following location:

```
C:\smartpower_source\smartpower\windows\SmartPower.
exe
```

Since the Smart Power device has already been set up, this monitoring application can now be started. Notice the locations in the Ampere View screenshot where the initialization status and firmware version are displayed. Checking the log check box will start the logging of the data being captured.

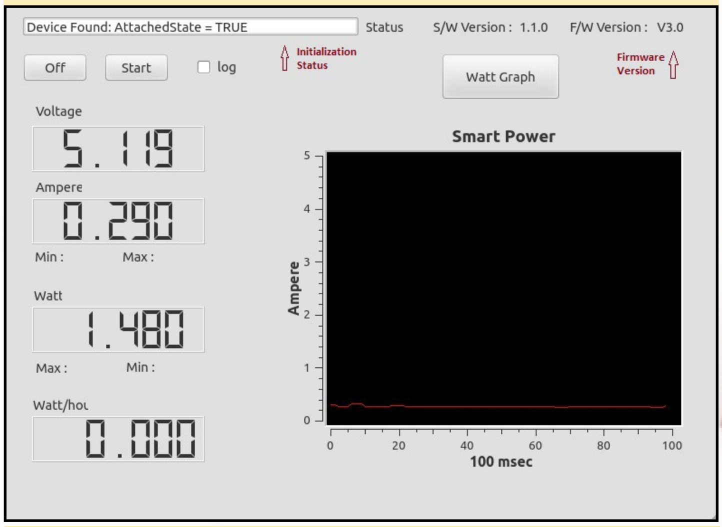
Clicking the Watt Graph button takes you to the Watt view, as shown in the screenshot.

When an error situation is encountered during initial setup, you will see a status message indicating why the error occurred:

**- The USB data cable between Smart Power and host computer is disconnected**

**- Using a defective USB data cable**

**Watt view**


Device Found: AttachedState = TRUE

Status S/W Version : 1.1.0 F/W Version : V3.0

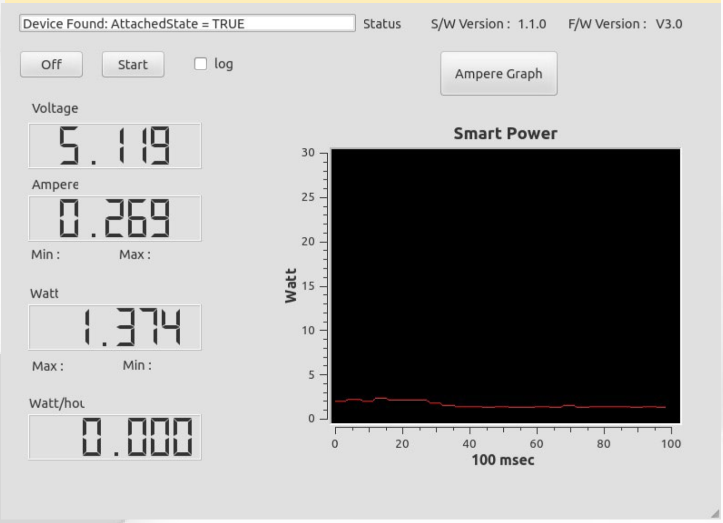
Voltage 5.119

Ampere 0.290

Watt 1.480

Watt/hou 0.000

**Ampere view**


Device Found: AttachedState = TRUE

Status S/W Version : 1.1.0 F/W Version : V3.0

Voltage 5.119

Ampere 0.269

Watt 1.374

Watt/hou 0.000

**- The USB port on the host computer or Smart Power is not functioning properly**

**- Running defective firmware or OS.**

The use of Wireshark like utilities (discussed later) may help narrow down some of these types of error causes.

## Monitoring application for Lubuntu

The SmartPower monitoring application needs an appropriate build environment, which can be installed by running the following command (one line) in a terminal window:

```
$ sudo apt-get install qt4-default qt4-designer
libqwt-dev libusb-1.0-0-dev
```

Download and unpack the SmartPower source code using the following commands:

```
$ cd ~ && mkdir src && cd src && mkdir sp && cd sp
$ mv ~/Downloads/smartpower_source.zip .
$ unzip smartpower_source.zip
$ cd smartpower_source
```

The versions of the installed build tools can then be checked by typing the following:

```
$ uic -version
Qt User Interface Compiler version 4.8.6

$ qmake -version
QMake version 2.01a
Using Qt version 4.8.6 in /usr/lib/arm-linux-gnueabi-
hf

$ make -version

GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying
conditions.
There is NO warranty; not even for MERCHANTABILITY or
FITNESS FOR A
PARTICULAR PURPOSE.

This program built for arm-unknown-linux-gnueabihf
```

The SmartPower application can then be built:

```
$ cd HIDAPI
$ qmake
$ make
$ cd ../smartpower
$ uic smartpower.ui > ui_smartpower.h
$ qmake
$ make
```

Create the relevant udev file using the following command and values.  Make sure to follow the hints in the comments.

```
$ sudo vi /etc/udev/rules.d/99-hiid.rules

# This is a sample udev file for HIDAPI devices which
# changes the permissions
# to 0666 (world readable/writable) for a specified
# device on Linux systems.
# If you are using the libusb implementation of hi
# dapi (hid-libusb.c), then
# use something like the following line, substituting
# the VID and PID with
# those of your device.  Note that for kernels before
# 2.6.24, you will need
# to substitute "usb" with "usb_device".  It
# shouldn't hurt to use two lines
# (one each way) for compatibility with older
# systems.
# HIDAPI/libusb
# SUBSYSTEM=="usb", ATTRS{idVendor}=="04d8",
# ATTRS{idProduct}=="003f", MODE="0666"

# If you are using the hidraw implementation, then do
# something like the
# following, substituting the VID and PID with your
# device.  Busnum 1 is USB.
# HIDAPI/hidraw
KERNEL=="hidraw*", ATTRS{busnum}=="1",
ATTRS{idVendor}=="04d8", ATTRS{idProduct}=="003f",
MODE="0666"

# Once done, _optionally_ rename this file for your
# device, and drop it into
# /etc/udev/rules.d and unplug and re-plug your
# device.  This is all that is
# necessary to see the new permissions.  Udev does
# not have to be restarted.

# Note that the hexadecimal values for VID and PID
# are case sensitive and
# must be lowercase.

# If you think permissions of 0666 are too loose,
then see:
```

```
# http://reactivated.net/writing_udev_rules.html for
# more information on finer
# grained permission setting.  For example, it might
# be sufficient to just
# set the group or user owner for specific devices
# (for example the plugdev
# group on some systems).
```

Change the privileges of this rules file using the command:

```
$ sudo chmod 0666 /etc/udev/rules.d/99-hiid.rules
```

After rebooting the system, the SmartPower monitoring application can be run using the commands:

```
$ cd ~/src/sp/smartpower_source/smartpower/linux/
SmartPower
```

The monitoring tool for Lubuntu has an user-interface identical to that of the Windows version. The images shown for the Windows example are applicable in this case too.

The USB device information and details of the Smart Power device can be reported using these commands (presuming bus 1, device 19):

```
$ odroid@u3-2:/etc/udev/rules.d$ lsusb
Bus 001 Device 019: ID 04d8:003f Microchip Technol-
ogy, Inc.

# Use Bus/Device information from previous command
$ lsusb -D /dev/bus/usb/001/019 output:
Device: ID 04d8:003f Microchip Technology, Inc.
Device Descriptor:
  bLength           18
  bDescriptorType           1
  bcdUSB            2.00
  bDeviceClass              0 (Defined at Interface
level)
  bDeviceSubClass           0
  bDeviceProtocol           0
  bMaxPacketSize0           8
  idVendor          0x04d8 Microchip Technology,
Inc.
  idProduct         0x003f
  bcdDevice         0.02
  iManufacturer             1 Microchip Technology
Inc.
  iProduct                  2 Simple HID Device Demo
  iSerial                   0
  bNumConfigurations  1
  Configuration Descriptor:
       bLength                9
       bDescriptorType        2
       wTotalLength           41
       bNumInterfaces         1
       bConfigurationValue    1
       iConfiguration         0
       bmAttributes           0xc0
       Self Powered
       MaxPower               100mA
       Interface Descriptor:
       bLength                9
       bDescriptorType        4
       bInterfaceNumber       0
       bAlternateSetting      0
       bNumEndpoints          2
       bInterfaceClass        3 Human Interface Device
       bInterfaceSubClass     0 No Subclass
bInterfaceProtocol    0 None
iInterface            0
HID Device Descriptor:
bLength               9
bDescriptorType       33
bcdHID                1.11
bCountryCode          0 Not supported
bNumDescriptors       1
bDescriptorType       34 Report
wDescriptorLength     28
Report Descriptors:
** UNAVAILABLE **
Endpoint Descriptor:
bLength               7
bDescriptorType       5
bEndpointAddress      0x81  EP 1 IN
bmAttributes          3
Transfer Type         Interrupt
Synch Type            None
Usage Type            Data
wMaxPacketSize        0x0040  1x 64 bytes
bInterval             1
Endpoint Descriptor:
bLength               7
bDescriptorType       5
bEndpointAddress      0x01  EP 1 OUT
bmAttributes          3
Transfer Type         Interrupt
Synch Type            None
Usage Type            Data
wMaxPacketSize        0x0040  1x 64 bytes
bInterval             1
Device Status:        0x0001
Self Powered
```

## Wireshark installation

The prebuilt wireshark and the tshark command-line packages can be installed by typing the following commands, then rebooting the system:

```
$ cd ~/
$ sudo apt-get install build-dep wireshark
$ sudo dpkg-reconfigure wireshark-common
$ sudo apt-get install tshark
```

The installed version of these utilities should be checked after rebooting:

```
$ wireshark --version
wireshark 1.10.6 (v1.10.6 from master-1.10)

Copyright 1998-2014 Gerald Combs <gerald@wireshark.
org> and contributors.
This is free software; see the source for copying
conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE.

Compiled (32-bit) with GTK+ 3.10.7, with Cairo
1.13.1, with Pango 1.36.1, with
GLib 2.39.91, with libpcap, with libz 1.2.8, with
POSIX capabilities (Linux),
without libnl, with SMI 0.4.8, with c-ares 1.10.0,
with Lua 5.2, without Python,
with GnuTLS 2.12.23, with Gcrypt 1.5.3, with MIT Ker-
beros, with GeoIP, with
PortAudio V19-devel (built Feb 25 2014 21:10:47),
with AirPcap.

Running on Linux 3.8.13.27, with locale en_US.UTF-8,
with libpcap version 1.5.3,
with libz 1.2.8, GnuTLS 2.12.23, Gcrypt 1.5.3, with-
out AirPcap.

Built using gcc 4.8.2.

$ tshark --version
TShark 1.10.6 (v1.10.6 from master-1.10)

Copyright 1998-2014 Gerald Combs <gerald@wireshark.
org> and contributors.
This is free software; see the source for copying
conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE.

Compiled (32-bit) with GLib 2.39.91, with libpcap,
with libz 1.2.8, with POSIX
capabilities (Linux), without libnl, with SMI 0.4.8,
with c-ares 1.10.0, with
Lua 5.2, without Python, with GnuTLS 2.12.23, with
Gcrypt 1.5.3, with MIT
Kerberos, with GeoIP.

Running on Linux 3.8.13.27, with locale en_US.UTF-8,
with libpcap version 1.5.3,
with libz 1.2.8.

Built using gcc 4.8.2.
```

For the selected Lubuntu version, usbmon driver is already present, and can be explicitly loaded and checked:

```
$ sudo mount -t debugfs none_debugs /sys/kernel/debug
mount: none_debugs already mounted or /sys/kernel/
debug busy
mount: according to mtab, none is already mounted on
/sys/kernel/debug

$ sudo modprobe usbmon

$ sudo ls /sys/kernel/debug/usb/usbmon
0s  0u   1s  1t   1u  2s   2t  2u
```

## Access privilege

Access to usbmon devices is typically possible only for superusers such as root.  However, it is not advisable to run protocol analyzers using root privileges.  To ensure safe access of usbmon interfaces to wireshark and tshark, one should give the user the proper privileges using the following commands:

```
$ sudo addgroup -system wireshark
$ sudo usermod -a -G wireshark odroid

$ sudo chgrp wireshark /dev/usbmon*
$ sudo ls -lsa /dev/usbmon*
0 crw------- 1 root wireshark 248,   0 sep 14 15:23 /
dev/usbmon0
0 crw------- 1 root wireshark 248,   1 sep 14 15:23 /
dev/usbmon1
0 crw------- 1 root wireshark 248,   2 sep 14 15:23 /
dev/usbmon2

$ sudo chmod g+r /dev/usbmon*
$ sudo ls -lsa /dev/usbmon*
0 crw-r----- 1 root wireshark 248,   0 sep 14 15:23 /
dev/usbmon0
```

```
0 crw-r----- 1 root wireshark 248,   1 sep 14 15:23 /
dev/usbmon1
0 crw-r----- 1 root wireshark 248,   2 sep 14 15:23 /
dev/usbmon2
```

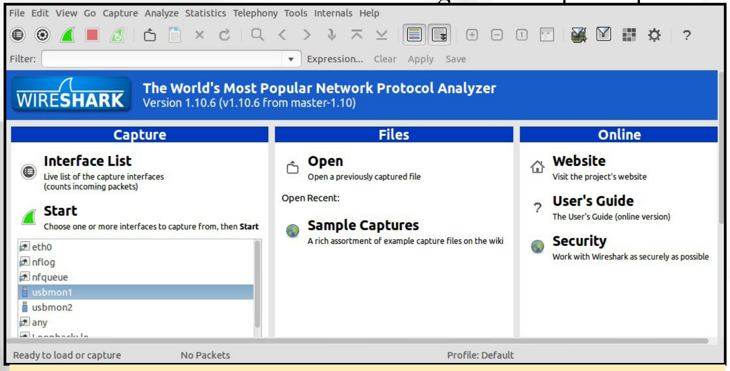
## Wireshark usage

The wireshark utility can be directly launched (without sudo) using the following command to display the usbmon interfaces as shown below.

```
$ wireshark
```

To ascertain the usbmon interface associated with the Smart Power device, click on the Interface List option under the Capture section of the wireshark main screen. One can immediately observe that one of the usbmon interfaces shows a high transfer (USB) rate. As can be seen in the screenshot, this happens to be the usbmon1 interface, which is associated with the Smart Power device (presuming that no other highly active USB device is attached to the host computer).

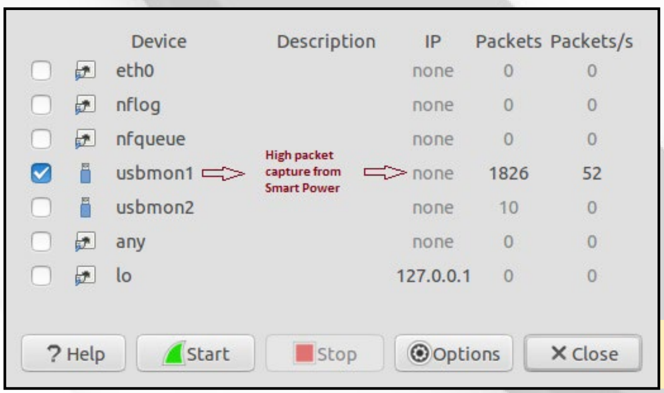
Click on the Start button to begin the capture process.


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
WIRESHARK
The World's Most Popular Network Protocol Analyzer
Version 1.10.6 (v1.10.6 from master-1.10)
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
eth0
nflog
nfqueue
usbmon1
usbmon2
any
Files
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files on the wiki
Online
Website
Visit the project's website
User's Guide
The User's Guide (online version)
Security
Work with Wireshark as securely as possible
Ready to load or capture
No Packets
Profile: Default

Wireshark main screen

Note that the high capture process can very quickly result in a very large output file. Scroll through the top section, where one line corresponds to one data-set capture. As you scroll, you can see that some of the lines correspond to the Volts, Amps, Watts and Watt-Hours data.

After about 100 data points, you can stop the capture process and save the data to a file for continued future analysis. Further study will give details about the protocol. This knowledge can be useful to understand the functioning of the Smart Power device and debug firmware updates if need be.

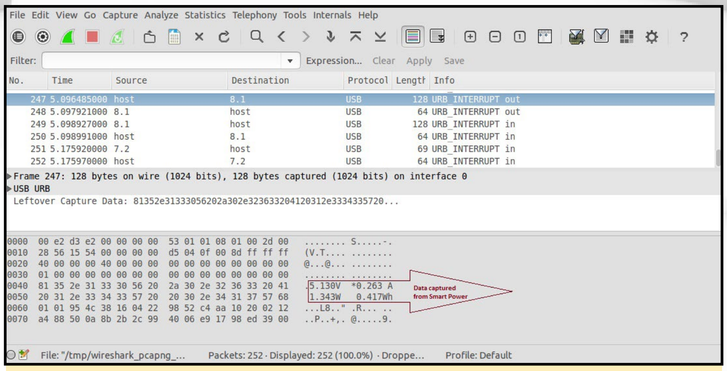


| | Device | Description | IP | Packets | Packets/s |
|---|---|---|---|---|---|
| ☐ | eth0 | | none | 0 | 0 |
| ☐ | nflog | | none | 0 | 0 |
| ☐ | nfqueue | | none | 0 | 0 |
| ✓ | usbmon1 | | none | 1826 | 52 |
| ☐ | usbmon2 | | none | 10 | 0 |
| ☐ | any | | none | 0 | 0 |
| ☐ | lo | | 127.0.0.1 | 0 | 0 |

Wireshark interfaces

## Tshark

The command-line equivalent of wireshark, called tshark, can be invoked using the command:


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
Data captured from Smart Power

```
No.  Time         Source      Destination  Protocol Length Info
 247 5.096485000 host        8.1          USB       128 URB_INTERRUPT out
 248 5.097921000 8.1         host         USB        64 URB_INTERRUPT out
 249 5.098927000 8.1         host         USB       128 URB_INTERRUPT in
 250 5.098991000 host        8.1          USB        64 URB_INTERRUPT in
 251 5.175920000 7.2         host         USB        69 URB_INTERRUPT in
 252 5.175970000 host        7.2          USB        64 URB_INTERRUPT in
▶Frame 247: 128 bytes on wire (1024 bits), 128 bytes captured (1024 bits) on interface 0
▶USB URB
 Leftover Capture Data: 81352e31333056202a302e323633204120312e3334335720...

0000  00 e2 d3 e2 00 00 00 00  53 01 01 08 01 00 2d 00   ........ S.....-.
0010  28 56 15 54 00 00 00 00  d5 04 0f 00 8d ff ff ff   (V.T.... ........
0020  40 00 00 00 40 00 00 00  00 00 00 00 00 00 00 00   @...@... ........
0030  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0040  81 35 2e 31 33 30 56 20  2a 30 2e 32 36 33 20 41   .5.130V  *0.263 A
0050  20 31 2e 33 34 33 57 20  20 30 2e 34 31 37 57 68    1.343W   0.417Wh
0060  01 01 95 4c 38 16 04 22  98 52 c4 aa 10 20 02 12   ...L8.." .R... ..
0070  a4 88 50 0a 8b 2b 2c 99  40 06 e9 17 98 ed 39 00   ..P..+,. @.....9.

File: "/tmp/wireshark_pcapng_...   Packets: 252 · Displayed: 252 (100.0%) · Droppe...   Profile: Default
```

Wireshark with data captured from Smart Power

```
$ tshark -D
1.  eth0
2.  nflog
3.  nfqueue
4.  usbmon1
5.  usbmon2
6.  any
7.  lo (Loopback)
```

Note that the sudo based invocation was not required to display the usbmon interfaces, due to the proper access privilege setup.

Tshark can be used to capture data via the usbmon2 interface (see wireshark notes above) using the following command so that the 1.pcap file can be viewed in wireshark directly.

```
$ tshark -i usbmon2 -w 1.pcap
```

## Other monitoring applications

Community member @muehlbau has provided a useful command-line utility to monitor the Smart Power device, which can be installed by typing the following series of commands into a Terminal window:

```
$ sudo apt-get install build-dep pkg-config libusb-
1.0-0-dev
$ cd ~/src && mkdir mu && cd mu
$ git clone https://github.com/muehlbau/odroid-smart-
power-linux
```

Build the utility:

```
$ cd odroid-smartpower-linux
$ make
```

It can then be launched using the smartpower command (which requires root privileges) and the results are written to a log file:

```
$ sudo ./smartpower capture-logfile
```

Community member @pcat provides yet another useful command-line tool to monitor the Smart Power device. The source code for it can be obtained and built from github using the commands:

```
$ cd ~/src && mkdir sp-cl1 && cd sp-cl1
$ git clone https://github.com/polarcat/smartpower
$ cd smartpower
$ make
```

It is beneficial to detect the hidraw devices using the command:

```
$ sudo find /dev/hid*
/dev/hidraw0
/dev/hidraw1
/dev/hidraw2
/dev/hidraw3
```

The above command can run with or without the Smart Power device attached. In the test here, indicated that hidraw3 corresponds to the Smart Power device. It can be checked with the following command:

```
odroid@u3-2:~/src/sp-cl1/smartpower$ cat /sys/class/
hidraw/hidraw3/device/uevent
DRIVER=hid-generic
HID_ID=0003:000004D8:0000003F
HID_NAME=Microchip Technology Inc.  Simple HID Device
Demo
HID_PHYS=usb-s5p-ehci-3.2.6/input0
HID_UNIQ=
MODALIAS=hid:b0003g0001v000004D8p0000003F
```

The generated smartpower binary can be launched using one of the commands to observe the periodically captured output:

```
$ cd ~/src/sp-cl1/smartpower
$ sudo ./smartpower
$ sudo ./smartpower -v -d /dev/hidraw3
```

Options can be listed using the help command:

```
$ sudo ./smartpower --help
```

To address an output delimiter bug and some other minor output formatting issues, I created a patch so that the Smart Power is autodetected, eliminating the need for the hidraw device-path (-d option) specification. The patch, which lists the changes to smartpower.c, is listed below:

```
85d85
< static char detected_dev[32] = {'\0', };
97c96
<               printf("0.%06u%c%s\n", 0, sep, data);
---
>               printf("0.0%c%s\n", sep, data);
169c168
<       printf("Version: %s\n\n", buf);
---
>       printf("Version: %s\n", buf);
188,191d187
<       if (errno == 22) {
<               printf("(=) Check invocation
syntax\n");
<               exit(errno);
<       }
294a289
>       i++;
302,303c298
<               i++;
<              snprintf(detected_dev, sizeof(detected_
dev), "%s", name);
---
>               fprintf(stderr, "Detected smartp at
%s\n", name);
309,312c304,305
<      if (i == 0) {
<       printf("(=) Smart Power device is disconnect-
ed.  Please connect and retry\n");
<       exit(1);
<              }
---
>      if (i == 0)
>       printf("(=) smart power device is not
connected\n");
371c364
<      printf("  -d, --dev <dev>     path to hidraw
device node (/dev/hid*) \n");
---
>      printf("  -d, --dev <dev>     path to hidraw
device node\n");
387,389d380
<      csv = 0;
<     sep = ' ';
<
416,418c406,410
<               continue;
<      }
<               if (opt(arg, "-c", "--csv")) {
---
>               csv = 0;
>               sep = ' ';
>               continue;
>      }
>      if (opt(arg, "-c", "--csv")) {
430,438c422,430
<     fd = smartp_probe();
<     if (dev) {
<      fd = smartp_open(dev);
<      if (fd < 0)
<               return fd;
<     }
<
<     if (verbose == 1) {
<      printf("Detected dev: [%s], requested dev:
[%s]\n", detected_dev, dev);
---
>     if (dev)
>      fd = smartp_open(dev);
>     else
>      fd = smartp_probe();
>
>     if (fd < 0)
>      return fd;
>
>     if (verbose == 1) {
```

For additional information or questions, please visit the original information sources at:

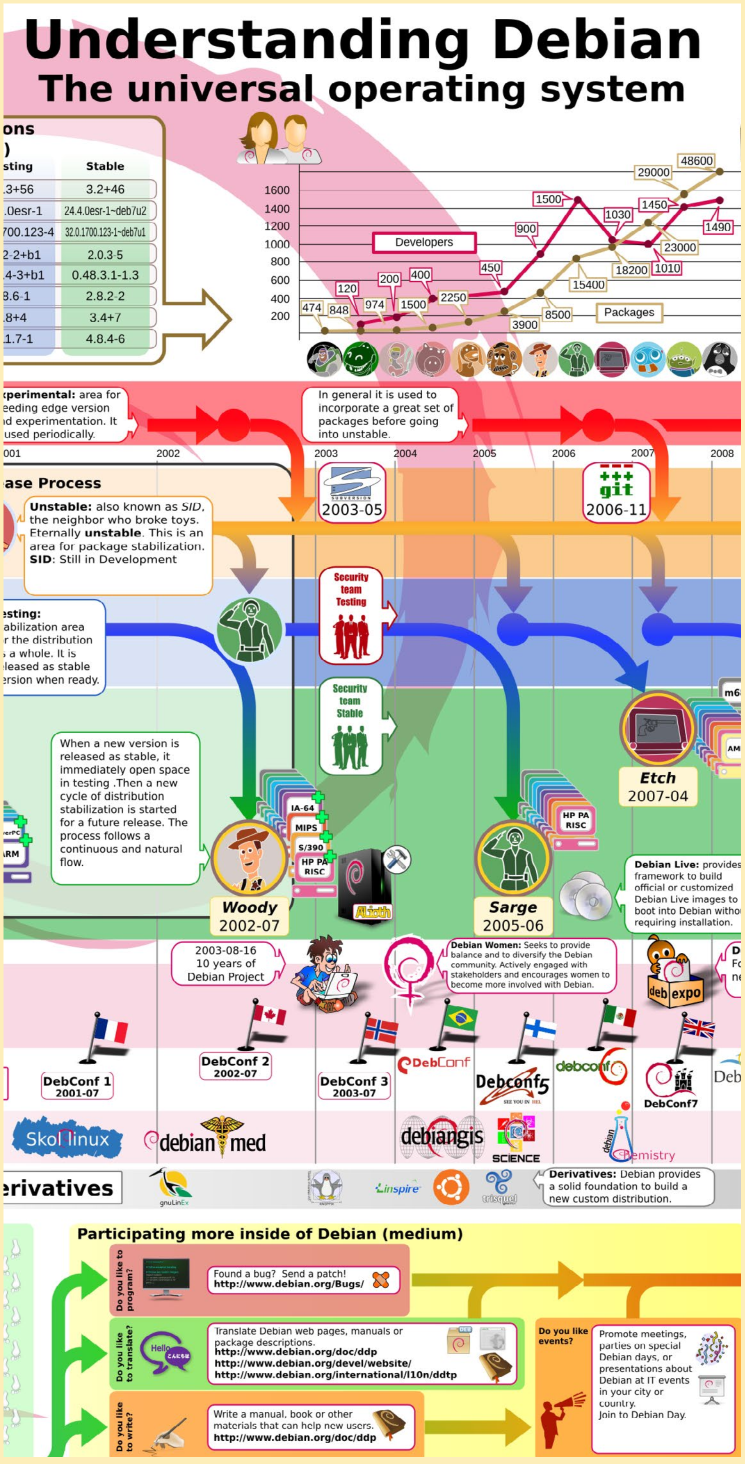
```
http://bit.ly/1vpEdkh
http://bit.ly/1oomVis
http://bit.ly/1uwjVaP
http://www.mingw.org
http://www.qt-project.org
http://qwt.sourceforge.net
http://code.wireshark.org
http://bit.ly/1BriMjG
http://bit.ly/1lh6G8v
```

# *ALL ABOUT DEBIAN*

## *AN EPIC INFOGRAPHIC*

**edited by Nicole Scott**

Check out an amazing infographic of the inner workings of the Debian operating system at http://bit.ly/1vJHXgB. It illustrates everything about the development process, as well as how to get involved in the Debian project yourself!


Understanding Debian
The universal operating system
Developers
Packages
Stable
Experimental: area for bleeding edge version and experimentation. It is used periodically.
In general it is used to incorporate a great set of packages before going into unstable.
Release Process
Unstable: also known as SID, the neighbor who broke toys. Eternally unstable. This is an area for package stabilization. SID: Still in Development
Testing: stabilization area for the distribution as a whole. It is released as stable version when ready.
When a new version is released as stable, it immediately open space in testing. Then a new cycle of distribution stabilization is started for a future release. The process follows a continuous and natural flow.
Security team Testing
Security team Stable
Woody 2002-07
Sarge 2005-06
Etch 2007-04
2003-05
2006-11
2003-08-16 10 years of Debian Project
Debian Women: Seeks to provide balance and to diversify the Debian community. Actively engaged with stakeholders and encourages women to become more involved with Debian.
Debian Live: provides framework to build official or customized Debian Live images to boot into Debian without requiring installation.
DebConf 1 2001-07
DebConf 2 2002-07
DebConf 3 2003-07
DebConf7
Derivatives
Derivatives: Debian provides a solid foundation to build a new custom distribution.
Participating more inside of Debian (medium)
Do you like to program?
Found a bug? Send a patch! http://www.debian.org/Bugs/
Do you like to translate?
Translate Debian web pages, manuals or package descriptions. http://www.debian.org/doc/ddp http://www.debian.org/devel/website/ http://www.debian.org/international/l10n/ddtp
Do you like to write?
Write a manual, book or other materials that can help new users. http://www.debian.org/doc/ddp
Do you like events?
Promote meetings, parties on special Debian days, or presentations about Debian at IT events in your city or country. Join to Debian Day.

# TUNE YOUR LINUX DESKTOP TO MONITOR PERFORMANCE AND WEATHER

## USING CONKY AND HARMATTAN

by Jussi Opas

It's important for many applications to monitor system resources such as processor usage, disk activity, storage space, and network traffic. The latter is especially important when large files are being transferred, such as downloading an operating system image or system update. Although the Gnome System Monitor can be used for this purpose, if one wants to measure statistics on network traffic on both wireless and wired network individually, it is not possible. The System Monitor consumes computation resources, so it would be good to have a lightweight, more configurable alternative, such as Conky. We also may want to measure ODROID-specific characteristics such as big.LITTLE cluster usage. In this article we describe our experience with using Conky on Linux with both an ODROID-XU and ODROID-U3 computers.

## Introduction

Conky has been around already for several years, as can be seen by reviewing the development logs at http://bit.ly/1mk0gHu. One can find many screenshots on the Internet where Conky is included as part of the desktop. In our experiments, we tuned performance and weather monitoring with Conky on two ODROIDs and three operating systems, as shown in the following table:

| OS | version | ODROID |
|---|---|---|
| Xubunu | 13.10 | XU |
| Debian | 7, Ezy Wheezy | U3 |
| Lubuntu | 14.04 | U3 |

## Installation

Conky can be downloaded and launched by typing the following command into a Terminal window:

```
$ sudo apt-get install conky &&
conky
```

After the program loads, the default Conky monitor is shown, where one can define what to show and how items to be monitored are shown. The configuration file is saved to the home directory at ~/.conkyrc. To find out more about .conkyrc, visit http://bit.ly/1CSX6Qd.

## Performance monitor

When we watchthe CPU in an ODROID, we are interested in frequency, frequency scaling governor, temperature, cluster in use, and utilization. The scaling governor can be fetched easily from the kernel files. The native freq_g conky function is used to get the current frequency, although it could also be fetched directly from file.

To show temperature at the correct magnitude, we need to divide the result by 1000. We can use the binary calculator bc to do the division:

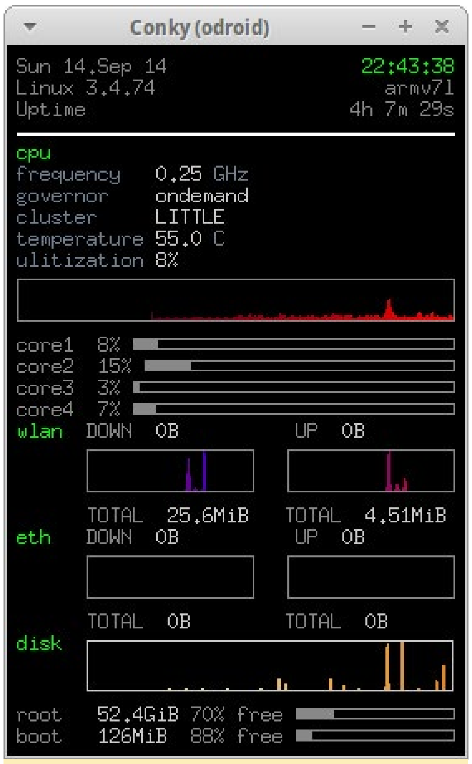
```
${exec echo "scale=1; $(cat /
sys/class/thermal/thermal_zone0/
temp)/1000" | bc}
```

We had to install `bc` in Debian separately with apt-get, since it wasn't installed by default. As an alternative, one could also use `awk` to divide by 1000 to get te correct temperature.

For the ODROID, we added our special definitions to display the utilization of all 4 cores. Each core utilization can be shown separately as a bar with the following tweak:

```
core1  ${cpu cpu0}% ${cpubar
cpu0}
```

With the XU especially, we also want to know which core cluster is in use (big



```
Conky (odroid)
Sun 14.Sep 14                22:43:38
Linux 3.4.74                   armv7l
Uptime                      4h 7m 29s
cpu
frequency   0.25 GHz
governor    ondemand
cluster     LITTLE
temperature 55.0 C
ulitization 8%
core1  8%
core2  15%
core3  3%
core4  7%
wlan   DOWN  0B          UP   0B
       TOTAL  25.6MiB    TOTAL  4.51MiB
eth    DOWN  0B          UP   0B
       TOTAL  0B         TOTAL  0B
disk
root    52.4GiB 70% free
boot    126MiB  88% free
```

**Conky monitor displaying CPU, temperature, and network information**

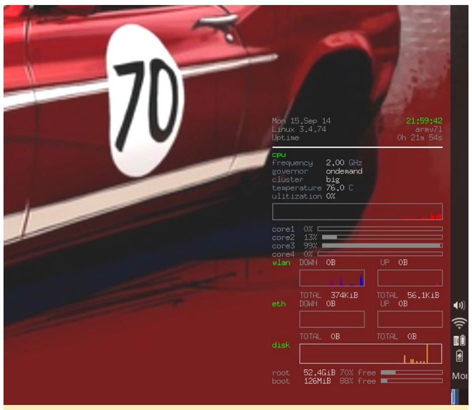
or LITTLE). The following instruction decides which core cluster is in use:

```
${if_match ${exec cat /dev/bL_
status | grep A7 | cut -c18} < 1}
big${else}LITTLE${endif}
```

The sample also shows, how to express an if-then statement in a conky definition file. The caveat of this definition is that conky must be invoked with sudo, because the /dev/bL_status file can not be accessed without root privileges.

Conky itself offers functions such as downspeedgraph for showing separately Ethernet or wireless traffic of downlink and of uplink. It is also possible to show aggregated traffic. We chose to show eth and wlan traffic separately. When one configures custom definitions, it is useful to look up the names that must be used in the definition file with the `ifconfig` command (for instance, eth0 and wlan6).

The background image is seen through the Conky monitor, since the own_window has been written into the .conkyrc parameters file. If the moni-



**Monitoring Conky on Xubuntu image on ODOID XU with the big cluster in use**

tor is defined to be transparent, then the wallpaper defines whether the used font is readable and distinguishable. To reasonably change used colors, they can be defined in conky configuration file as follows

```
color8 888888
```

Because color may vary in different locations in wallpaper, this does not still guarantee that text would always be easily readable. Instead, when an own window is used, then its defined background is always same and the content of the monitor is always visible as well.

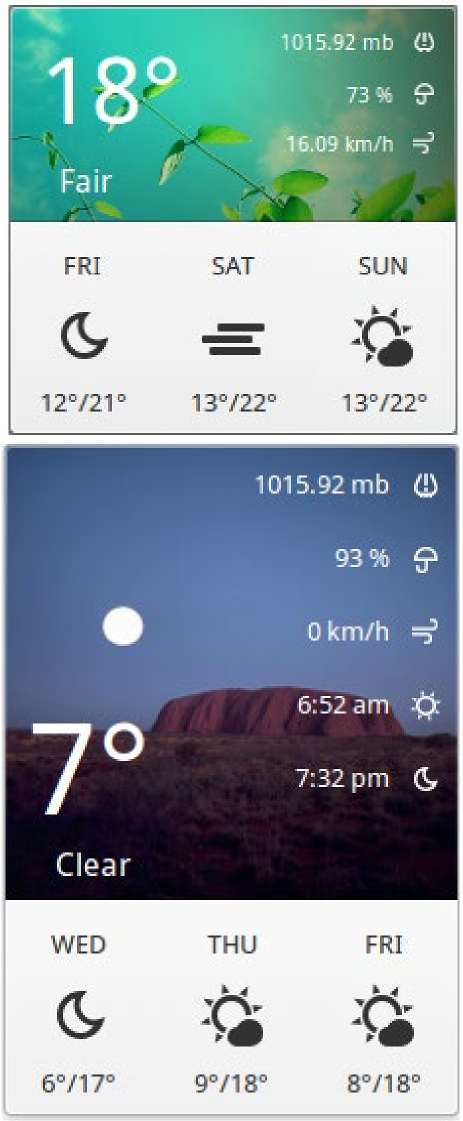
## Weather Monitor

On the Internet, there are plenty of examples of how to use Conky as part of a desktop. Often, there is also current weather or some weather forecast shown. We experimented with Conky Harmattan, and tuned it to work on Xubuntu, Lubuntu and Debian images.

For variety, the Harmattan Conky Pack offers 15 different flavors with various modes, which can be downloaded from http://bit.ly/1rrxV20.

The Harmattan weather monitors come in various sizes, and the background weather image changes dynamically by temperature and weather type. If something goes wrong, or one can not get Harmattan to work with the first attempt, it is better to uninstall and install again.

During the first time installation,



**Example Conky Harmattan configurations**

there is a tricky part, which requires looking up the weather code for the desired city. This involves visiting a Yahoo URL and copying the city code.

One must first go to http://weather.yahoo.com, search for the city, then copy the code from the address bar. For instance, the following address contains the number 44418, which is the code required for Harmattan.

```
https://weather.yahoo.com/united-
kingdom/england/london-44418/
```

The other tricky part in configuring Harmattan is how to make the widget appropriately colored and visible in each Linux distribution. As we had no explicit instructions, we used a trial-and-error method to find our own configuration.

The common tweaks that seems to work on all platforms are show in the following list:

```
double_buffer yes
update_interval 5
own_window yes
own_window_transparent yes
```

The double_buffer option is used to reduce flicker, and we tuned the update interval to be 5 seconds. In Xubuntu we also used:

```
own_window_type override
```

For Debian and Lubuntu, the specialized additional definition is:

```
own_window_hints
undecorate,sticky,skip_
taskbar,skip_pager,below
```

With these configurations, we achieved a stable and well behaving Harmattan, as shown in the screenshot from Debian.

## Startup

The Harmattan installation adds a file to start at boot time into the file ~/.start_conky, and the custom performance monitor can be invoked similarly. The content of the file is as follows:

```
#!/bin/sh
sleep 20
conky -d -c ~/.conkyrc
sleep 5
conky -d -c ~/conky/.conkyrc_XU
exit
```

This definition invokes the weather monitor 20 seconds after boot, and the custom XU-specific conky monitor is invoked 5 seconds after that. Harmattan uses the Internet to fetch weather forecast data, therefore it is necessary to leave enough time for the OS to start up and establish a wireless or wired connection.

## Multiple monitors

Several performance and metric monitors can be added to a single desktop. For instance, we may be interested in both weather and system performance at the same time. One possible future configuration might be to show all of the data in a combined widget. If we decide to use two windows instead, then we can close the performance monitor when it's not needed.

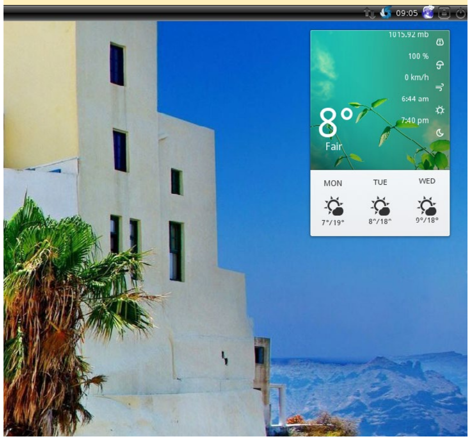
## Technical Notes

With transparent monitor painting, the order of the desktop icons may be hidden if the window settings are incorrect. The hidden icons are visible only when mouse is hovered over them, which can be fixed by tuning the settings and reducing the widget window size.

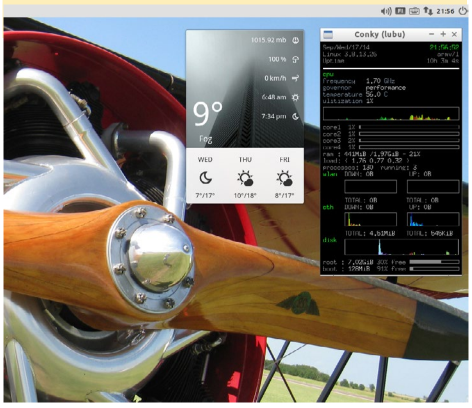
## Sample configuration file

```
 background yes
cpu_avg_samples 2
net_avg_samples 2
out_to_console no
font 7x13
use_xft no
own_window yes
### black own window monitor
own_window_transparent no
own_window_colour black
### transparent background moni-
tor
#own_window_transparent yes
#own_window_type override
#own_window_hints
undecorate,sticky,skip_
taskbar,skip_pager,below
double_buffer yes # double buff-
ering removes flicker
on_bottom yes
update_interval 1
minimum_size 5 5
draw_shades no
draw_outline no
draw_borders no
stippled_borders 0
border_margin 10
border_width 2
default_color white
default_shade_color white
default_outline_color white
alignment bottom_right
gap_x 40 # 20
gap_y 100 # 20
use_spacer yes
no_buffers no
uppercase no
color2 CCCCCC
color8 888888
TEXT
${color8}${time %a %d.%b %y}
$alignr ${color green}${time
%k:%M:%S}
${color8}$sysname $kernel $alignr
$machine
Uptime  $alignr $uptime
${color white}${hr 2}
${color green}cpu
${color slate gray}frequency
${color2}${freq_g } ${color slate
gray}GHz
```

**Harmattan conky in use on the Debian Ezy Wheezy image on ODROID U3**

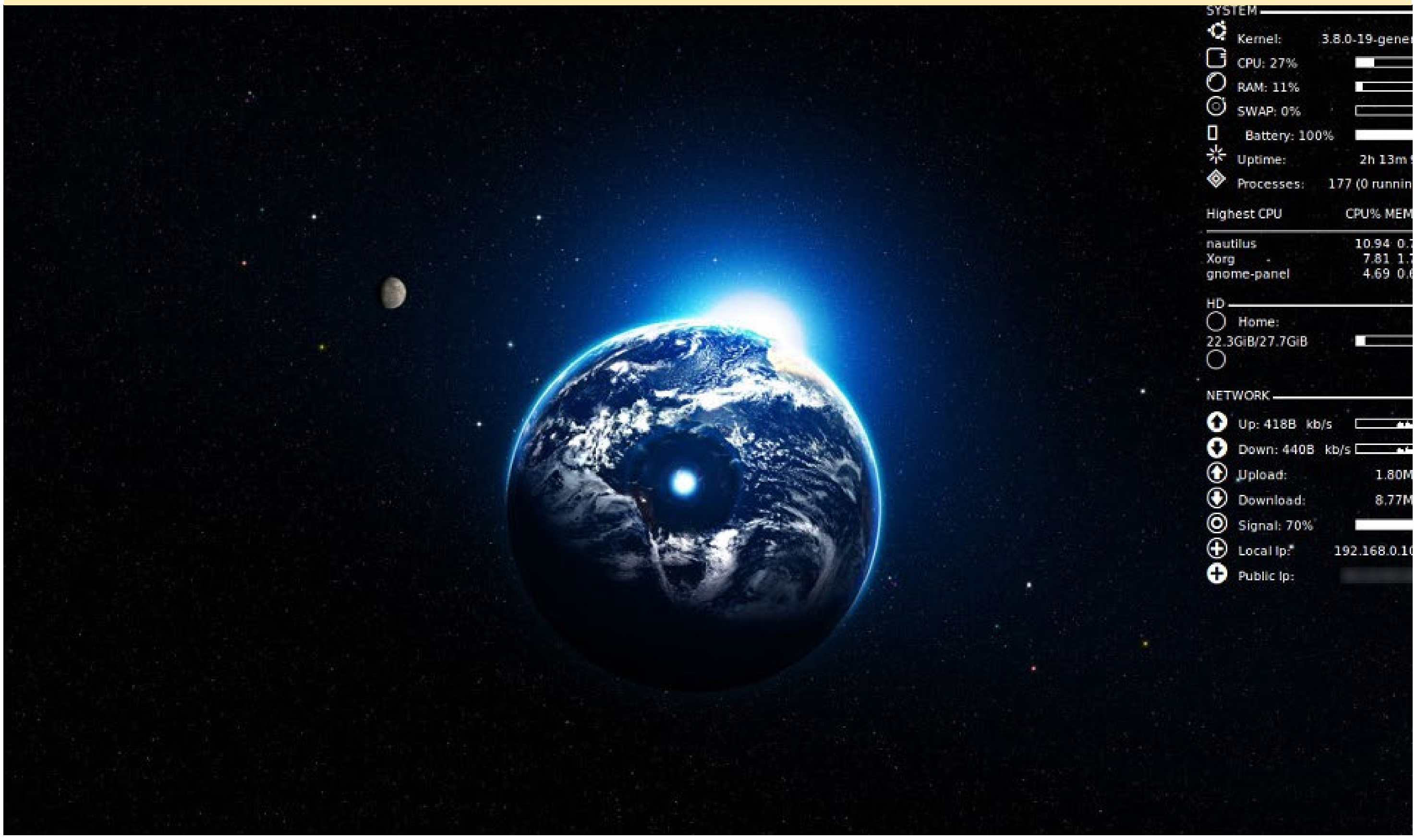


**Performance and weather monitors used together on a Lubuntu image, showing foggy weather and an idle processor**



```
${color slate gray}governor
${color2}${exec cat /sys/devices/
system/cpu/cpu0/cpufreq/scal-
ing_governor}
${color slate gray}clus-
ter     ${color2}${if_match
${exec cat /dev/bL_status | grep
A7 | cut -c18} < 1}big${else}
LITTLE${endif}
${color slate gray}temperature
${color2}${exec echo "scale=1;
$(cat /sys/class/thermal/thermal_
zone0/temp)/1000" | bc}${color
slate gray} C
${color slate gray}ulitization
${color2}${cpu}%
${color8}${cpugraph 25 ff0000
ff00ff}
core1  ${cpu cpu0}% ${cpubar
cpu0}
core2  ${cpu cpu1}% ${cpubar
cpu1}
core3  ${cpu cpu2}% ${cpubar
cpu2}
core4  ${cpu cpu4}% ${cpubar
cpu3}
${color green}wlan${color8}  DOWN
${color2}${downspeed wlan6}
${color8}UP  ${color2}${upspeed
wlan6}
${color8}      ${downspeedgraph
wlan6 25,100 ff0000 0000ff}
  ${color8}${upspeedgraph wlan6
25,100 0000ff ff0000}
${color8}      TO-
TAL  ${color2}${totaldown
wlan6}          ${color8}TOTAL
${color2}${totalup wlan6}
${color green}eth${color8}   DOWN
${color2}${downspeed eth0}
${color8}UP  ${color2}${upspeed
eth0}
${color8}      ${downspeedgraph
eth0 25,100 ff0000 0000ff}
  ${color8}${upspeedgraph eth0
25,100 0000ff ff0000}
${color8}      TO-
TAL  ${color2}${totaldown
eth0}   ${color8}TOTAL
${color2}${totalup eth0}
${color green}disk ${color2}
${diskiograph 30,220 fef7b2
e18522}
${color8}root   ${color2}${fs_
size /} ${color8}${fs_free_perc
/}% free ${fs_bar /}
${color8}boot   ${color2}${fs_
size /media/boot} ${color8}${fs_
free_perc /media/boot}% free
${fs_bar /media/boot}
```

**Harmattan conky in use on the Debian Ezy Wheezy image on an ODROID U3, showing network, system and software statistics in real time**



```
SYSTEM
Kernel:        3.8.0-19-gener
CPU: 27%
RAM: 11%
SWAP: 0%
 Battery: 100%
Uptime:        2h 13m
Processes:     177 (0 runnin

Highest CPU         CPU% MEM
nautilus            10.94 0.
Xorg                 7.81 1.
gnome-panel          4.69 0.

HD
Home:
22.3GiB/27.7GiB

NETWORK
Up: 418B  kb/s
Down: 440B  kb/s
Upload:             1.80M
Download:           8.77M
Signal: 70%
Local Ip:      192.168.0.10
Public Ip:     [redacted]
```

# HARDKERNEL AT ARM TECHCON 2014

## SHOWING OFF THE XU3

by Rob Roy

We had a lot of fun at ARM TechCon 2014! Several members of Hardkernel made the trip from South Korea, and Mauro came all the way from Brazil for a 2-day technology extravaganza in Santa Clara, California. We set up games, demos, and even had a cute Android robot mascot that sang and danced for us. As usual, the Hardkernel booth had lots of visitors who were interested in seeing exactly what the XU3 can do.

Left to right: Bo, Rob Roy, Justin, Mauro, Ryan and Lisa at the Hardkernel ARM TechCon 2014 booth

We featured a sneak peek of the magazine, along with an Angry Birds competition!

Mauro and Suriyan prepared a demo XU3 version of Ubuntu 14.04, including a KVM virtual machine that ran Android inside of Ubuntu while also performing several graphics demos and a hardware-rendered 3D skybox.

Thank you to everyone who stopped by the booth! Make sure to get your tickets early next year to hang out with the Hardkernel team.

# MEET AN ODROIDIAN

## BRUNO DOICHE: ART EDITOR OF ODROID MAGAZINE

***edited by Rob Roy***

*Please tell us a little about yourself.*

As I joked in the "About Me" from Issue 1 of ODROID Magazine, I'm just a regular guy! Like most of our readers, I believe I'm a dude that looks forward to knowing and using computers that are not your regular Wintel machine.

But, I'm your average white collar that works on the IT industry managing Unix and Linux servers, SAN storage, and database management. Before that, I used to work as a photographer after being a photographer's assistant for a couple of years, then before that worked for quite some time in the magazine publishing area doing computer and games magazines. That's how I got involved in the magazine: one day, Rob posted in the forums asking if someone could give him some help, I thought "well, that will be fun to do", and the rest you can read since then around here. When I have time while doing the magazine layout, I like to insert the goofy/slapstick jokes that you guys are probably used to reading in every issue by now!

*How did you get started with computers?*

Well, when I was about 9 years old, I visited a cousin and saw a computer for the first time in my life. It was an MSX hotbit, which is a Brazilian clone of the Japanese Hitbit model, and I remember playing(and beating) the game called Yie Ar Kung Fu. Then, in the next school year I started having computer classes with an Apple II clone called the TK3000, using LOGO and BASIC. And I nagged my parents until they got me a computer for Christmas.

On the advice of a friend, I got myself an MSX. At the time, I didn't understand the concept of different computer platforms, and took me quite a while to learn that my cousin's computer was in fact the same platform that I had. Things were very different back then, you had a friend that had a Commodore Amiga, another that had an IBM PC, and another with an Apple -- it was a ZOO! We tried to run each other's software and we were mostly frustrated until we figured out that stuff.

By 1993 I had my first IBM PC, and stood with it until I got into college, where I went to study Computer Science, and at the lab I worked with Macintoshes. I dropped out of college and went straight to work in design, and ended up graduating as a Photographer and Designer. I was working with publishing mostly, and it was always this organic thing doing art and computer related stuff.

I got my current job and at last graduated in Computer Science and worked mostly with the IT industry. So I got myself an IBM Power4 server, then an IBM Power5, to get my AIX certificate. But they are insultingly power consuming and noisy to have them working for

**From left to right, the 3 first computers that I ever used: MSX Hotbit, TK3000 Apple II clone and MSX Expert**

**Above: This one was an IBM Power4 machine that I once owned, it had about half the processing power that a ODROID XU3 but it weights 35.5 kg (78.0 lb) versus 100g (0.22 lb)!**
**Right: One storage array from the same family of the Power4 with a whopping 540GB capacity, today you can fit 640GB worth of eMMCs in your closed fist**

you at home, and a friend of mine told me about his ODROIDS, so I bought one to use at home to do all the things that I never wanted to leave my desktops/laptops on for all the time. I bought an X2, booted up with one of Rob Roy's Linux distros, and it is still running. If not for the occasional power outage when it rains or things like that, the X2 has been running smoothly since then.

*What types of projects have you done with your ODROIDs?*

I can't live without a home network, so a personal file server is a must. Getting the X2 to transcode video to my playstation 3 was a fun project to do also. Then, setting up Shairport to stream audio from iTunes to a good stereo speaker was another. Lately, I got a new Mac and exported all my mp3s to the X2. Now it also runs as an iTunes server with thousands of songs.

I use the X2 as a torrent machine, and run a program called Sick Rage to get all the TV shows that I want to follow. I also did a silly script with crontab to use the program periscope to fetch all of my downloaded shows' subtitles. There are some virtualisation involving containers and KVM that I still want to have some time to do. Openstack with my U3s is another pet project, and I'm still looking to put all my ODROIDs into a compact computer case powered by a single ATX power supply, but I'm not too engaged with all of that, due to my steady job. Hooking the U3 up to a Motorola Atrix was a fun thing to do as well!

**One of my trusty bikes that I bought used from a guy in Kansas for about $300, then took to Brazil, which was a sweet deal for a pro bike**

*What other hobbies and interests do you have besides computing?*

I love to cycle! I have 3 bicycles, and when I'm able to go biking on a regular basis and listen to some music while doing it I really get into my happy place. I collect vinyl records, both old and new, and although I'm currently very focused on an Atkins diet, I enjoy cooking a lot, so my fiancé was not the most pleased that I stopped cooking risotto for her. I still like to photograph a lot, but just as a hobby. And of course I spend a lot of time looking into design publishing material even today.

*What improvements would you like to see for future ODROIDs?*

Well, a SATA controller and gigabit ethernet are a no-brainer. I know the Hardkernel guys and really, it's not a point of can vs. can't. These guys are doing a great product that really blows out the competition, and there are different ways to look at what to expect into the next generation of ODROIDs:

- **For a file server with high performance, include SATA + 2 gigabit ethernets**
- **As a gaming machine, it needs a better GPU and a VGA/DVI standard port**
- **To support virtualisation, include 2 physical processors, more RAM, and Linux running PAE**

*Which ODROID is your favourite?*

The X2 is my home server, running my SMB file server and the Playstation media server. It handles my torrents and does pretty much anything that I need server-wise. But once I had the U3 hooked on a Motorola Atrix running OpenMSX, that made me think of claiming the U3 to be my favourite but it was just a U3's trick to get attention!

**Yie Ar Kung-Fu and pretty every MSX game runs on one of my U3s, I'll eventually fit a U3 on a hotbit case!**