

ODROID

Años Dos
Num #13
Ene 2015

Magazine

Edición en la Nube



Docker: Desarrollar, Enviar y Ejecutar Aplicaciones en cualquier lugar con Contenedores

Seafile: Software en la Nube Personal Open Source

OwnCloud: Sincroniza y Comparte Archivos Usando tu Servidor Privado ODROID



ODROID-XU3 VS ODROID-U3: COMPARAMOS SU POTENCIAL DE JUEGO

- Programación sin Microsoft: Pon en marcha un Servidor ASP.NET y Mono
- GNU Radio: Sistemas de Radio Real e Investigación de Comunicaciones Inalámbricas

Qué defendemos.

Nos esmeramos en presentar una tecnología punta, futura, joven, técnica y para la sociedad de hoy.

Nuestra filosofía se basa en los desarrolladores. Continuamente nos esforzamos por mantener estrechas relaciones con éstos en todo el mundo.

Por eso, siempre podrás confiar en la calidad y experiencia que representa la marca distintiva de nuestros productos.

Simple, moderno y único.

De modo que tienes a tu alcance lo mejor



HARDKERNEL



Ahora estamos enviando los dispositivos ODROID U3 a los países de la UE! Ven y visita nuestra tienda online!

Dirección: Max-Pollin-Straße 1
85104 Pförring Alemania

Teléfono & Fax

telf : +49 (0) 8403 / 920-920

email : service@pollin.de

Nuestros productos ODROID se pueden encontrar en: <http://bit.ly/1tXPXwe>





Este mes, nuestra atención se centra en demostrar qué los **ODROIDs** pueden ser utilizados como servidores privados en la nube. El almacenamiento de archivos “en la nube” normalmente implica suscribirse a un costoso servicio de terceros, con la posibilidad de que los datos puedan ser hackeados o queden desprotegidos, como hemos visto en noticias recientes con fotos de personajes famosos y correos electrónicos corporativos que se han publicado sin permiso. Con la instalación y configuración de software como Seafile y OwnCloud, puedes mantener tu propio nivel de seguridad usando cualquier dispositivo **ODROID** para alojar archivos en tu casa u oficina. De esta manera, puedes limitar su exposición a una intranet local segura, por lo que la posibilidad de que un intruso desconocido pueda acceder a los ficheros se reduce considerablemente.

Un tema candente en el mundo de la informática en la nube es **Docker**, que permite que las aplicaciones puedan ser desarrolladas sobre la plataforma que elija el desarrollador y luego, ser instalada y ejecutada en casi cualquier arquitectura, incluyendo **ODROIDs**. Como se ha demostrado en la última conferencia **I/O** de Google, **Docker** ofrece un entorno muy estable para distribuir y compartimentar aplicaciones con fines de escalabilidad. Permite que las aplicaciones se puedan montar rápidamente a partir de varios componentes que son completamente portátiles. Fred Meyer, un experto de **Docker** quien recientemente se ha unido al equipo **ODROID Magazine**, presenta la primera parte de una serie interesante con el propósito de conseguir que **Docker** pueda ejecutarse en un dispositivo **ARM**.

Ahora que **ODROID-C1** ha estado disponible durante algún tiempo, el foro **ODROID** ha crecido casi en 8.000 miembros, con muchas sugerencias, consejos y tutoriales publicados diariamente. Tobias ha subido recientemente muchos de sus paquetes de juegos y software útiles al servidor de la comunidad de **Hardkernel**. También ha elaborado una guía para conectarse a su repositorio para poder obtener actualizaciones del kernel mediante un simple comando `apt-get`, que es una manera más apropiada de actualizar software personalizado para **ODROID** de **Debian** y **Ubuntu**.

Como siempre, os traemos un análisis de varios juegos divertidos y de varios programas para **Android**, así como una comparación de la potencia de juego del **ODROID-XU3** frente a la del **ODROID-U3**. Nanik continúa con su excelente serie sobre Desarrollo **Android** con una introducción a la aplicación **Zygote**, y destacamos un tutorial sobre cómo usar tu **ODROID** como una radio digital moderna utilizando el paquete **GNU Radio**.

ODROID Magazine, que se publica mensualmente en <http://magazine.odroid.com/>, es la fuente de todas las cosas **ODROIDianas**. • **Hard Kernel, Ltd.** • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815 • fabricantes de la familia **ODROID** de placas de desarrollo quad-core y la primera arquitectura **ARM** “big.LITTLE” del mundo basada en una única placa. Únete a la comunidad **ODROID** con miembros en más de 135 países en <http://forum.odroid.com/> y explora las nuevas tecnologías que te ofrece **Hardkernel** en <http://www.hardkernel.com/>.



HARDKERNEL

ODROID

Magazine



**Rob Roy,
Editor Jefe**

Soy un programador informático que vive y trabaja en San Francisco, CA, en el diseño y desarrollo de aplicaciones web para clients locales sobre mi cluster de ODROID. Mis principales lenguajes son jQuery, angular JS y HTML5/CSS3. También desarrollo sistemas operativos precompilados, Kernels personalizados y aplicaciones optimizadas para la plataforma ODROID basadas en las versiones oficiales de Hardkernel, por los cuales he ganado varios Premios. Utilizo mi ODROIDs para diversos fines, como centro multimedia, servidor web, desarrollo de aplicaciones, estación de trabajo y como plataforma de juegos. Puedes echar un vistazo a mi colección de 100 GB de software e imágenes ODROID en <http://bit.ly/1fsaXQs>.



**Bob Lechnowsky,
Editor**

Soy el presidente de Respectech, Inc., Consultoría tecnológica en Ukiah, CA, EE.UU. que fundé en 2001. Con mi experiencia en electrónica y programación dirijo a un equipo de expertos, además de desarrollar soluciones personalizadas a empresas, desde pequeños negocios a compañías internacionales. Los ODROIDs son una de las herramientas de las que dispongo para hacer frente a estos proyectos. Mis lenguajes favoritos son Rebol y Red, ambos se ejecutan en los sistemas ARM como el ODROID-U3. En cuanto a aficiones, si necesitas alguna, yo estaría encantado de ofrecerte alguna de la más ya que tengo demasiadas. Eso ayudaría a que tuviese más tiempo para estar con mi maravillosa esposa y mis cuatro hijos estupendos.



**Bruno Doiche,
Editor Artístico Senior**

Consiguió sus habilidades informáticas después de lograr que una fibra óptica volviera a la vida, lograr que su Macintosh volviese de la muerte, lograr que una PS3 volviese de la muerte, lograr que el T400 de su novia volviese de la muerte (una transferencia de datos dd al viejo estilo), y lidiando con las entrañas de su permanente centro de datos de trabajo.



**Nicole Scott,
Editor Artístico**

Soy una experta en Producción Transmedia y Estrategia Digital especializada en la optimización online y estrategias de marketing, administración de medios sociales y producción multimedia impresa, web, vídeo y cine. Gestiono múltiples cuentas con agencias y productores de cine, desde Analytics y Adwords a la edición de vídeo y maquetación DVD. Tengo un ODROID-U3 que utilizo para ejecutar un servidor web sandbox. Vivo en el área de la Bahía de California, y disfruta haciendo senderismo, acampada y tocando música. Visita mi web en <http://www.nicolecscott.com>.



**James LeFevour,
Editor Artístico**

Soy un especialista en medios digitales que disfruta trabajando como freelance en marketing de redes sociales y administración de sitios web. Cuanto más aprendo sobre las posibilidades de ODROID más me ilusiona probar cosas nuevas con él. Me traslade a San Diego desde el Medio Oeste de los EE.UU. Todavía estoy bastante enamorado de muchos aspectos que la mayoría de la gente de la Costa Oeste ya dan por sentado. Vivo con mi encantadora esposa y nuestro adorable conejo mascota; el cual mantiene mis libros y material informático en constante peligro.



**Manuel Adamuz,
Editor Español**

Tengo 31 años y vivo en Sevilla, España, y nací en Granada. Estoy casado con una mujer maravillosa y tengo un hijo. Hace unos años trabajé como técnico informático y programador, pero mi trabajo actual está relacionado con la gestión de calidad y las tecnologías de la información: ISO 9001, ISO 27001, ISO 20000 Soy un apasionado de la informática, especialmente de los microordenadores como el ODROID, Raspberry Pi, etc. Me encanta experimentar con estos equipos y traducir ODROID Magazine. Mi esposa dice que estoy loco porque sólo pienso en ODROID. Mi otra afición es la bicicleta de montaña, a veces participo en competiciones semiprofesionales.



GNU RADIO - 6



JUEGOS ANDROID: METAL SLUG DEFENSE - 7



INFORMATICA EN LA NUBE - 8



JUEGOS ANDROID: HEAVENSTRIKE RIVALS - 16



REPOSITORIO DE KERNEL - 17



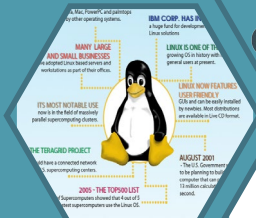
JUEGOS LINUX - 20



IMAGENES DE LA COMUNIDAD - 25



DOCKER - 26



HISTORIA DE LINUX - 30



MONO - 31



DESARROLLO ANDROID - 32



GRAFICOS JAVA - 34



SEAFIL - 38



HISTORIA DE HARDKERNEL - 41



CONOCIENDO A UN ODROIDIAN - 42

GNU RADIO

COMO LLEVAR TU PROGRAMAS DE RADIO PERSONALES EN EL SIGLO XXI

por @denash

GNU Radio es un conjunto de herramientas de desarrollo de software libre y de código abierto que proporciona bloques de procesamiento de señales para implementar software radio. Se puede utilizar con hardware RF comercial de bajo coste para crear radios definidas por software, o sin hardware en entornos de simulación. Es ampliamente utilizado por aficionados, en entornos académicos y comerciales para apoyar la investigación de comunicaciones inalámbricas y sistemas de radio en el mundo real.

Para utilizar GNU Radio, arranca una distribución Debian o Ubuntu y elige uno de los siguientes métodos de instalación:

1. Escriba lo siguiente en una ventana de terminal para instalar GNU Radio desde el repositorio de Debian

```
$ wget -c \
http://ftp.debian.org/debian/
pool/main/q/qwtplot3d/libqwt-
plot3d-qt4-0_0.2.7+svn191-7_arm-
hf.deb
$ wget -c \
http://ftp.debian.org/debian/
pool/main/q/qwtplot3d/libqwt-
plot3d-qt4-dev_0.2.7+svn191-7_
armhf.deb
$ sudo dpkg -i libqwtplot3d-
qt4-*.deb
$ gnuradio
```

2. Utiliza este script precompilado para desarrollar GNU Radio desde la fuente: <http://bit.ly/IAWW3vr>. Guárdalo en una carpeta temporal con el nombre de "build-gnuradio". Luego, escribe lo siguiente tras navegar hasta la carpeta temporal:

```
$ sudo chmod +x ./build-gnuradio
&& ./build-gnuradio && gnuradio
```

3. Sigue estas instrucciones para crear tu propio script build-gnuradio:

Descarga el script de desarrollo estándar de GNU Radio:

```
$ wget http://www.sbrac.org/files/
build-gnuradio
```

Crea dos variables locales:

```
$ TEST=-DCMAKE_CXX_
FLAGS:STRING="-march=armv7-a
-mcpu=cortex-a9 -mfpu=neon \
-mfloat-abi=hard"

$ TEST2=-DCMAKE_C_FLAGS:STRING="-
march=armv7-a -mcpu=cortex-a9
-mfpu=neon \
-mfloat-abi=hard"
```



Podríamos modificar esta clásica radio, aunque no creo que lo hiciéramos si tuviésemos una en nuestras manos

Busca todas las ocurrencias de cmake en el script build-gnuradio y añade "\$TEST" "\$TEST2" a los argumentos. En torno a la línea 348, encontrarás:

```
for dir in /lib /usr/lib /usr/
lib64 /lib64 /usr/lib/x86_64-
linux-gnu /usr/lib/i386-linux-gnu
```

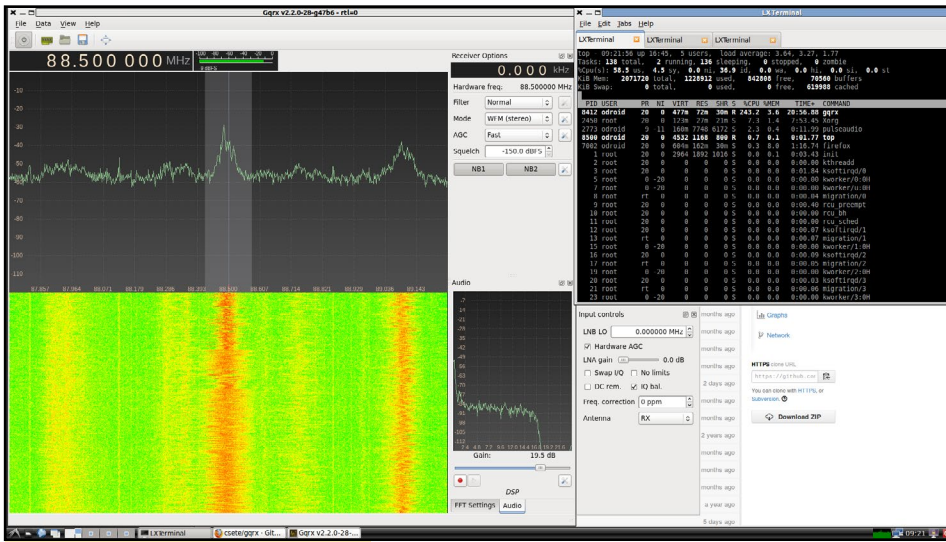
y añade el siguiente fragmento:

```
/usr/lib/arm-linux-gnueabi /
usr/lib/arm-linux-gnueabi
```

Por último, ejecuta el script:

```
$ ./build-gnuradio
```

Ten en cuenta que gnuradio no se debe compilar usando la marca -j4 ya que GNU Radio parece dividirse cuando se compila en paralelo.



Ejecutando GQRX

Añadir una entrada de micrófono

I. Ejecuta la aplicación “PulseAudio VolumeControl”, que se encuentra en el menú de Aplicaciones.

2. En la pestaña Configuración, selecciona “Analog Stereo + Analog Mono Input”

3. Instala “GNOME ALSA Mixer” vía Centro de Software de Ubuntu y ejecútalo.

4. Ignorar la ventana de error. Parece que se trata de un problema de permisos al guardar el archivo de configuración.

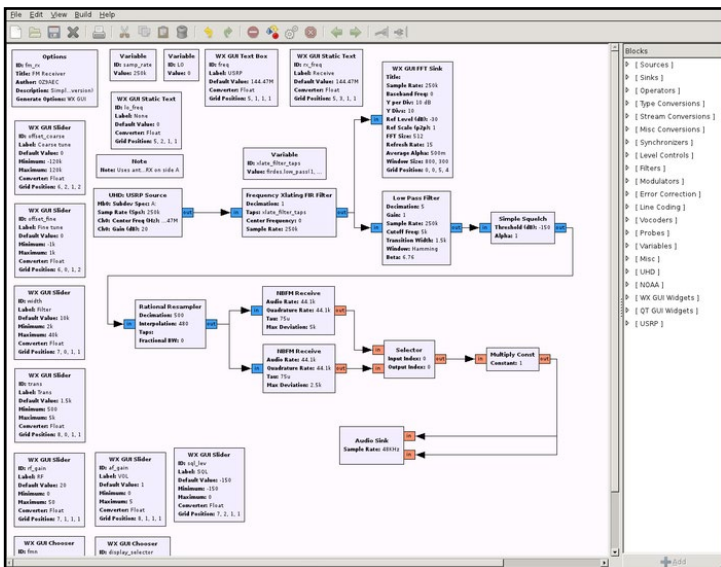
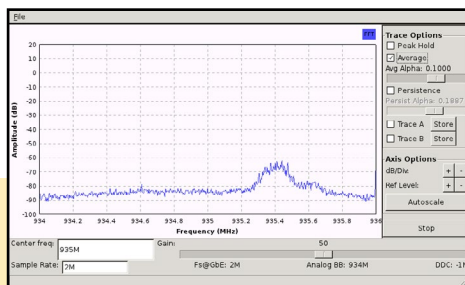
5. Comprueba los siguientes 3 elementos del Mixer GUI: “MIC Bias VCM Bandgap”, “MIC1 Mix” y “Left ADC Mixer MIC1”

Usando la herramienta uhd_fft.py para analizar un canal de transmisión GSM

6. Puedes ajustar el nivel de entrada o la opción de mejora con la barra de control “ADCL”, “ADCL Boo”, “MICI”, y “MICI Boo”

7. Puedes ver el nivel de entrada del micrófono en tiempo real en la pestaña “PulseAudio VolumeControl” del Dispositivo de entrada.

Puedes encontrar más información
sobre GNU Radio en la web principal
<http://gnuradio.org>.



Captura de pantalla del GRC con un receptor FM de banda estrecha

METAL SLUG
DEFENSE
UN NUEVO ENFOQUE DE
UNA SERIE MUY QUERIDA

por Bruno Doiche

Me encanta el juego Metal Slug, que he jugado en una maquina arcade, en mi NEO GEO CD, en un teléfono celular y emulandolo en todos los ordenadores que he tenido - incluso en los 4 o 5 potentes servidores que he tenido.

Sin embargo, tras innumerables juegos, pensé que ya lo había visto todo sobre esta serie y de repente... ¡BAM! SNK lanza un juego de defensa basado en Metal Slug.

Echa un vistazo por ti mismo - ¡Es un juego muy divertido!

<http://bit.ly/1iH5Z2k>



ODROID MAGAZINE 7

MI PROPIO OWN CLOUD

GUARDA TUS ARCHIVOS DE FORMA SEGURA CON UN SERVIDOR PERSONAL EN LA NUBE

por Venkat Bommakanti

OwnCloud es una aplicación para compartir y sincronizar archivos a nivel de empresa que puedes alojar en tu propio centro de datos, en tus servidores, usando tu propio almacenamiento. OwnCloud proporciona un acceso universal a archivos a través de una única interfaz de usuario para todos tus sistemas, independientemente de la arquitectura. Los usuarios pueden acceder a los archivos de la empresa desde cualquier dispositivo, en cualquier momento y desde cualquier lugar, mientras que la TI permite gestionar, controlar y auditar la actividad y el intercambio de archivos para garantizar que se cumplen las medidas de seguridad y conformidad.

En este artículo, muestro los detalles de instalación, configuración y uso de las versiones más recientes del mejor software de su clase para implementar una solución OwnCloud segura y robusta, que consta de los siguientes componentes:

Odroid XU3 o XU3 Lite con 16TB de almacenamiento por USB3 (más 16TB para backup) y Ethernet Gigabit - Software LEMP (Linux 3.10.60, Enginx 1.4.6, MySQL 5.5.40, Php 5.5.9) - phpMyAdmin 4.0.10 - OwnCloud 7.0.4

La documentación online para ownCloud es escasa, obsoleta y confusa en muchos de los casos, además de errónea, inexacta y sin revisar. Se han analizado diversos recursos de información, y se ha montado cuidadosamente una configuración de trabajo para que tu viaje a través del proceso de configuración sea lo más sencillo posible.

Requisitos

1. Una placa **ODROID XU3-Lite o XU3**, con un adaptador de corriente adecuado. Aunque este artículo se centra en un **XU3-Lite**, también se puede aplicar a un **U3** o a un **C1**.
2. Un módulo **eMMC 5.0** de **16GB** o un tarjeta **MicroSD Clase 10** con la última imagen de escritorio **Lubuntu** específica para **XU3-lite**
3. Un disco duro externo **USB3** de **16TB**, como un **WD Ultra** o **Toshiba Canvio** que se usará para el almacenamiento de datos **ownCloud** primario. Se puede añadir un segundo disco duro externo **USB3** de **16TB** para realizar copias de seguridad de los datos primarios.

Instalar Lubuntu

Instala la última imagen para **XU3** en el módulo **eMMC** y arrancar el sistema con la pantalla **HDMI** conectada. Ejecuta “**ODROID Utility**” para ampliar la partición del sistema operativo. Reinicia y ejecuta la utilidad de nuevo para actualizar el kernel, los controladores de vídeo y el resto de cuestiones relacionadas con el sistema. Reinicie una vez más antes de continuar con el siguiente paso.

Preparar el sistema

Si lo deseas puedes hacer una **Backup** de los archivos del sistema operativo y software en el **HD** externo con la utilidad **dd**. Con el **XU3** apagado, haz lo siguiente:

- **Conecta el HD externo principal, que se utiliza para almacenar los datos owncloud principales, al Puerto USB3 Host Tipo A,**
- **Conecta el HD externo secundario, que será utilizado como backup de los datos ownCloud, a uno de los muchos puertos USB2 Tipo A. Puesto que las copias de seguridad se pueden programar durante las horas de poca actividad, una conexión USB2 será suficiente para esta unidad.**
- **Conecta el cable USB3 OTG al puerto USB3 y fija el otro extremo del cable al dongle Ethernet Gigabit. Conecta el dongle a tu router usando un cable normal Cat5E o Cat6.**

Puesto que normalmente los dos discos duros vienen formateados en NTFS de fábrica, deberían ser detectados y montados automáticamente.

Instalar MySQL

En lugar de utilizar livianas opciones de gestión de datos mediante el sistema SQLite por defecto, he elegido el popular y ampliable MySQL RDBMS para gestionar los metadatos administrativos del ownCloud.

Instala el software MySQL con el siguiente comando:

```
$ sudo apt-get install mysql-server mysql-client
```

Reinicia el sistema y comprueba la instalación:

```
$ mysql -V
mysql Ver 14.14 Distrib 5.5.40, for debian-linux-gnu (armv7l) using readline 6.3
```

Configura la contraseña de root en el primer inicio:

```
$ mysql -u root -p
```

Introduce una contraseña y anótala en algún lugar por seguridad. Para este ejemplo, he utilizado “odroid” como contraseña para el usuario root. También se puede comprobar la instalación utilizando los siguientes comandos SQL desde la interfaz de línea de comandos de MySQL:

```
mysql> SHOW VARIABLES LIKE "%version%";
mysql> STATUS;
mysql> show databases;
mysql> select user,host from mysql.user;
mysql> exit
```

De forma alternativa, puedes comprobar la instalación mediante la aplicación de administrador de MySQL de este modo:

```
$ mysqladmin -u root -p version
```

Instala la base de datos del sistema y asegura la instalación usando los siguientes comandos, Una vez instalado, MySQL estará listo para ser usado por ownCloud:

```
$ sudo mysql_install_db
```

```
$ sudo mysql_secure_installation
```

Instalar nginx

Con el fin de crear una instalación eficiente y sólida, hemos optado por el ágil servidor web nginx frente al servidor apache por defecto. Puedes consultar el número de Agosto del 2014 de ODDROID Magazine para obtener instrucciones específicas de cómo instalar y configurar nginx.

Los pasos en general son los siguientes:

Instala nginx utilizando el comando:

```
$ sudo apt-get install nginx-full
```

Verifica la instalación de nginx

Comprueba el nombre de usuario que tiene la instalación nginx, se utilizará más adelante:

```
$ sudo grep user /etc/nginx/nginx.conf user www-data;
```

Instala las credenciales SSL utilizando los comandos (cada comando en una única línea):

```
$ sudo cd /etc/nginx/ && sudo mkdir ssl
$ sudo openssl req -x509 -nodes -days 365 \
-newkey rsa:2048 -keyout \
/etc/nginx/ssl/nginx.key -out \
/etc/nginx/ssl/nginx.crt
```

Actualiza la configuración de nginx para atender las necesidades de SSL, PHP5 y owncloud:

```
$ sudo cd /etc/nginx/sites-available
$ sudo cp default default-orig
$ sudo medit default
```

Reemplaza el bloque {...} del servidor existente con la siguiente configuración. Cada fragmento de configuración debería estar en su propia línea:

```
...
# our php-handler - add this
upstream php-handler {
    server unix:/var/run/php5-fpm.sock;
}

# update section like so:
server {
    listen 80 default_server;
    listen [::]:80 default_server ipv6only=on;

    # ssl support
    listen 443 ssl;

    root /usr/share/nginx/html;

    # try php file execution first
    index index.php index.html;

    # Make site accessible from http://your-XU3-
    host-ip-addr/
    server_name <your-XU3-host-ip-addr>;
```

```
        # ssl credentials
        ssl_certificate      /etc/nginx/ssl/nginx.crt;
        ssl_certificate_key  /etc/nginx/ssl/nginx.key;

        # set max upload size
        client_max_body_size      10G;
        fastcgi_buffers            64 4K;
        client_body_buffer_size   2M;

        # setup calendar, contact, webdav options
        rewrite ^/caldav(.*)$ /remote.php/caldav$1 re-
        direct;
        rewrite ^/carddav(.*)$ /remote.php/carddav$1
        redirect;
        rewrite ^/webdav(.*)$ /remote.php/webdav$1 re-
        direct;

        location = /robots.txt {
            allow all;
            log_not_found off;
            access_log off;
        }

        # disabling of .ht* checks doesn't work (from
        here) for nginx.
        # so using /oc-data as the ownCloud data direc-
        tory, instead of
        # the typical data directory: /usr/share/nginx/html/
        ownCloud/data.
        # retained for future support when issue gets fixed in
        ownCloud
        location ~ ^/(?\.ht|oc-data|config|db_struc-
        ture\.xml|README){
            deny all;
        }

        location / {
            # First attempt to serve request as
            file, then
            # as directory, then fall back to dis-
            playing a 404.
            try_files $uri $uri/ index.php;

            # The following 2 rules are only needed
            with webfinger
            rewrite ^/.well-known/host-meta /pub-
            lic.php?service=host-meta last;
            rewrite ^/.well-known/host-meta.json /
            public.php?service=host-meta-json last;
            rewrite ^/.well-known/carddav /remote.
            php/carddav/ redirect;
            rewrite ^/.well-known/caldav /remote.
            php/caldav/ redirect;
            rewrite ^(/core/doc/[^\/]++)$ $1/index.
            html;
        }

        # redirect server error pages to the static
        pages
        error_page 404 /404.html;
        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root /usr/share/nginx/html;
        }

        # pass the PHP scripts to FastCGI server lis-
        tening on fpm-socket
        location ~ \.php(?:$|/) {
            fastcgi_split_path_info ^(.+\.(php)
            (/.+)$;

            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $docu-
            ment_root$fastcgi_script_name;
            # $fastcgi_path_info parse fails
            in latest php5-fpm.  disable it.
            # fastcgi_param PATH_INFO $fastcgi_
            path_info;
```

```

        fastcgi_pass php-handler;
        fastcgi_read_timeout 600;
    }

    # since "phpmyadmin" is a db-admin app, obfus-
    cate it by using a
    # random (or is it not) appname "OYA16z2-xFg" ;!).
    make it use
        # a BASIC authentication dialog prior to
    displaying its own login page.
    # the BASIC authentication credentials
    are placed in the phpmyadmin_pass
    # file
    location /OYA16z2-xFg {
        auth_basic "PHPMyAdmin Login";
        auth_basic_user_file /etc/nginx/phpmyad-
    min_pass;
    }
    ...

```

Guarda el archivo de configuración de nginx tras completar el paso anterior.

Crea páginas de error HTML con marcadores de posición si no existen:

```

/usr/share/nginx/html/404.html
/usr/share/nginx/html/50x.html

```

Instalar php5

Escribe los siguientes comandos para instalar los requisitos previos para PHP5:

```

$ sudo apt-get install autoconf automake autotools-
dev libtool curl
$ sudo apt-get install libcurl4-openssl-dev lbzip2
$ sudo apt-get install php5 php5-dev php5-cgi php5-
fpm php5-curl php5-gd
$ sudo apt-get install php5-mysql php5-gmp php5-
imagick php5-ldap php5-intl
$ sudo apt-get install php5-ldap php5-mcrypt libm-
crypt-dev php-xml-parser
$ sudo apt-get install php5-xsl php-apc phpmyadmin

```

phpMyAdmin, una herramienta basada en PHP, es una útil aplicación para la gestión de bases de datos MySQL. Durante su instalación, ignora la configuración del servidor web ya que nginx no aparece como una opción.

En el paso donde se configura la base de datos MySQL, selecciona "YES" y usa tu contraseña preferida, que por simplicidad en este ejemplo es "odroid". Asegúrate de usar una contraseña más segura en tu instalación.

Ten en cuenta que en el anterior apartado, ya habíamos incluido la configuración php5-fpm necesaria. Sin embargo, necesitamos hacer un pequeño cambio en la configuración:

```

$ sudo cd /etc/php5/fpm/pool.d/
$ sudo cp www.conf www.conf-orig
$ sudo medit www.conf

```

Agrega la siguiente configuración al fichero existente, para que coincida con la configuración de nginx ya establecida en el paso anterior:

```
listen = /var/run/php5-fpm.sock
```

La carpeta raíz de nginx es /usr/share/nginx/html, donde debes crear un archivo de prueba php, que usarás más tarde para probar la instalación de PHP5:

```

$ sudo cd /usr/share/nginx/html
$ su
# echo '<?php echo exec(`whoami`); ?>' > info.php
# echo '<?php phpinfo(); ?>' >> info.php

```

Mejora la seguridad de ejecución de los archivos fijando los siguientes parámetros en el archivo de configuración de PHP5:

```
$ sudo medit /etc/php5/fpm/php.ini
```

Aplica estas opciones:

```

cgi.fix_pathinfo=0
display_errors = On
display_startup_errors = On
output_buffering = 0

```

Cambia las siguientes opciones en el mismo archivo según tus necesidades:

```

upload_max_filesize = 50M
max_file_uploads = 5
post_max_size = 60M
default_socket_timeout = 600

```

Guarda los cambios. Ten en cuenta que algunos de los cambios de configuración anteriores están relacionados con la instalación de ownCloud. Después, aplica los cambios de configuración de los componentes instalados:

```

$ sudo service php5-fpm stop && sudo /etc/init.d/
mysql stop && sudo service nginx stop
$ sudo service nginx start && sudo /etc/init.d/mysql
start && sudo service php5-fpm start

```

Para verificar la instalación inicia el navegador web e introduce `http://<dirección IP XU3>/info.php`. Puesto que previamente creamos un simple script PHP, el resultado debe coincidir con la captura de pantalla que se muestra en la Figura 1.

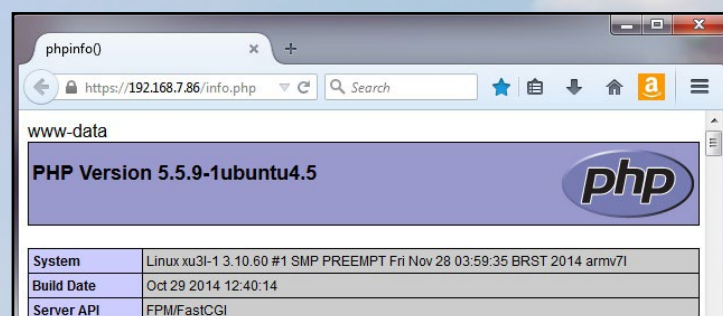


Figura 1 - página de prueba PHP que muestra el resultado de info.php

Si revisar el resultado, observarás que un módulo PHP5 llamado mcrypt está desactivado, lo puedes activar con el siguiente comando:

```
$ sudo php5enmod mcrypt
```

Reinicia php5-fpm, mysql y nginx con los comandos anteriores y actualiza la página del navegador para revisar el script de información PHP. Ahora debería aparecer el módulo mcrypt activado.

Instalar ownCloud

Crea un directorio en tu directorio home y navega hasta él:

```
$ mkdir ownCloud && cd ownCloud
```

Descarga el último tarball de ownCloud para Linux y su archivo hash MD5 correspondiente:

```
$ wget https://download.ownCloud.org/community/own-Cloud-7.0.4.tar.bz2
$ wget http://bit.ly/1GIchxr
```

Verifica la integridad del archivo ownCloud:

```
$ cat ownCloud-7.0.4.tar.bz2.md5
6d4a3f9275d1f2b2607e7e6484051d4c -
$ md5sum ownCloud-7.0.4.tar.bz2
6d4a3f9275d1f2b2607e7e6484051d4c ownCloud-7.0.4.tar.bz2
```

Si los números md5sum coinciden, podemos instalarlo. La autenticidad del archivo también se puede comprobar usando las instrucciones de <http://bit.ly/13Nlfeu>.

Expande el tarball a un nuevo subdirectorio:

```
$ mkdir ~/zBU/ownCloud/oc
$ cd ~/zBU/ownCloud/oc && cp ../ownCloud-7.0.4.tar.bz2 .
$ tar -xjf ownCloud-7.0.4.tar.bz2
```

Después, mueve el contenido del tarball al directorio raíz de nginx en /usr/share/nginx/html:

```
$ sudo mv ownCloud /usr/share/nginx/html/
```

La instalación ownCloud viene con un archivo de configuración php de ejemplo llamado config.sample.php. Haz una copia del mismo y edita la copia con la configuración adecuada:

```
$ cd /usr/share/nginx/html/ownCloud/config
$ sudo cp config.sample.php config.php
$ sudo medit ./config.php
```

Actualiza la siguiente sección, colocando la dirección IP de tu XU3 en <XU3-host-ip-address> sin utilizar los símbolos “<” y “>”:

```
'apps_paths' => array(
    array(
        'path'=> '/usr/share/nginx/html/own-Cloud/apps',
        'url' => '/apps',
        'writable' => true,
    ),
),

'trusted_domains' =>
array (
    'localhost',
    '<XU3-host-ip-address>',
),
```

Crea el directorio de datos ownCloud junto con un directorio para las copias de seguridad. Luego, cambia las propiedades y otros atributos de los diferentes directorios de owncloud:

```
$ cd /usr/share/nginx/html/ownCloud
$ sudo mkdir oc-data && sudo mkdir oc-data-bu
$ sudo chown -R root:root /usr/share/nginx/html/ownCloud/
$ sudo chown -R www-data:www-data /usr/share/nginx/html/ownCloud/config/
$ sudo chown -R www-data:www-data /usr/share/nginx/html/ownCloud/oc-data/
$ sudo chmod 0775 /usr/share/nginx/html/ownCloud/oc-data/
$ sudo chown -R odroid:odroid /usr/share/nginx/html/ownCloud/oc-data-bu/
$ sudo chmod 0775 /usr/share/nginx/html/ownCloud/oc-data-bu/
$ sudo chown root:root /usr/share/nginx/html/ownCloud/.htaccess
$ sudo chown -R www-data:www-data /usr/share/nginx/html/ownCloud/apps/
```

Los subdirectorios config/, oc-data/ y apps/ deben pertenecer al directorio www-data, creado anteriormente. Además, ten en cuenta que el directorio de copias de seguridad oc-data-bu puede ser utilizado por tareas de cron-job (bajo el usuario odroid) para hacer backup del contenido del directorio de datos de owncloud. Esto permite al usuario odroid restaurar los datos en caso de ser necesario.

Por defecto, la instalación de ownCloud supone el uso de Apache, que depende del archivo .htaccess para garantizar las restricciones de acceso correctas, lo cual dificulta el funcionamiento de nginx. Para hacer frente a esto, tenemos que mover los directorios oc-data-bu y oc-data desde la estructura de directorios de nginx al directorio raíz del sistema “/”:

```
$ sudo mv /usr/share/nginx/html/ownCloud/oc-data /
$ sudo mv /usr/share/nginx/html/ownCloud/oc-data-bu /
```

Las propiedades y los permisos se mantendrán sin cambios, permitiendo a ownCloud acceder correctamente a los directorios. Ahora podemos usar estos directorios para crear puntos

de montaje para los dos discos duros externos USB3. Actualiza el archivo fstab para que los puntos de montaje se mantengan aunque se reinicie el sistema.

```
$ cd /etc
$ sudo medit ./fstab
```

Agrega las siguientes entradas, cada una en una sola línea:

```
# WD My Passport Ultra 1TB - external HD #1
/dev/sda1 /oc-data ext4 defaults,errors=remount-
ro,noatime,nodiratime 0 2

# Toshiba Canvio 1TB - external HD #2
/dev/sdb1 /oc-data-bu ext4 defaults,errors=remount-
ro,noatime,nodiratime 0 2
```

Tras realizar una copia de todos los datos existentes en las unidades de disco duro, puede utilizar la utilidad GParted para reformatearlas en ext4, luego reinicia el sistema. Las entradas del sistema de archivos deben ser verificadas:

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
...
/dev/sda1        917G   72M   871G   1% /oc-data
/dev/sdb1        917G   72M   871G   1% /oc-data-
bu
```

Esto garantiza que ownCloud usará las grandes unidades de disco duro de 1 TB de una forma razonablemente rápida. Aunque 1TB puede no ser suficiente en algunos casos, ¡Definitivamente es mejor que unos pocos e insignificantes gigabytes!

Configurar phpmyadmin

Aunque la instalación de phpMyAdmin ya ha sido tratada, su configuración está incompleta. Si la instalación de MySQL muestra que funciona con phpMyAdmin, podemos suponer que una parte importante de la instalación ya se ha completado correctamente. Tenemos que preparar phpMyAdmin para que pueda ser usado bajo nginx, de modo que comprueba esto:

```
$ sudo ln -s /usr/share/phpmyadmin /usr/share/nginx/html
$ ls -ltr /usr/share/nginx/html
...
lrwxrwxrwx 1 root root 21 Dec 12 13:46 phpmyadmin
-> /usr/share/phpmyadmin
```

Luego, crea las credenciales de autenticación básicas:

```
$ openssl passwd
Password: birdsong
Verifying - Password:
Warning: truncating password to 8 characters
PUzMLT4M8HMDY
```

Y después, crea un archivo de contraseñas:

```
$ cd /etc/nginx
```

```
$ sudo touch phpmyadmin_pass
$ sudo medit ./phpmyadmin_pass
```

Agrega lo siguiente en la primera línea y guarda el archivo:

```
zWarlock:PUzMLT4M8HMDY
```

Aunque utilizamos una autenticación básica como ejemplo, es posible que desees un método más sólido en tu instalación.

Normalmente, phpMyAdmin es accesible a través de la url: `http://<XU3-Lite-ip-address>/phpmyadmin`. Sin embargo y puesto que usamos una ofuscación que hemos configurado con anterioridad, tendremos que utilizar la dirección URL `http://<XU3-Lite-ip-address>/OYA16z2xFg`, debería aparecer una pantalla de pre-acceso como la siguiente:

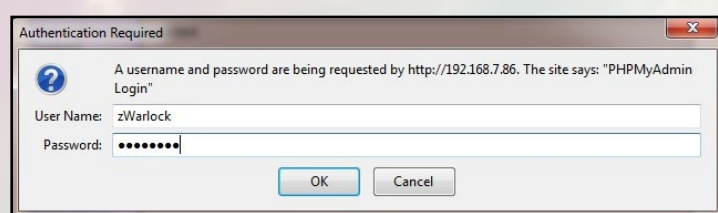


Figura 2: pantalla pre-acceso de phpmyadmin

Usa las mismas credenciales especificadas en la configuración de nginx (username: zWarlock, password: birdsong). Haz clic en Aceptar, debería llevarte a la página de inicio de sesión de phpMyAdmin, similar a la mostrada en la Figura 3.

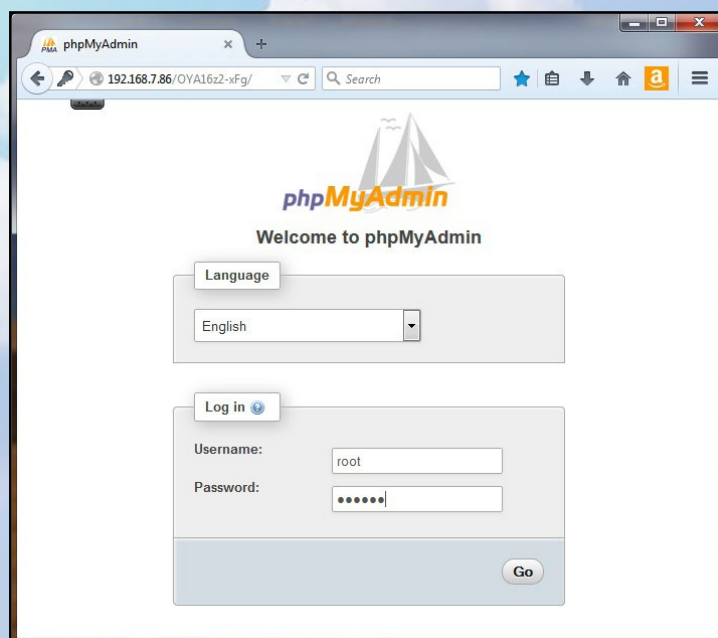


Figura 3: Página de inicio de sesión de phpmyadmin

Usa las siguientes credenciales de acceso especificadas anteriormente (username: root, password: odroid). Tras iniciar sesión correctamente, aparecerá una página de inicio como la que se muestra en la Figura 4.

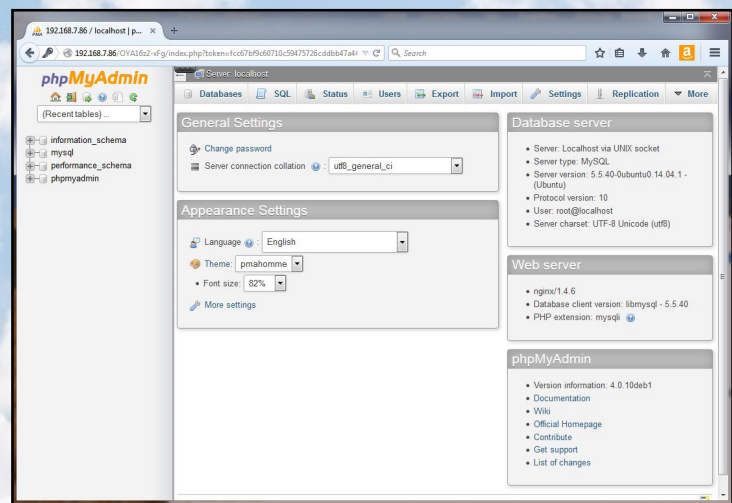


Figura 4: Página de Inicio de phpMyAdmin

Puedes ver las bases de datos MySQL a la izquierda de la pantalla. La Figura 5 muestra los usuarios preliminares en la base de datos. Reinicia el sistema antes de pasar al paso final.

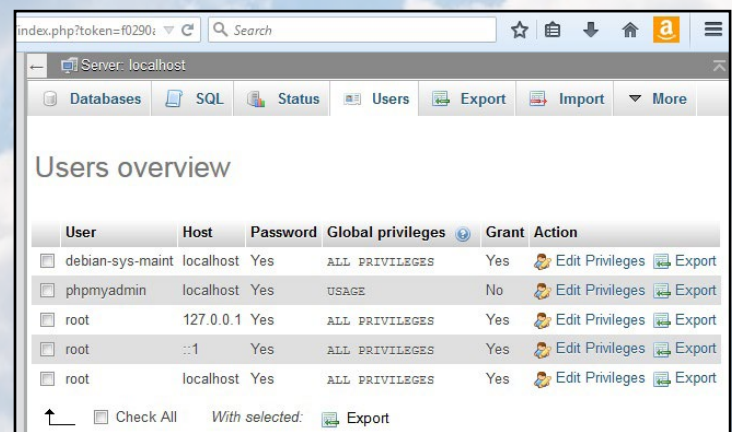
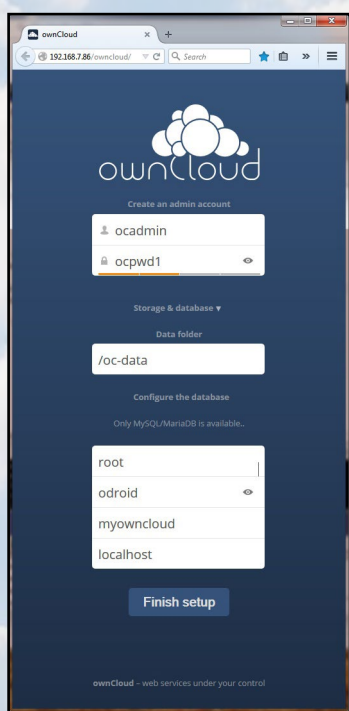


Figura 5: phpmyadmin mostrando los usuarios preliminares



Completar la instalación de ownCloud

Acceda al sitio de ownCloud con la URL `http://<XU3-Lite-ip-address>/ownCloud`. Debería ser direccionado a la página de configuración de ownCloud. Rellene el formulario con la información que se muestra en la Figura 6. Anota las credenciales (nombre de usuario: ocadmin, contraseña: ocpwd1).

Figura 6: Penúltimo paso del ownCloud con los datos cumplimentados

Haga clic en “Finish Setup”, regresará a una página con la lista de clientes de escritorio disponibles para ser instalados. Podemos comprobar si la base de datos ownCloud y los usuarios se han configurado correctamente con phpMyAdmin, como se ve en la figura 7.

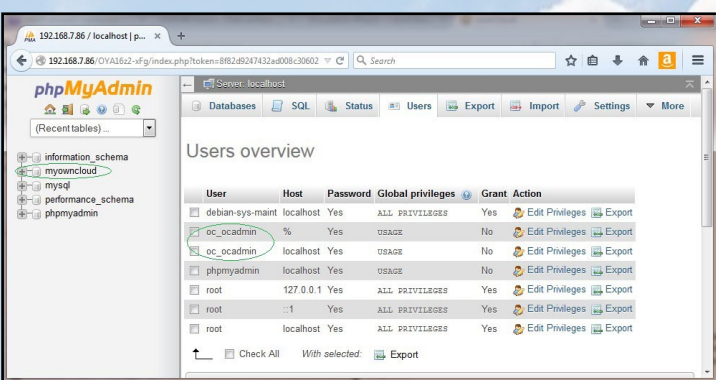


Figura 7: base de datos de ownCloud y usuarios en phpMyAdmin

OwnCloud muestra la página que aparece en la Figura 8 indicando los clientes disponibles. Haz clic en la opción

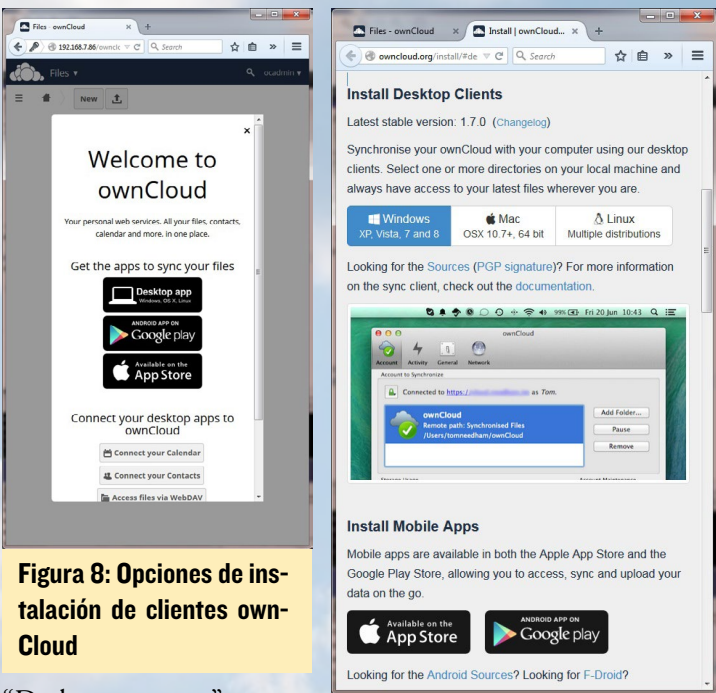


Figura 8: Opciones de instalación de clientes ownCloud

“Desktop app”, se mostrará una página similar a la Figura 9

He seleccionado la opción de Windows para usarlo en mi sistema Win7. Puedes seleccionar la opción más adecuada para tu caso. Aparecerá un mensaje para descargar la versión 1.7.0 de cliente de escritorio. Selecciona “Save File”, y tras completarse la descarga ejecutarlo para instalar el cliente. Cuando aparezca la ventana de acceso del cliente, utiliza las credenciales de acceso de la figura 6 (username: ocadmin, password: ocpwd1). Se mostrará una ventana para configurar la sincronización de carpetas como muestra la Figura 10.

Figura 9: Instalación del cliente de escritorio de ownCloud

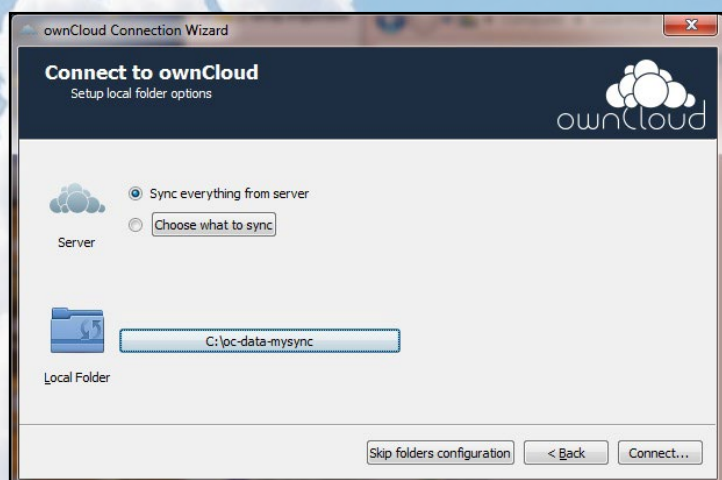


Figura 10: Sincronización de carpetas del cliente ownCloud

Crea un directorio local en c:\ oc-datos-mySync que es utilizado para la sincronización local. Haz clic en Connect y espere a que finalice, se mostrará la pantalla final del cliente. Rellenado con la información adecuada, debe verse como la Figura 11.

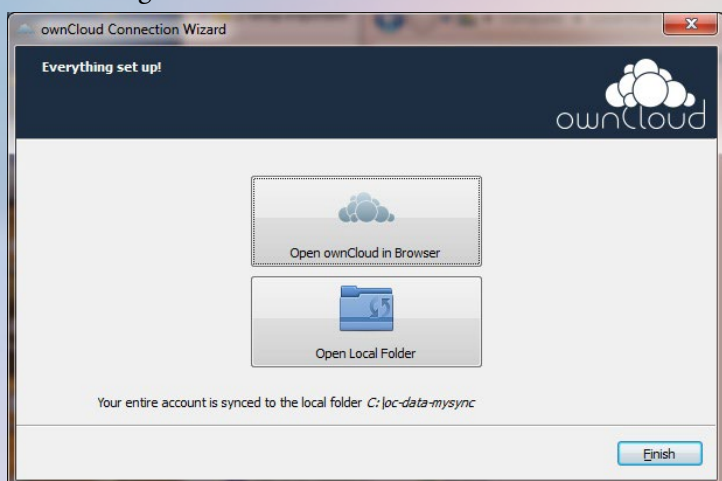


Figura 11: Configuración final del cliente ownCloud

Haz clic en la opción “ Open ownCloud in Browser “, se mostrará una página de acceso a ownCloud. Introduzca las credenciales, debería ver una página como la Figura 12. Por último, cierra la ventana de bienvenida.

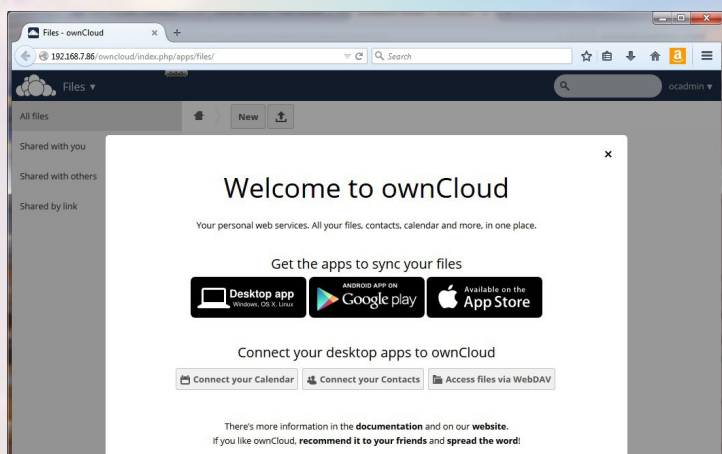


Figura 12: Primer acceso al cliente ownCloud

El proceso de instalación modifica la configuración PHP de ownCloud hasta el punto que ownCloud no permite cargar cualquier archivo, en lugar de tener que añadir una configuración especial para usar una aplicación de almacenamiento de terceros llamada ObjectStore. Sin embargo, la finalidad de un sistema en la nube independiente es no tener que depender de un almacenamiento de datos externo. Para abordar esta cuestión, modifica la configuración PHP de ownCloud para que coincida con la que se muestra a continuación:

```
$ sudo medit /usr/share/nginx/html/ownCloud/config/
config.php

'logtimezone' => 'America/Los Angeles',
'log_rotate_size' => 1048576000, // ~1GB
'openssl' =>
array (
    'config' => '/usr/lib/ssl/openssl.cnf',
),
'mount_file' => 'oc-data/mount.json',
```

A continuación, suprime la entrada de objectstore buscando la etiqueta “objectstore”, tras localizarla elimínala. Guarde el archivo y reinicia el sistema. Una vez que ODROID se haya reiniciado, puedes crear carpetas específicas para almacenar diferentes tipos de archivos. He creado las siguientes carpetas y luego, he subido algunos contenidos de ejemplo en cada una de ellas con la finalidad de verificar que funciona ownCloud:

- pix: para almacenar fotografías
- audio: para almacenar archivos mp3
- video: para almacenar archivos mp4 y flv

La figura 13 muestra una ventana pop-up donde se reproduce un archivo de vídeo por medio del reproductor nativo y directamente en la sesión del cliente ownCloud.

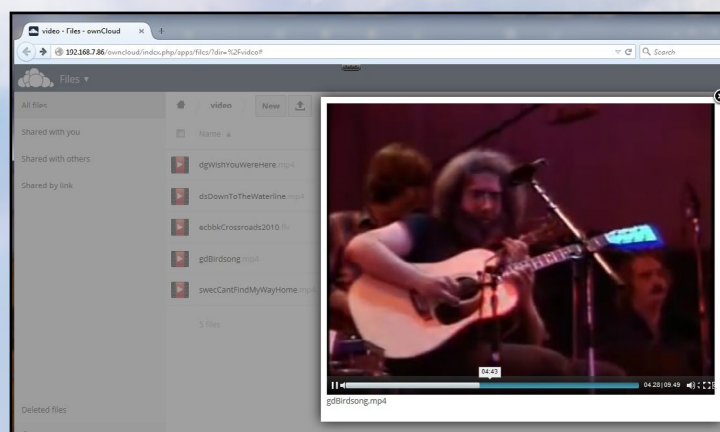


Figura 13: Cliente ownCloud reproduciendo un vídeo

Para comprobar que todo se ha instalado correctamente, puedes realizar una última comprobación para ver si el sistema XU3 en realidad, tiene presente los archivos y se sirve de ellos con los siguientes comandos:

HEAVENSTRIKE RIVALS

UN CLASICO RPG PARA LOS AMANTES DEL FINAL FANTASY COMO NOSOTROS

por Bruno Doiche

A king del género RPG, Square Enix es un editor comercial que tiene horas y horas de nuestras vidas consumidas con todos sus juegos, por lo que si estás ejecutando nuestra última versión de Android en tu ODROID, no te pierdas la oportunidad de echar una ojeada a Heavenstrike Rivals.

Disfruta de un sistema de combate diseñado especialmente para dispositivos móviles fácil de aprender, pero con grandes posibilidades estratégicas, rápido combate jugador-contrajugador y cientos de personajes únicos para coleccionar, crecer y evolucionar.

¡Desafia al mundo y sube de ranking hasta que lo gobiernes todo!

<http://bit.ly/1BKpbrc>



```
$ su
# cd /oc-data/ocadmin
# find . -name "*"
.
./files
./files/audio
./files/audio/gdPeggyO.mp3
./files/audio/rre1759.mp3
./files/audio/jgBrokedownPalace.mp3
./files/video
./files/video/dgWishYouWereHere.mp4
./files/video/swecCantFindMyWayHome.mp4
./files/video/gdBirdsong.mp4
./files/video/ecbbkCrossroads2010.flv
./files/video/dsDownToTheWaterline.mp4
./files/pix
./files/pix/acharipicb.png
...
```

¿No es divertido? Bueno, ahí lo tienen: uno de los más rápidos y más reducidos sistemas owncloud escalables de alta capacidad. Lo mejor de todo, al igual que ocurre con casi todos los proyectos basados en ODROID, es que puedes llevártelo en el bolsillo de la camisa.

Configuración adicional

- Activar y testear el acceso https
- Mejorar la configuración utilizando la información de <http://bit.ly/1tshZRW>
- Desarrollar un cron job utilizando rsync para hacer copia de seguridad de la configuración ownCloud, los datos y la información importante de la base de datos
- Profundizar en las aplicaciones integradas en ownCloud, tales como apps de terceros que instalan calendarios para ser alojados en ownCloud.
- Utilizar opciones de almacenamiento externos como Google Docs o ObjectStore Swift
- ¡Y mucho más!!

Para obtener más información o realizar preguntas sobre ownCloud, por favor visita las fuentes de información originales haciendo clic en cualquiera de los siguientes enlaces:

- <http://bit.ly/13Nlfeu>
- <http://bit.ly/1H8B8uo>
- <http://bit.ly/13NwWlo>
- <http://bit.ly/1rtFE33>
- <http://bit.ly/1tshZRW>
- <http://bit.ly/1kssJLF>
- <http://bit.ly/1D1R7s6>
- <http://bit.ly/1JWN0C6>
- <http://bit.ly/1Ez6ZXy>

REPOSITORIO DE KERNEL MANTENIDO POR USUARIOS

GESTIONA TUS PAQUETES DE SOFTWARE CON ACTUALIZACIONES AUTOMÁTICAS USANDO APT-GET

por Tobias Schaaf

Desde hace algún tiempo, he estado desarrollando paquetes de Debian para facilitar la instalación de juegos y programas que exporto a ODROID, y son almacenados en el servidor patrocinado por Hardkernel en <http://bit.ly/13v98ly>. Desde este repositorio puedes descargar e instalar manualmente muchos paquetes de software, lo que te ahorra la molestia de tener que recompilarlos por tu cuenta. Sin embargo, he mejorado recientemente la forma en la que estos programas pueden ser instalados en tu sistema local.

He estado experimentando con un simple repositorio de Debian que permitirá la instalación de paquetes con un simple comando “apt-get install”, y la actualización de paquetes ya instalados con el comando “apt-get upgrade”. He desarrollado recientemente un contexto donde esto es posible, incluso podemos actualizar los kernels con el comando “apt-get upgrade”, que es una función que a priori no estaba disponible desde Hardkernel.

Quiero compartir la configuración actual de mi primer repositorio ODROID con las personas que deseen probarlo. Probablemente añadiré más repositorios siempre que lo vea necesario.

Empecemos

Ten en cuenta que los siguientes pasos debe hacerse como root escribiendo “sudo su” y escribiendo la contraseña de

administración, que suele ser “odroid”. En primer lugar, ve al directorio /etc /apt/ sources. list.d/ de tu distribución. Una vez en el directorio, puede descargar cuantos archivos quieras del repositorio:

```
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-main.list
# main package list for all ODROIDS and all Distributions (Debian/Ubuntu)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-U.list
# package list for ODROID U2/U3 devices and all Distributions (Kernel and Headers)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-X.list
# package list for ODROID X devices and all Distributions (Kernel and Headers)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-X2.list
# package list for ODROID X2 devices and all Distributions (Kernel and Headers)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-XU.list
# package list for ODROID XU devices and all Distributions (Kernel and Headers)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-XU3.list
# package list for ODROID XU3 devices and all Distributions (Kernel and Headers)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-C1.list
# package list for ODROID C1 devices and all Distributions (Kernel and Headers)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-all-testing.list
# package list all ODROID devices and all Distributions unstable packages
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-wheezy-main.list
# package list for all ODROID devices but for Debian Wheezy (ex. not for Ubuntu 14.04)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-wheezy-backports.list
# package list for all ODROID devices but for Debian Wheezy (backports of libraries ex. SDL2)
$ wget http://oph.mdrjr.net/meveric/sources.lists/meveric-wheezy-testing.list
# package list for all ODROID devices but for Debian Wheezy (packages for testing ex. XBMC 13)
```

Por favor, asegúrate de que sólo descargas los paquetes adecuados para tu dispositivo. Las actualizaciones del kernel para ODROID-X2 no funcionarán en el ODROID-U3.

A continuación, es necesario descargar e instalar mi clave de firma para decirle al programa apt que los paquetes firmados con esa clave están listos para usarse:

```
$ wget -O- http://oph.mdrjr.net/meveric/meveric.asc | apt-key add -
```

Después, es necesario actualizar la lista de paquetes con el siguiente comando:

```
$ apt-get update
```

Si los has hecho todo bien, se debe ejecutar sin ningún problema, lo que significa que ahora estás listo para actualizar e instalar los paquetes a través del comando “apt-get”

Actualizar el kernel usando apt-get

Una de las funciones más útiles del repositorio de Debian es la opción de actualizar tus Kernels de forma automática a través de las actualizaciones del sistema. Para ello, he creado un “meta-package” que garantizará que tu Kernel siempre estará actualizado. Voy a utilizar el ODROID-U como ejemplo, pero es igual para ODROID-X y ODROID-X2, también para otros modelos más modernos.

En primer lugar, asegúrate de que tienes lo siguiente en tu archivo /etc/apt/sources.list.d/meveric.list, y que ya has ejecutado el comando “apt-get update” para descargar la lista de paquetes más reciente:

```
$ deb http://oph.mdrjr.net/meveric/ all u
```

A continuación, puedes instalar el siguiente meta-package para conseguir todas las actualizaciones que ofrezco del kernel:

```
$ apt-get install linux-headers-armhf-odroid-u
$ apt-get install linux-image-armhf-odroid-u
```

El paquete contiene los archivos cabecera para el kernel, que a veces son necesarios si deseas compilar tus propios módulos del kernel, como cuando instalamos a una tarjeta de sonido externa o un periférico similar. El paquete contiene el kernel y los módulos en sí. Con estos paquetes instalados, consigues el kernel más reciente y puedes actualizarlo automáticamente usando “apt-get”.

Notas

Una vez que ya tengas instalado uno de mis kernels y siempre recibas una actua-

lización, el kernel y los archivos cabecera anteriores serán desinstalados antes de instalar el nuevo kernel. El sistema te avisará de ello, puesto que vas a desinstalar el kernel que actualmente se está ejecutando y te preguntará si deseas detener esta operación. Tiene que contestar que “no” para que continúe la instalación.

Ten cuidado con este paso, ya que tras eliminar el kernel no debe reiniciar el ODROID hasta que se instale el nuevo kernel, puesto que tu ODROID no arrancará sin un kernel. Pero no te preocupes, tu ODROID funcionará indefinidamente hasta que reinicies, incluso durante días y semanas, lo que te da tiempo de sobra para solucionar cualquier problema. Si algo sale mal y el sistema se reinicia, todavía es posible reparar la instalación utilizando otro equipo.

Si ya tienes instalado un kernel creado por mí, no tiene que preocuparse, ya que los pasos anteriores deberían funcionar correctamente. Si tienes un Kernel de Hard-Kernel inferior a la versión 3.8.13.26, debería funcionar también. Sin embargo, si la versión de tu kernel es la 3.8.13.26 (escribe uname -a para ver tu versión del kernel), entonces tienes que limpiar en primer lugar el directorio /boot, ya que el paquete del kernel puede contener los mismos ficheros que los que ya están copiados en tu directorio boot, lo cual impediría continuar con la instalación.

```
$ rm -f /boot/*-3.8.13.26 for U3
$ rm -f /boot/*-3.10.51 # for XU3
```

Lista de paquetes

Esta sección contiene una lista de los paquetes que puedes encontrar en mi repositorio a partir de enero de 2015. La lista será actualizada sin previo aviso, así que asegúrate de visitar periódicamente el hilo del foro que aparece al final de este artículo si quieres saber si se han añadido nuevos paquetes.

Nombre

```
linux-headers-armhf-odroid-u
linux-image-armhf-odroid-u
linux-headers-armhf-odroid-x
linux-image-armhf-odroid-x
linux-headers-armhf-odroid-x2
linux-image-armhf-odroid-x2
linux-headers-armhf-odroid-xu
linux-image-armhf-odroid-xu
linux-headers-armhf-odroid-xu3
linux-image-armhf-odroid-xu3
linux-headers-armhf-odroid-cl
linux-image-armhf-odroid-cl
armagetronad-odroid-launcher
chromium-bsu-odroid
eduke32-odroid
emulationstation-odroid
freedroidrpg-odroid
hedgewars-odroid-launcher
libgl-odroid
libglew-odroid
libglues-odroid

mario-odroid
neverball-odroid-launcher
neverputt-odroid-launcher
opencpn-odroid
shmupacabra-odroid
smc-odroid
supertux2-odroid
supertuxkart-odroid-launcher
sw-odroid
valyriatear-odroid
yquake2-odroid
clementine-odroid
mono-odroid
retroarch-odroid

xf86-video-armsoc-odroid
```

Descripción

```
Meta Package for Kernel Headers of U devices
Meta Package for Kernel Image of U devices
Meta Package for Kernel Headers of X devices
Meta Package for Kernel Image of X devices
Meta Package for Kernel Headers of X2 devices
Meta Package for Kernel Image of X2 devices
Meta Package for Kernel Headers of XU devices
Meta Package for Kernel Image of XU devices
Meta Package for Kernel Headers of XU3 devices
Meta Package for Kernel Image of XU3 devices
Meta Package for Kernel Headers of C1 devices
Meta Package for Kernel Image of C1 devices
Meta Package for glshim version of Armagetron
Up-Down Shooting game using glshim
Remake of Duke Nukem 3D using glshim
A graphical and themeable emulator front-end
Diablo game with Tux using glshim acceleration
Worms-like action game using glshim
glshim OpenGL -> OpenGL ES wrapper
libGLEW linked against glshim (for some games)
libGLU for OpenGL ES linked against glshim
(needed for some games)
Super Mario and Valves Portal mixed using glshim
3D Puzzle game using glshim
3D Puzzle/Golf game using glshim
Naval map and route using glshim
A hard and fast arcade shooter using glshim
Super Mario Chronicles, using glshim
Super Mario Clone with Tux using glshim
Run SuperTuxCart (3D Mario Kart clone) glshim
Shadow Warrior clone using glshim and OpenGL
Very nice looking RPG game
Quake 2 remake in OpenGL using glshim
Music Player to organize your music and streams
Mono (C#) lib and dev files for Debian Wheezy
Retroarch Frontend for Libretro cores
(Multi System Emulator)
ARMSoc framebuffer drivers for Mali GPUs used
for Exynos 4412 series
```

xbmc-odroid	XBMC Gotham 13.2 for Debian Wheezy
antimicro-odroid	Tool for mapping keyboard events to gamepads and joysticks
clipgrab	Tool to download movies from online websites, such as youtube or dailymotion
corsixth-odroid	Theme Hospital Clone (very funny Hospital simulation)
dlx-rebirth-odroid	Descent 1 Rebirth OpenGL ES version
d2x-rebirth-odroid	Descent 2 Rebirth OpenGL ES version
desmume-odroid	Nintendo DS/i Emulator
doom3-odroid	Famous 3D First Person Shooter
dosbox-odroid	ARMv7a optimized version of DOS Emulator
dunelegacy	Dune 2 remake with enhanced features using SDL
etr-odroid	Extreme Tux Racer OpenGL ES version
fheroes2-odroid	Heroes of Might and Magic 2 remake
flare-engine-odroid	Free/Libre Action Roleplaying Engine
ffmpeg-odroid	A complete, cross-platform solution to record, convert and stream audio and video
frogatto-odroid-720	Very good looking jump platformer where you play as a frog using GLES1 (720p binary)
frogatto-odroid-1080	Very good looking jump platformer where you play as a frog using GLES1 (1080p binary)
fs-uae	Amiga Emulator with OpenGL ES 1 support
homeworldsdl-odroid	Port of the famous Real Time Space Strategy game Homeworld with OpenGL ES support
hurrican-odroid	Remake of the classic Turrican using OpenGL ES for lots of special effects
ioquake3-odroid	Open Source Quake 3 remake for OpenGL ES
ja2-stracciatella	Jagget Alliance 2 remake in SDL, allows to replay JA2 on your ODROID in FullHD
jk3-odroid	Jedi Knight 3 - Jedi Academy for OpenGL ES
libsodium-odroid	easy-to-use encryption and decryption library
mednafen-odroid	A MultiSystemEmulator which allows you to play GBA, NES, and many other console games
openggs-odroid	Great Giana Sisters remake.. C64 version, as well as total remake with different levels
openomf-odroid	Open Source remake of One Must Fall 2097
opentyrian-odroid	Arcade Shooter
openxcom-odroid	UFO: Enemy Unknown (X-COM: UFO Defence) remake with high resolution and new features
ppsspp-odroid	PlayStation Portable Emulator
retroarch-cores-good	Libretro cores for retroarch used in GameStation Turbo Image
retroarch-cores-bad	Additional libretro cores for retroarch not used in GameStation Turbo Image
rickyd-odroid	Rick Dangerous Clone using SDL2
scummvm-odroid	ScummVM Engine for multiple Adventure games
smw-bin	Super Mario War - A fighting/Jump and Run inspired by Super Mario
smw-leveledit	Level Editor for Super Mario War
toppler-odroid	Toppler Tower is a Nebulus Clone in SDL
uqm-hd-odroid	Ur-Quan Master HD / HD remake of Ur-Quan Master (Star Control 2)
vcmi-odroid	Heroes of Might and Magic III Engine to play HoMM3 on the ODROID

Si tienes alguna pregunta sobre el repositorio, no dude en escribir en el post original de los foros ODROID, <http://bit.ly/1wEbfzC>, y trataré de ayudarte siempre que pueda. Parece complicado, pero una vez que el repositorio esté configurado correctamente, podrás instalar y actualizar programas, juegos y kernel con un simple comando “apt-get”.

ACTUALIZA DESDE LA 13.10 A LA 14.04 QUEDATE TRANQUILO HASTA ABRIL DEL 2019 CON UNA VERSION LTS

por Rob Roy

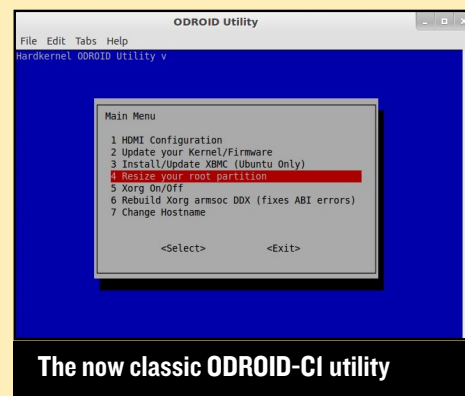
He creado un extenso repositorio público de imágenes Ubuntu 13.04 y 13.10 durante el año pasado, y he recibido varias peticiones para actualizarlas a Ubuntu 14.04. Debido a que mi librería contiene más de 100 GB de software y distribuciones, he decidido compartir las instrucciones para actualizar cualquiera de mis imágenes a Trusty Thar, que se puede hacer fácilmente desde la línea de comandos. Aunque Ubuntu 13.10 ya no recibe actualizaciones de software y de seguridad, yo personalmente todavía utilizo las versio-

nes 13.10 debido a su estabilidad y fiabilidad, pero Ubuntu 14.04 ofrece soporte hasta abril de 2019, así que tiene sentido actualizar a esta versión.

Para empezar, descargar cualquiera de mis imágenes Ubuntu 13.10 desde el servidor Hardkernel en <http://bit.ly/1rhHymu>, copiala en un eMMC o tarjeta SD y arráncala. El primer paso es descargar el script ODROID Utility:

```
$ sudo -s
$ wget -O /usr/local/bin/odroid-utility.sh \
https://raw.githubusercontent.com/mdljr/odroid-utility/master/odroid-utility.sh
$ chmod +x /usr/local/bin/odroid-utility.sh
$ odroid-utility.sh
```

Selecciona la opción para cambiar el tamaño de la partición raíz, lo que requerirá un reinicio. Una vez que el escritorio aparece de nuevo, selecciona “Software Updates” en el menú Aplicaciones o es-



cribe “sudo do-release-upgrade” en la ventana de Terminal. Siga las instrucciones para actualizar pero no reinicies cuando te lo pregunte. Ejecuta ODROID Utility una vez más y actualiza el kernel, firmware y los drivers de video.

Una vez que ODROID Utility haya finalizado la actualización, reinicia y verifica que el nuevo sistema operativo ha sido instalado escribiendo en Terminal:

```
$ lsb_release -a
Distributor ID: Ubuntu
Description: Ubuntu 14.04.1 LTS
Release: 14.04
Codename: trusty
```

JUEGOS LINUX

COMPARANDO LA POTENCIA DE JUEGO DEL U3 FRENTE AL XU3

por Tobias Schaaf



Recientemente, he conseguido que una versión de mi popular imagen de juegos ODROID GameStation Turbo funcione en el ODROID-XU3. Aunque no es perfecta, ofrece una experiencia de usuario similar a la versión U3. Ahora que tengo la misma imagen ejecutándose tanto en el U3 como en el XU3, es el momento de comparar el potencia de ambas placas para determinar cuál de las dos tiene un mejor rendimiento como plataforma de juegos, y que fallos puedo encontrar.

Introducción

Obviamente, la ODROID-XU3 tiene más puertos USB2 y un puerto USB3, que es una gran ventaja sobre la U3. Además, el bus de hardware y el módulo eMMC son mucho más rápido en el XU3. He usado el comando “hdparm -tT /dev /mmcblk0” para probar la velocidad del eMMC, muestra que la velocidad de lectura es de 80 a 90 MB/seg con un promedio de 84 MB/seg en el XU3. Después creé un archivo de 4 GB con el comando “pv/dev/urandom > test.file” para evaluar la tasa de escritura, mostro una velocidad de unos 4,2 MB/seg, que no está nada mal, teniendo en cuenta que son datos generados de forma aleatoria. Tras crear el archivo, realice otra prueba enviando el archivo a /dev/null con el comando “pv test.file> /dev/null”, pero esta vez los resultados

fueron un tanto diferentes: reporto una velocidad de 117 MB/seg. Hice la misma prueba en otro XU3 diferente con un eMMC diferente (uno era de 64 GB y el otro de 16 GB) con el mismo resultado exacto. Rehíce la prueba en mi ordenador portátil Linux con un disco duro SATA normal y llegue a los 95 MB/s, con valores que a veces descendieron por debajo de los 80 MB/seg. El ODROID ofrece unos 115-117 MB/s constantes, por lo que la velocidad de lectura es muy buena. Después, probé la velocidad de escritura del módulo eMMC con el comando “pv/dev/zero > test.file”, mostró un promedio de alrededor de 30 MB/s con picos de hasta 35 MB/seg.

Notas

Aunque no he hecho la misma prueba de lectura/escritura en el U3, puesto que la velocidad del eMMC ya es conocida por pruebas anteriores, he creado un archivo de prueba con diversos datos. Observé que la velocidad de creación del archivo era casi la misma. El U3 creó el archivo a 4 MB/seg, tan sólo un 5% más lento que en el caso del XU3. Sin embargo, en la XU3 la temperatura de la CPU subió a 67-70 °C con el ventilador girando constantemente a gran velocidad, mientras que el U3 se ejecutaba a 50 °C sin iniciar el ventilador.

En términos generales, el XU3 es más ruidoso que el U3. Incluso al ralentí, el

XU3 nunca baja de los 55 °C y en el momento en el que inicias una tarea, uno de los núcleos se acerca al 100% y la temperatura alcanza los 65 °C al instante.

La velocidad de escritura de XU3 es más bien lenta si la comparamos con su excelente velocidad de lectura, con una buena tarjeta microSD es probable que puedas lograr la misma velocidad de escritura que en eMMC. Esto también significa que incluso si utilizamos el adaptador LAN GigaBit USB3 nunca conseguiremos más de 30 MB/seg al copiar un archivo por red.

También son divertidos (aunque no más que la prueba de rendimiento) los resultados del comando “pv /dev/zero > /dev/null”:

- XU3 alcanza 3.2GB/seg con la CPU a 75 °C y el ventilador girando a toda velocidad
- U3 alcanza 3.5GB/sec con la CPU a 50 °C sin que el ventilador gire
- XU-Lite alcanza 2.4GB/seg con la CPU a 56 °C sin que el ventilador gire

Otra cosa que observé es que el sistema operativo del XU3 es un poco inestable. XBMC tiende a bloquear el XU3 al cambiar entre programas o películas con demasiada frecuencia. Incluso los juegos que se ejecutan desde el escritorio pueden causar que el XU3 falle o se cuelgue de vez en cuando. Aunque el rendimiento de la XU3 es muy bueno en

general, preenta un punto débil, por lo que debes colocar el XU3 de manera que puedas reiniciarlo con facilidad.

Rendimiento OpenGLES

La siguiente prueba que hice fue ejecutar la demo glmark2-es2 para ver cómo funciona el nuevo Mali T628 del XU3 comparado con el Mali 400 del U3. Me sorprendí al ver que glmark2-es2 detecta el OpenGLES 3.0 y fue capaz de ejecutar todas las pruebas mientras que el U3 tuvo problemas en algunas.

También me sorprendieron los resultados. En algunos casos, el ODROID-U3 fue 5-10 FPS más rápido que el XU3, pero los resultados variaban mucho. De hecho, ambos dispositivos eran más lentos de lo que deberían ser, pero no estoy seguro de cuál es el problema. Aunque el U3 muestra un promedio de 67 FPS, llegando a los 79 FPS. He visto que el rendimiento del éste ha sido mucho mejor en el pasado, con valores de hasta 109 FPS usando la misma prueba de rendimiento. Así que algo está ralentizando esta prueba, aunque es probable que no afecte a la experiencia en conjunto.

Por otro lado el XU3 muestra un promedio de 66 FPS con picos de hasta 73 FPS, pero también he visto valores de 140 FPS que indican que el T628, en realidad, debería tener mejor rendimiento. Otra anomalía es que el XU3 es incapaz de ejecutar la prueba glmark2-es2 en modo pantalla completa, dando como resultado una imagen fija. Sin embargo, las pruebas parecen ejecutarse en segundo plano con valores de más de 1.500 FPS.

Además note que ejecutando la prueba en modo ventana, sin tener ventanas en segundo plano, los resultados eran los mismos con más de 1.500 FPS. También quise ejecutar juegos OpenGLES nativos y comparar su velocidad, así que utilicé los que son más exigentes con el hardware. Los resultados se muestran a continuación:

Doom 3

Doom 3 tiene una demo obsoleta que se puede usar para testear el rendimiento de tu hardware. La demo ejecuta un nivel con diversos monstruos y con un montón de efectos. Calcula el tiempo que el juego necesita para completar la demo y proporciona el promedio de FPS. En el XU3, el juego tiene algunos problemas menores. Al tiempo que se ejecuta, aparecen algunos problemas técnicos en forma de fisuras por alguna causa desconocida. Pero incluso con estos problemas, el juego es muy jugable obteniendo un resultado final de 29 FPS, mientras que el U3 alcanza 24,5 FPS sin ninguno de los fallos que aparecen en las pruebas con el XU3. No estoy seguro si los fallos son simplemente problemas de interpretación o si en realidad están afectando al rendimiento, pero incluso con estas fisuras, el XU3 ejecuta este juego un 18% más rápido que el U3.

Extreme Tux Racer, Homeworld, Jedi Knight 3, Frogatto y UFO-AI

No he sido capaz de encontrar un contador de FPS para Extreme Tux Racer, pero puedo decir que el juego se ejecuta a toda velocidad tanto en el U3 y XU3, pero en el XU3 tiene un problema con el vídeo cada vez que se mueve la cámara. Esto está presente en todos los juegos que he intentado ejecutar de forma nativa bajo OpenGLES, incluso usando glshim. Homeworld, que utiliza OpenGLES 1.1 se ejecuta muy bien. Las fisuras en el XU3 aun así está presente pero son casi imperceptibles, ya que la cámara nunca gira lo suficientemente rápido como para que sean visible. Jedi Knight-3 se inició en el XU3, pero no fue capaz de que mostrara en ventana, sólo se ejecuta el audio del juego. Frogatto también muestra el error de fisuras, pero funciona perfectamente. El XU3, de hecho, corrige un problema con la transparencia presente en el U3. El agua se ve mejor en la XU3 que en el U3.

UFO-AI me sorprendió al ejecutarse muy bien en el XU3. El U3 tiene problemas con este juego, que en mi opinión, están relacionados con el buffer de textura. Hay que reducir bastante los gráficos para poder jugar y en algunos casos éstos fallan y el juego se cuelga. Sólo se puede jugar usando una resolución baja de 256x256 píxeles. Si tiene suerte, puedes usar 512x512, pero los problemas gráficos aparecerán pronto. El XU3 puede alcanzar mapas de 1024x1024 píxeles y parece que funciona bien. Pero a partir de 2048x2048, el rendimiento disminuye en gran medida cuando se utiliza el modo Battlescape, mientras que el contador de FPS se mantiene en 50 FPS constantes al tiempo que se usa el menú del juego y el modo globescape. El U3 presenta muchos más problemas que el XU3 cuando se juega UFO-AI.

Rendimiento GLSHIM

Glshim es un contenedor para OpenGL que permite ejecutar ciertos juegos OpenGL en dispositivos OpenGLES como ODROID. Sólo es compatible con OpenGL 1.x por ahora y no todas las funciones están disponibles. Hay algunos juegos que se pueden ejecutar utilizando OpenGL, pero en realidad hay muchos que funcionan correctamente. Por lo tanto, glshim es una buena prueba de rendimiento, especialmente porque algunos de los programas tienen altas demandas de hardware.

Eduke32, Super-Tux2, Chromium B.S.U., Hedgewars y Secret Maryo Chronicles

Eduke tiene algunos problemas con la XU3. Por ejemplo, cuando se ejecuta en pantalla completa con la misma resolución del escritorio, aparece un error EGL y no se ve nada. Sin embargo, puedo ejecutarlo en modo ventana, pero la tasa de fotogramas desciende a 27FPS. Cuando uso una resolución diferente a la

resolución del escritorio, el juego se inicia con una imagen ligeramente fuera de lugar, pero presenta 40- 49 FPS con un promedio de alrededor de 47 FPS. En el U3, el juego se ejecuta sin problemas con unos 60FPS estables. resolution for the game than the desktop resolution, the game starts with a slightly misplaced image, but it holds about 40-49 FPS with an average of about 47 FPS. On the U3, the game runs without issues at a steady 60FPS.

SuperTux 2 has acceptable performance on both devices. The XU3 suf- del escritorio, el juego se inicia con una imagen ligeramente fuera de lugar, pero presenta 40-49 FPS con un promedio de alrededor de 47 FPS. En el U3, el juego se ejecuta sin problemas con unos 60 FPS estables.

SuperTux 2 tiene un rendimiento aceptable en ambos dispositivos. El XU3 tiene el problema de las fisuras al desplazarte por la pantalla, pero la versión para U3 funciona muy bien. En el U3, este juego se ejecuta con un promedio de 68 FPS, mientras que en el XU3 se ejecuta entre los 58 y 62 FPS.

Mi versión parcheada de Chromium B.S.U. funciona muy bien en la U3 a 1080p con alrededor de 50 FPS, aunque durante el jugo el FPS disminuye lentamente. En el XU3, la tasa de fotogramas puede alcanzar los 55 FPS, pero a veces baja a 44 FPS. El juego sigue siendo muy jugable, pero tiene los mismos problemas que encuentro en otros juegos, el juego no se ejecuta en pantalla completa con la misma resolución del escritorio.

Hedgewars tampoco funciona en pantalla completa con la resolución del escritorio. Al seleccionar una resolución diferente se obtiene una tasa de 45 a 49 FPS, con una resolución de 1360x786 en modo de ventana. Entre 22 y 27 FPS se pueden alcanzar con el XU3, mientras que en el U3 se ejecuta a pantalla completa 1920x1080 a una velocidad constante de 60 FPS. En el modo de ventana muestra 40 FPS.

Secret Maryo Chronicles no tiene

contador de FPS, así que mis impresiones se basan en la apariencia. El U3 lo ejecuta a unos impresionantes 1080p con todo detalle y se ve muy suave. Utilizando el modo de ventana a una resolución de 1360x768 sigue viéndose bien, pero se puede sentir que el juego va algo forzado. En el XU3, aún existe el problema de resolución de pantalla mencionado anteriormente, pero el rendimiento es aceptable. De hecho, en el modo ventana se percibe algo más rápido en el XU3 que en el U3.

Coclucción OpenGLES

Me quedé decepcionado con el rendimiento OpenGLES del XU3. Pensaba que sólo era un problema con OpenGLES 1.1, ya que Doom 3 se ejecutaba más rápido que en el U3, podría estar relacionado con la potencia de la CPU del XU3. Por desgracia, sólo hay pocos juegos que utilizan OpenGLES 2.0 o incluso 3.0 en Linux, por lo que resulta difícil compararlos entre sí.

Rendimiento SDL

BLITTING TEST	393.5
PLASMA	370.9
ROTDODNER	363.8
ROTDODNER NEAR	371.7
ROTDODNER FAR	356.5
RADIAL SLUR	209.6
3D BUNNY	62.8

GPMark con 320x240 en XU3

BLITTING TEST	36.7
PLASMA	32.4
ROTDODNER	32.9
ROTDODNER NEAR	33.7
ROTDODNER FAR	32.0
RADIAL SLUR	47.7
3D BUNNY	43.0

GPMark con 320x240 en U3

BLITTING TEST	340.9
PLASMA	287.7
ROTDODNER	312.8
ROTDODNER NEAR	313.0
ROTDODNER FAR	310.0
RADIAL SLUR	149.0
3D BUNNY	36.8

GPMark con 640x480 en XU3

BLITTING TEST	393.5
PLASMA	370.9
ROTDODNER	363.8
ROTDODNER NEAR	371.7
ROTDODNER FAR	356.5
RADIAL SLUR	209.6
3D BUNNY	62.8

GPMark con 640x480 en U3

Puesto que OpenGLES mostro escaso rendimiento en mis experimentos, me quedé con ganas de probar el rendimiento de SDL, ya que su velocidad depende en su mayoría de la potencia de la CPU. He de suponer que SDL debe ser mejor en el XU3 que en el U3.

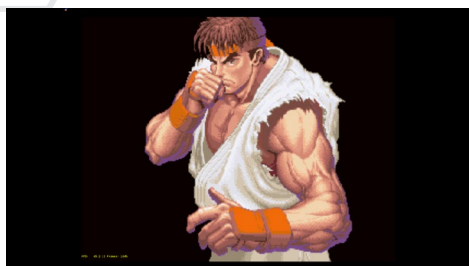
Ejecute algunos juegos como Jagged Alliance 2, Dune Legacy y freedroid RPG, así como también una prueba de rendimiento SDL llamada gpmark para comparar el rendimiento del XU3 frente al U3. Como sospechaba, el rendimiento del XU3 es superior al del U3. Incluso se resuelve el problema con la resolución de pantalla completa que me encontré con aplicaciones OpenGLES, lo que significa que puedo ejecutar juegos con la misma resolución que la del escritorio con SDL.

Emuladores

Los emuladores son también una prueba muy buena para ver el rendimiento, ya que a menudo necesitan bastante CPU junto con algo de potencia gráfica para poder emular diferentes sistemas. Un buen ejemplo es Retroarch, que utiliza diferentes tecnologías para emular diferentes sistemas como el SNES, NDS, GBA, 3DO y muchos otros. Utiliza OpenGLES 2.0 para mostrar el contenido mediante aceleración por hardware, pero también utiliza SDL para trazar el contenido, OpenAL para el sonido y udev para la entrada de controladores. Tantas tecnologías diferentes trabajando juntas necesitan de un exigente hardware, por lo que se espera que el XU3 tenga un mejor rendimiento al tener una CPU más potente.

Emulación 3DO Retroarch

Elegí unos cuantos motores de Ret-



La Intro de Street Fighter II muestra 49,2 FPS en el XU3 frente a 29 FPS del U3



La portada del Street Fighter II muestra 58,7 FPS en el XU3 frente a 42,2 FPS del U3



La jugabilidad del Street Fighter II muestra 48,8 FPS en el XU3 frente a 27,8 FPS del U3

roarch que consume mucha CPU, y uno de los complementos más recientes para Retroarch, el emulador de 3DO que generalmente es muy exigente con la CPU. Use Super Street Fighter II para probar el rendimiento de ambos dispositivos. Por alguna razón, fui incapaz de tomar capturas de pantalla directamente desde el U3 así que sólo puedo incluir imágenes del XU3.

Emulación NDS Retroarch

Aunque NSD no es el motor más reciente disponible para Retroarch, tiene algunas imágenes en 3D que puede ser muy exigentes. Con el compilador JIT recientemente añadido para las placas ARM, La emulación NDS es casi perfecta en todos los ODROIDs con cierto margen de mejora. Use Rune Factory 3 y Bleach the 3rd Phantom para las prue-



La Portada del 3 del Rune Factory 3 muestra casi 60 FPS tanto el XU3 como en el U3

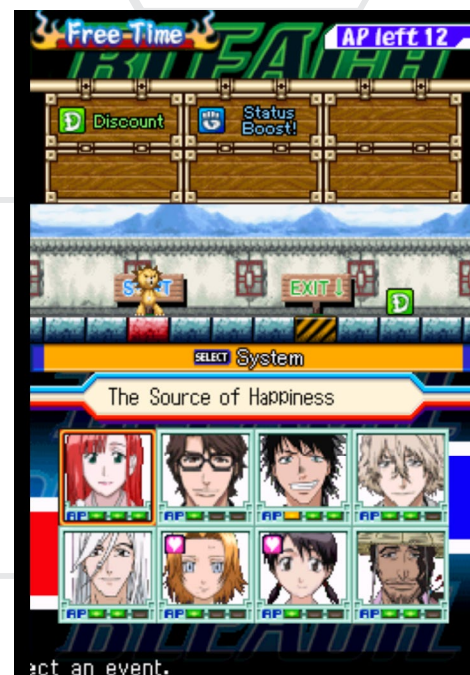


La jugabilidad del Rune Factory 3 muestra 57,5 FPS en el XU3 frente a 45 FPS en el U3

bas. Rune Factory 3 utiliza rasgos 3D y Bleach tiene un pesado funcionamiento en segundo plano, ambos consumen mucha CPU.

Phantom, donde se puede elegir hablar con amigos y aliados. En segundo plano, el logotipo de Bleach en constante movimiento consume un montón de CPU, proporcionando 50 FPS en el XU3 frente a los 42 FPS en el U3.

Podría haber realizado más pruebas con el emulador NDS, pero está bas-



Una escena en Bleach el 3rd Phantom, donde se puede elegir hablar con amigos y aliados. En el fondo, el logotipo de Bleach es constante movimiento que consume un montón de CPU, emitiendo 50 FPS en el XU3 frente a 42 FPS del U3

tante claro que Retroarch tiene mejor rendimiento en el XU3 que en el U3, lo cual no me sorprende ya que la CPU del XU3 es mucho más potente que la CPU del U3. La CPU superior puede llegar a tener hasta un 50% más de velocidad, con una media de 20 a 25% más de velocidad que en el caso del U3.

Emulación FS-UAE Amiga

Aunque FS-EAU depende en gran medida de OpenGL, funciona sorprendentemente bien en el XU3. No descubrí ningún problema y el rendimiento es muy bueno. Elegí un juego más exigente para esta prueba llamado Banshee. La versión AGA requiere más

Banshee es un gran juego multi-jugador cooperativo para Amiga



memoria y una CPU más rápida.

Puse ambos ODROIDs a emular un Amiga A1200 con una CPU 68020 a la mayor velocidad posible, incluyendo la memoria chip de 2 MB y la memoria rápida de 4 GB usando Kickstart 2.04. El XU3 gana una vez más al U3 por su potente CPU. Mientras que el juego en el XU3 funciona bien en toda velocidad sin retardos, el juego es muy lento en el U3, con cortes de sonido y pésima jugabilidad.

Emulación PPSSPP Playstation Portable

PPSSPP es uno de los mejores emuladores disponibles, el rendimiento y gráficos que ofrece son simplemente impresionantes, te permite ejecutar grandes juegos 3D en alta resolución. Para esta prueba, he deshabilitado la opción frame



Asphalt Urban GT2 en el ODROID-XU funciona genial con alrededor de 30 a 35 FPS en el XU3, y sólo consigue alrededor de 13 FPS en el U3, pero puede funcionar mejor en el U3 con la configuración correcta

skip para ver la tasa de frames real y use una resolución de 2x.

Jugué a Asphalt Urban GT2 y a Ragnarok Tactics, ya que tradicionalmente he usado estos juegos para probar el rendimiento del emulador PPSSPP cuando desarrollaba mis imágenes GameStation Turbo. Asphalt Urban GT2 es muy exigente con el hardware Y de hecho se vuelve lento si se aumenta el salto de frames. También tiene otros problemas de degradación que en otros juegos, por el contrario aumenta el rendimiento.

Ragnarok Tactics es un gracioso juego RPG/TBS al estilo anime en el mundo



Tácticas Ragnarok se ejecuta con unos constantes 60 FPS en el XU3 y 57 FPS en el U3 con cascadas en pantalla y muchos modelos poligonales

de Ragnarok Online. En la época en la que era difícil conseguir que PPSSPP funcionase correctamente en ODROID, era uno de los primeros juegos con el que lo intenté. Había diferentes tipos de errores en el funcionamiento del juego, como personajes que iban siempre en una única dirección y la jugabilidad no era muy fluida. Sin embargo, estoy impresionado de como el rendimiento del juego ha llegado tan lejos.



Soul Calibur se ejecuta a 60 FPS en el XU3 y a 60 FPS en el U3 - ¡Basta ya de seguir haciendo comparaciones!

En el último momento, decidí probar un juego más. Ya que la gente disfruta con los juegos de lucha, escogí Soul Calibur. Podría haber usado el Tekken 6 como hace HardKernel para sus demos, pero pienso que el Tekken es bastante aburrido con poca acción y gráficos mediocres. Prefiero Soul Calibur al Tekken, ya que incluso con la Dreamcast, Soul Calibur siempre tenía unos gráficos impresionantes, con reflejos, fluidos movimientos de los personajes, espadas y todo tipo de armas.

Aunque Soul Calibur se ejecuta a la misma velocidad en el XU3 y en el U3, me da la sensación que va algo más rápido en el XU3. Los menús responden mejor, aunque tras un breve "shock" inicial

el U3 es tan rápido como su hermano mayor. En definitiva, PPSSPP muestra muy bien cómo un emulador puede aprovechar el hardware. PPSSPP incluso tiene una opción para usar OpenGL ES 3.0, que en teoría permitiría más efectos en el XU3. Sin embargo, el proyecto PPSSPP ahora está en fase de reestructuración porque están pasando de SDL a SDL2, lamentablemente dará lugar a una versión más reciente de PPSSPP que será temporalmente incapaz de funcionar en la plataforma ODROID.

Al igual que con otros emuladores, PPSSPP tiene presente que el XU3 todavía le queda potencial de reserva, mientras que el U3 a menudo llega a su límite. Lo que significa que, en lugar de una resolución 2x, probablemente sea posible utilizar una resolución 3x en el XU3, mejorando aún más los gráficos y haciendo que los juegos se vean como si estuvieras jugando en una Xbox 360.

Reflexiones finales

Aunque el rendimiento del XU3 es increíble, tiene muchos fallos. XBMC no funciona correctamente, y las funciones del decodificador MFC en XBMC son simplemente una solución ingeniosa, obliga al sistema a usar MFC en lugar de comprobar si está realmente disponible.

OpenGL ES parece estar un tanto dañado en el XU3, aun cuando es compatible con la versión 3.0. Los Juegos OpenGL ES nativos así como glshim, parecen ejecutarse más lentos en el XU3 que en la U3, pese a que debería tener un mejor rendimiento en el XU3. Sólo Doom3 con algunos problemas técnicos es capaz de aprovechar el potencial superior del XU3 para mejorar el rendimiento de los gráficos.

Esto me lleva a la conclusión de que OpenGL ES 2.0 (y probablemente 3.0 también) están funcionando muy bien en el XU3, mientras que el rendimiento del OpenGL ES 1.1 es peor en el XU3 que en el U3. Los problemas con vsync, resolución de pantalla y de fisuras indican que hay alguna incompatibilidad

con el XU3, probablemente se trata de un problema con el driver xf86-video-armsoc. Parece incapaz de gestionar las diferentes modos, como lo hace el U3, lo que significa que probablemente haya una solución si alguien es capaz de arreglar el driver xf86-video-armsoc.

Por otro lado, cuando se inicia la aceleración 3D mediante SDL, el rendimiento es muy bueno y no hay problemas residuales. El XU3 muestra claramente que su CPU ofrece una gran ventaja cuando se trata de emular otros sistemas, el XU3 hace un muy buen trabajo con la mayoría de los emuladores.

Por consiguiente, te sugiero que siempre que quieras ejecutar OpenGLES nativo con glshim, utilices el U3 hasta que se resuelvan los problemas con el XU3. Sin embargo, para la emulación, el XU3 es impresionante y muy recomendable, ya que todos los emuladores pueden hacer uso de su potente CPU dando mejores resultados, del orden de un 15-50% sobre el U3.

Lamentablemente, el XU3 falla o se cuelga a menudo, lo que disminuye su experiencia de juego. Una solución es utilizar un módulo eMMC, que permite reiniciar el XU3 rápidamente. Si puedes asumir el tener que reiniciar el equipo de vez en cuando, es el dispositivo perfecto para los juegos y/o para reemplazar tu ordenador de escritorio. Curiosamente, los juegos disponibles para el XU3 son muy estables, ya que el XU3 nunca se cuelga durante el juego, pero si en el arranque o al salir. Así que al menos mientras juegas, no perderás tus progresos.

Todas las pruebas se realizaron con Debian Wheezy usando ODROID GameStation Turbo, así que no puedo decir si los juegos que he probado tendrían un comportamiento distinto usando Ubuntu 14.04. Estoy evaluando todavía Debian Jessie para ver si resuelve algunos de los problemas ya detectados, pudiendo haber margen de mejora que daría como resultado cambiar a un sistema operativo más actualizado.

IMAGENES DE LA COMUNIDAD

por Rob Roy

Hardkernel produce muchas imágenes precompiladas para ser utilizadas en el U3 y XU3 como Android y Ubuntu. Por otro lado algunos ODROIDians han creado distribuciones con fines específicos basadas en versiones oficiales y las comparte con la comunidad de código abierto. Aquí tienes una breve lista de contribuciones que se han publicado en los foros ODROID:

OpenELEC

U3/XU3: <http://bit.ly/1t6fWgr>

Gamestation Turbo

U3: <http://bit.ly/1nVvQqz>

XU3: <http://bit.ly/1ASFO5O>

Cyanogenmod 11

U3: <http://bit.ly/1ASG8BL>

XU3: <http://bit.ly/1qMA6Oq>

Max2Play

U3: <http://bit.ly/1HMovDY>

Trusty Dev Centre

U3: <http://bit.ly/1t6h1ov>

Ubuntu Server

U3: <http://bit.ly/1CMYC8K>

Debian

U3: <http://bit.ly/13zNTiG>

Robotics (ROS + OpenCV + PCL)

U3: <http://bit.ly/16TLG3V>

XU3: <http://bit.ly/1xlEPbZ>

Android Pocket Rocket

U3: <http://bit.ly/1H2Legq>

XU3: <http://bit.ly/1wrlB0L>

Arch Linux (ALARM)

U3: <http://bit.ly/1wOEzng>

Kali Linux

U3/XU3: <http://bit.ly/1sZsZ7x>



openelec
openembedded linux entertainment center



cyanogen(mod)



ODROID



DOCKER: DESARROLLAR, ENVIAR Y EJECUTAR APLICACIONES EN CUALQUIER LUGAR

PARTE I - EMPEZAMOS CON CONTENEDORES

por Fred Meyer



Docker es una plataforma para desarrolladores y administradores de sistemas que permite desarrollar, enviar y ejecutar aplicaciones. Docker te permite ensamblar rápidamente aplicaciones a partir de componentes y eliminar la fricción que puede aparecer al enviar código. Permite que tu código sea testeado y utilizado en producción lo más rápido posible. Consta de los siguientes componentes:

- Docker Engine, que es una ligera y potente tecnología de virtualización contenedor de código abierto combinada con un flujo de trabajo para desarrollar y empaquetar aplicaciones.
- Docker Hub (<https://hub.docker.com>), que es un servicio SaaS para compartir y gestionar aplicaciones.

Docker (<https://www.docker.com/whatisdocker>) te permite alojar diversas aplicaciones al mismo tiempo en un único ODROID, lo que hace muy fácil su mantenimiento. He estado ejecutando un miniDLNA Docker desde hace varias semanas y es absolutamente estable, me proporciona música en casa. Docker te permite ejecutar muchas aplicaciones famosas de Linux, tales como:

- owncloud
- lamp
- openstack (dockenstack)
- node.js

- roundcube
- servio DLNA/Server
- madsonic
- webproxy/webfilter
- DHCP/DNS-Server, dnsmasq
- cloudprint (con cups)
- y mucho más



Distribuciones Linux compatibles con Docker

Todo se ejecuta dentro del propio contenedor Docker. El "sistema" Linux de cada contenedor puede estar basado en CentOS, Ubuntu, Fedora o Arch Linux (por nombrar algunos), bien por elección personal o bien por requisito de la aplicación. Esta técnica hace un uso eficiente de los recursos de ODROID y al mismo tiempo mantiene tu sistema operativo base/host limpio. Docker te asegura que si algo sale mal con una aplicación, el resto de contenedores de aplicaciones no se verán afectados y no será necesario volver a instalar el sistema.



Instalar y configurar Docker es el primer paso hacia un sistema estable

Prerequisitos

En tu ODROID, tendrás que ejecutar un sistema operativo con soporte para Docker. Mientras que Ubuntu 14.04 viene con soporte Docker, el verdadero kernel del XU3 requiere de una recompilación, encontrarás las instrucciones sobre cómo hacerlo más adelante. También puedes utilizar Arch Linux que viene con el kernel listo y una versión más actualizada de Docker.

Con tu ODROID funcionando, instala los binarios Docker desde el repositorio principal:

```
Ubuntu
$ sudo apt-get install docker.io

ARCH Linux
$ pacman -S docker
```

Imagen base

Sugiero empezar con una imagen Ubuntu 14.04, ya que tiene la base necesaria para compilar Docker desde la fuente. En general, este primer paso

está explicado en <http://bit.ly/1tn21Z9>. Para el XU3 es necesario algunos ajustes.

Una forma rápida de obtener una imagen base es a través del repositorio público de imágenes Docker, llamado Docker Hub y que está disponible en <http://bit.ly/1y1SMvO>. También he añadido mis imágenes base compiladas manualmente a Docker Hub. Si deseas evitar el desarrollo de las mismas, puede acceder y descargarlas fácilmente.



Ubuntu ejecuta Docker muy bien, y es el SO preferido por muchos usuarios

Ten en cuenta que las versiones actuales de Docker y Docker Hub no tienen en cuenta la arquitectura para la cual ha sido desarrollada la imagen. Todas las imágenes estándar están diseñadas para la arquitectura x86, y la función de autodesarrollo que ofrece el registro Docker sólo está disponible para x86. Sin embargo, Docker está basado en Linux y puesto que Linux soporta muchas arquitecturas, otros desarrolladores han añadido imágenes de otras arquitecturas.

Se ha establecido un estándar común para los nombres que es conocido por todos, donde los colaboradores citan la arquitectura de la imagen dentro del nombre de la imagen. Para la arquitectura ODROID, busca imágenes que lleven “armhf” en el nombre.

Para mayor comodidad, mi imagen base precompilada Ubuntu Trusty 14.04 está disponible a través del repositorio público de Docker en <https://registry.hub.docker.com/u/hominidae/armhf-ubuntu>. Escriba lo siguiente para cogerla e incluirla en tus propios desarrollos y proyectos Docker:

```
$ sudo docker pull hominidae/armhf-ubuntu
```

A continuación, con el fin de realizar una prueba con tu contenedor recién creado y poder ver simplemente el archivo `lsb-release` dentro, escribe:

```
$ sudo docker run hominidae/armhf-ubuntu cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04"
```

Por el contrario, si te gusta desarrollar tu imagen de cero, los siguientes apartados describen cómo hacerlo, usando un ODROID para realizar la compilación.

Desarrollar partiendo de cero

Hay una forma de crear imágenes Docker basadas en ARM desde un host Docker x86, ejecutando una versión de `qemuarm-static`, como se describe en <http://bit.ly/1CNgX5O>, pero te sugiero hacer esto más bien en tu ODROID.



Los entusiastas de Linux buscan instalar Docker para dar un paseo por la playa

Instalar la utilidad debootstrap

La utilidad `debootstrap` es necesaria para crear un tarball base de Debian/Ubuntu. Siga los pasos recomendados para su instalación en tu distribución Linux. Por ejemplo, en Arch Linux instala primero `yaourt`.

Crear el árbol minbase y tarball

En la línea de comandos escribe

lo siguiente, elaborará un árbol `ubuntu:trusty` en el directorio “ubuntu” relativo a donde ejecutas el comando:

```
$ sudo debootstrap --variant=minbase \
--include=iproute,iputils-ping \
--ARCH armhf trusty ./ubuntu \
http://ports.ubuntu.com/ubuntu-ports/
```

Instalar el paquete

Puedes instalarlo en tu sistema en cualquier momento usando:

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

Modificar la lista de fuentes

Copiar `sources.list` de la imagen estándar `ubuntu 14.04` para XU3 en tu árbol `ubuntu:trusty` recién creado es un buen comienzo para crear una imagen Docker que sea capaz de mantenerse por sí misma ejecutando un simple “`apt-get update && apt-get upgrade`”. Escribe lo siguiente en la pantalla de Terminal:

```
$ sudo cp /etc/apt/sources.list
./ubuntu/etc/apt/
```

La siguiente línea creará y agregará el árbol/tarball como una imagen, denominada “ubuntu” y etiquetada como “latest” a tu repositorio local Docker:

```
$ sudo tar -C ubuntu -c . | sudo
docker import - ubuntu
```

A continuación, ejecuta una prueba de tu contenedor recién creado escribiendo los siguientes comandos en una ventana de terminal

```
$ sudo docker run ubuntu cat \
/etc/lsb-release \
DISTRIB_ID=Ubuntu \
DISTRIB_RELEASE=14.04 \
DISTRIB_CODENAME=trusty \
DISTRIB_DESCRIPTION=\
"Ubuntu 14.04"
```

Retocar la imagen

Normalmente utilizarás un Dockerfile (<http://bit.ly/1x8oBnN>) para empezar a personalizar y mejorar tu imagen base. Con Dockerfile, enseñarás a Docker a instalar aplicaciones adicionales. Durante este paso, sólo se ejecuta el comando docker dentro del contenedor, sin ningún servicio (como initd/systemd), lo que hará que los comandos install para ciertas aplicaciones/servicios fallen.



Ubuntu Tweak es una aplicación diseñada para que configurar Ubuntu sea algo sencillo para cualquiera.

Para evitar esto, es necesario realizar algunos ajustes, como se muestra en el ejemplo de <http://bit.ly/13KBsRm>:

```
$ sudo docker run ubuntu echo
'#!/bin/sh' > /usr/sbin/policy-rc.d \
    && echo 'exit 101' >> /usr/sbin/policy-rc.d \
    && chmod +x /usr/sbin/policy-rc.d \
    \
    && dpkg-divert --local --rename --add /sbin/initctl \
    && cp -a /usr/sbin/policy-rc.d /sbin/initctl \
    && sed -i 's/^exit.*/exit 0/' /sbin/initctl \
    \
    && echo 'force-unsafe-io' > /etc/dpkg/dpkg.cfg.d/docker-apt-speedup \
```

```
\
    && echo 'DPkg::Post-Invoke {
"rm -f /var/cache/apt/ARCHIVES/*.deb /var/cache/apt/ARCHIVES/
partial/*.deb /var/cache/apt/*.bin || true"; };' > /etc/apt/apt.
conf.d/docker-clean \
    && echo 'APT::Update::Post-Invoke { "rm -f /var/cache/apt/
ARCHIVES/*.deb /var/cache/apt/ARCHIVES/partial/*.deb /var/cache/
apt/*.bin || true"; };' >> /etc/apt/apt.conf.d/docker-clean \
    && echo 'Dir::Cache::pkgcache
""; Dir::Cache::srcpkgcache
"";' >> /etc/apt/apt.conf.d/docker-clean \
    \
    && echo 'Acquire::Languages
"none";' > /etc/apt/apt.conf.d/docker-no-languages \
    \
    && echo 'Acquire::GzipIndexes
"true"; Acquire::CompressionTypes
::Order:: "gz";' > /etc/apt/apt.conf.d/docker-gzip-indexes
```

Finalizar la imagen

Para guardar la imagen, es necesario cometer los cambios desde arriba y etiquetar la imagen antes de pasar al siguiente paso. En primer lugar, coge el último id del contenedor de la última ejecución. Luego, utiliza ese id (los 3 primeros dígitos es suficiente) para cometer los cambios y etiquetar la nueva imagen resultante:

```
$ sudo docker ps -l
$ sudo docker commit <id> ubuntu:14.04
```

Explorar Docker

Ahora tiene funcionando una imagen armhf con Ubuntu Trusty 14.04 para empezar. Ejecuta este comando para ver las imágenes disponibles:

```
$ sudo docker images
```

En algún momento, especialmente

cuando inicias un nuevo proyecto, es bueno tener tu contenedor Ubuntu actualizado añadiendo la siguiente línea en tu Dockerfile:

```
RUN apt-get update && apt-get
upgrade)
```

También querrás guardar la imagen, una panorámica de tu “sistema” para una posterior reutilización, escribiendo:

```
$ sudo docker save <image-id>
<name>.tar
```

Puedes explorar otros comandos docker-cli en <http://bit.ly/13KDxwN>.

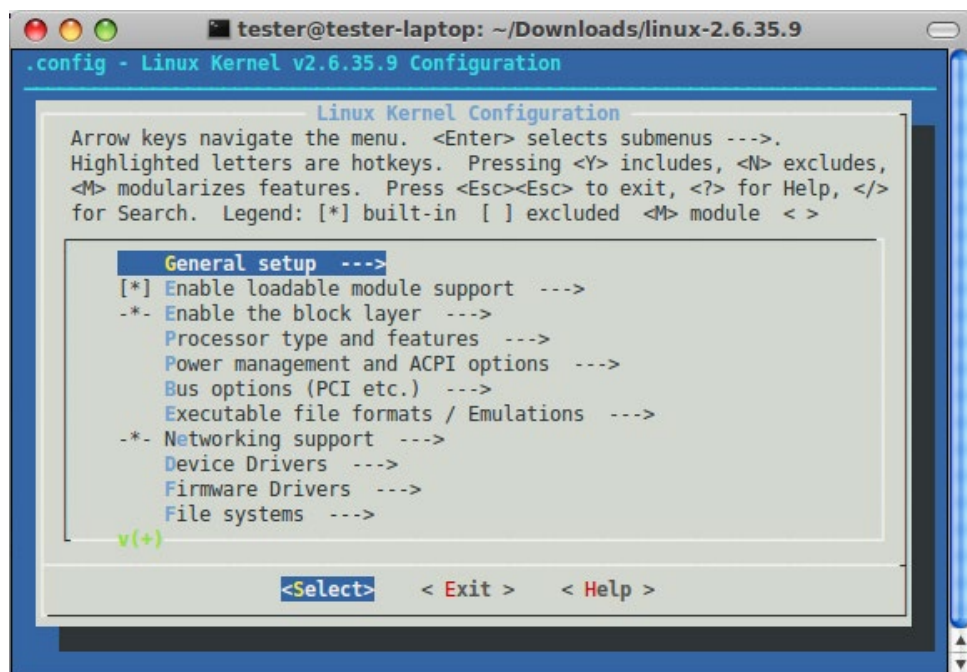
Actualizar los binarios de Docker

Docker está todavía en desarrollo, se añaden constantemente nuevas funcionalidades y la API está en continua evolución. Lo bueno de Docker es que si quieres, puedes actualizar tu host ODROID a la última versión de los binarios. Puedes recompilar Docker, ya que se compila dentro de un contenedor ubuntu 14.04 y crear/installar los binarios siguiendo las instrucciones de <http://bit.ly/16U9epg>.

Como se ha mencionado anteriormente, Docker no es consciente de la arquitectura en la que se instala, lo que significa que tampoco lo es el contenedor Docker para compilar los binarios docker. Para ejecutar la compilación en tu host ODROID, tendrá que hacer dos ajustes en la fuente obtenida de git.

El Dockerfile compilará GO (<https://golang.org>) para todas las arquitecturas conocidas/compatibles. Obviamente, esto supone que la compilación sea ejecutada en máquinas basadas en x86, y que sean capaces de hacer una compilación cruzada. Simplemente elimina todas las arquitecturas en el Dockerfile a excepción de “Linux/ARM”.

En segundo lugar, el demonio Docker que compilará los binarios tiene una marca de datos para la arquitectura



Un ejemplo de aplicación de menú, es uno de los pasos básicos en la compilación de un kernel. Hay una gran cantidad de opciones, ¡así que tómate tu tiempo! que necesites!

amd64 en su código. Tendrás que quitar la marca en el archivo `docker/daemon/daemon.go` del árbol de fuentes de git, para conseguir un demonio Docker que funcione.

Compilar kernel

Ubuntu es el sistema operativo host por defecto para muchos usuarios, utiliza el kernel suministrado por el equipo Hardkernel. Sin embargo, la compilación actual del kernel no incluye los parámetros adecuados para soportar Docker. Los siguientes pasos fijan esta configuración, empezando por comprobar si tu kernel ya permite Docker.

Instalar lxc

Aunque lxc no es necesario para Docker, ya que Docker viene con su propia implementación llamada libcontainer, este paquete es una pequeña herramienta es útil para comprobar/examinar la configuración del Kernel

```
$ sudo apt-get install lxc
```

Ahora comprueba si tu kernel en ejecución necesita una configuración específica y si es necesaria una recompilación:

```
$ lxc-checkconfig
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled
Multiple /dev/pts instances: enabled

--- Control groups ---
Cgroup: enabled
Cgroup clone_children flag: enabled
Cgroup device: enabled
Cgroup sched: enabled
Cgroup cpu account: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled

--- Misc ---
Veth pair device: enabled
Macvlan: enabled
Vlan: enabled
File capabilities: enabled
```

Si ves deshabilitados cualquiera de los parámetros del kernel anteriores, necesitas preparar una nueva configuración del kernel y volver a compilarlo. Ten en

cuenta que puedes apuntar la herramienta `lxc-checkConfig` hacia un config-file del kernel, lo que te permite probar un kernel sin tener que arrancarlo:

```
usage: $ CONFIG=/path/to/config /usr/bin/lxc-checkconfig
```

Preparar una configuración adecuada y compilar el kernel

En primer lugar, coge un árbol de compilación de kernel. Ten en cuenta que estas instrucciones son para el kernel 3.10.y. Las instrucciones para la compilación del kernel están detalladas en la Wiki de ODROID en <http://bit.ly/1ATKTLh>. Ve a la Sección de “Linux” y lee el apartado titulado “Kernel Re-build Guide”, con los siguientes pasos adicionales:

1. En el paso 2 de la Wiki, durante `menuconfig`, siga los siguientes pasos de configuración:

- a. bajo la entrada Filesystems, desactivar el soporte para XFS.

El Kernel 3.10.y tiene un error de configuración de dependencia en su árbol de compilación... No serás capaz de habilitar la siguiente parte, hasta que no desactives XFS.

- b. En General -> Namespaces, activa “User Namespaces”

- c. En General -> cgroup, activar todas las opciones

- d. En Devices -> Character Devices, activa “support for multiple dev/pts instances”

- e. Guarda la configuración y salte de `menuconfig`

- f. Vuelve a comprobar la nueva configuración antes de compilar, escribe:

```
$ CONFIG=../.config /usr/bin/lxc-checkconfig
```

Es de esperar que tu nueva configuración ahora tenga todas las funciones necesarias.

2. Continúa para compilar e instalar

el kernel, tal como se aparece en la Wiki.
kernel, as laid out in the Wiki.

el kernel, tal como se aparece en la Wiki.

3. Una vez que hayas arrancado tu kernel recién compilado, verifica su configuración de nuevo, escribe "lxc-check-Config".

¡Enhorabuena! Tras completar estos pasos, ahora dispones de un kernel apropiado para usar aplicaciones en contenedores de Docker en tu ODROID.

Notas

Al habilitar Docker en kernel 3.10.y, tu host perderá la capacidad de soportar el formato de ficheros XFS. Esta es la razón principal por la que el equipo de Hardkernel no ha proporcionado un kernel con Docker activado (todavía).

Si tienes acceso a una imagen Arch Linux para el XU3, hay un acceso directo para establecer una configuración del kernel para trabajar. Puesto que ARCH Linux para ODROID-XU3 viene con un kernel 3.10.y con Docker ya activado, puede extraer la configuración desde allí. Para usar una instalación de ARCH Linux en ejecución, escribe lo siguiente en un Terminal:

```
$ zcat /proc/config.gz > .config-arch && CONFIG=../.config-arch /usr/bin/lxc-checkconfig
```

Luego descarga el archivo .config-arch e inclúyelo en el paso 2 de la guía de compilación del kernel de la Wiki.

En la próxima entrega de esta serie, presentaré algunas de mis imágenes Docker precompiladas para que puedas tener un sistema Docker funcionando aún más rápido.

>information ~]\$ help -

1971

June 1971 **Richard Matthew Stallman** joined MIT Artificial Intelligence Laboratory as a programmer. Sharing codes was done freely that time.



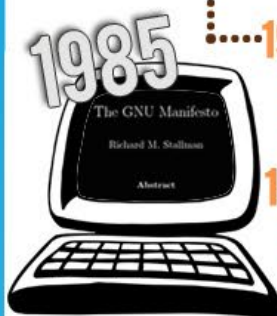
1980

A business model emerged from software compiled to run on **different PCs** but companies restricted code sharing, copyrighting their software instead.

Years after, Stallman founded the **Free Software Foundation** believing that software has to be **free** always.



1985



1985

STALLMAN PUBLISHED THE GNU Manifesto in a bid to create a **free OS** called **GNU** that would be compatible with Unix.

1985

ALSO SAW THE CREATION OF **MINIX**, an OS from scratch for the Intel i386 platform by Professor Andy Tanenbaum.



1989

Stallman released GNU **General Public Licence (GPL)**, the first program independent. All of Stallman's work was under this licence.



Finnish student **Linus Torvalds** of **University of Helsinki** came across Minix and wanted to upgrade it by putting in more features.

1989

As he was barred from further improving **Minix**, he wrote his own kernel now known as **Linux** and released it **under GPL**.



1991

Mid-September 1991 - **Linux version 0.01** was released and put on the internet



October 5, 1991 - **Linux version 0.02** came about

December 1991 - **Linux version 0.10** came but had support only for AT hard disks with no login.



Linux version 0.11 was considered better with support for multilingual keyboards floppy disk drivers, support for VGA, EGA, Hercules, etc.

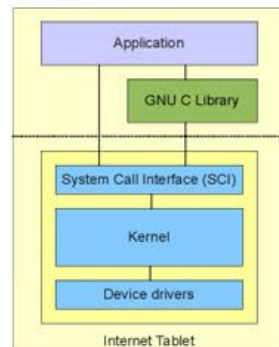
Linux codes then spread **worldwide** via ftp sites.

1993 - Patrick Volkerding writes and releases **Slackware**, the first standalone version of Linux.

Linux was licensed under **GNU General Public Licence** ensuring that the source codes are free for all to copy, study and to change.

1993

1996



CONFIGURAR ASP.NET Y MONO

COMO DESARROLLAR UN SERVIDOR SIN MICROSOFT

por Rui Carmo

Estas son mis notas para configurar ASP.NET vNext y un entorno de ejecución Mono super vanguardista en dispositivos ODROID con punto flotante de hardware.

Compilar Mono desde la Fuente

```
# I'm going to do everything as
root
su -
# grab minimal dependencies
apt-get install build-essential
mono-runtime autoconf libtool
automake
# import required certificates
mozroots --import --ask-remove
--machine

# Now check out the Mono tree
(this alone will take ages if you
have a slow SD card)
git clone git://github.com/mono/
mono.git
cd mono
./autogen.sh --prefix=/usr/local
# grab the bootstrap compiler
make get-monolite-latest
# now is a nice time to go off
and take a long stroll by the
beach
make
# use this instead if you have
distcc like me, it will speed up
building the native bits:
# DISTCC_NODES="node1 node2
node3 node4 localhost" make -j5
CC=distcc
# now install it locally
```

```
make install
# Should report 3.10.1 (or above)
and hardware floating point
mono --version
```

Resultado de ejemplo

```
Mono JIT compiler version 3.10.1
(master/8dal86e Sat Oct 25
19:32:35 WEST 2014)
Copyright (C) 2002-2014 Novell,
Inc, Xamarin Inc and Contribu-
tors. www.mono-project.com

      TLS:          __thread
      SIGSEGV:      normal
      Notifications: epoll
      Architecture:
armel,vfp+hard
      Disabled:     none
      Misc:         softdebug
      LLVM:         supported,
not enabled.
      GC:           sgen
```

Soporte LLVM

Es posible configurar Mono para que use LLVM en lugar de su JIT integrado, pero requiere seleccionar el Git correcto y pasar `--enable-llvm = yes` a `autogen.sh` y `--llvm` a `mono`, que no es muy útil, ya que los pros y contras del rendimiento de la RAM son discutibles. El tiempo de arranque, en particular, parece verse afectado considerablemente.



vNext

Este script utiliza `myget.org` para co-ger los desarrollo de paquete vNext de noche por Eilon Lipton, que funciona con Microsoft, por lo que tus resultados puede variar dependiendo de cómo de estables sean las noches.

```
# grab K tools
curl https://raw.githubusercontent.com/aspnet/Home/master/kvminstall.sh | sh && source
~/kre/kvm/kvm.sh
kvm upgrade

# add the package repo certifi-
cates
sudo certmgr -ssl -m https://
nuget.org
sudo certmgr -ssl -m https://www.
myget.org
mozroots --import --sync
# run the samples
git clone https://github.com/
aspnet/home
cd cd home/samples/HelloWeb
kpm restore -s https://www.myget.
org/F/aspnetvnext/
```

Por cualquier pregunta sobre la configuración de .NET en un dispositivo ARM, por favor consulta el artículo original en <http://bit.ly/1AZH3hW>. Este artículo se presenta bajo licencia Creative Commons (<http://bit.ly/1jsHqrq>).

DESARROLLO ANDROID: EL POTENCIAL DE ZYGOTE

por Nanik Tolaram



Desarrollar Aplicaciones se ha convertido en algo muy simple desde que apareció el desarrollo con Android. Hay un montón de recursos en Internet para aprender. Como desarrollador de Android, conocerás la gran variedad de APIs que hay disponibles para su aplicación, pero a veces es mejor detenerse y pensar un poco en cómo se ejecuta, en realidad la aplicación dentro de Android. ¿Qué parte de Android se encarga de la app y que control hace de ella? En este artículo trataremos de responder a estas preguntas.

Zygote

Sabemos que Android utiliza la máquina virtual de Java para ejecutar aplicaciones, y que esta máquina virtual se llama Dalvik, que pasó a llamarse art en Lollipop/Android 5.0. Dalvik es una implementación de una máquina virtual Java, pero no es el servicio que controla el lanzamiento de la aplicación. Hay otro pequeño componente que controla el proceso de extremo a extremo que se llama Zygote.

Echemos un vistazo a lo que la Wikipedia dice del Zygote: “... En los organismos multicelulares es la etapa de desarrollo temprana del embrión. En los organismos unicelulares el Zygote se divide para producir descendencia, generalmente a través de mitosis, el proceso de la división celular.” Ignorando la parte relativa a la biología, podemos ver que zygote es el duplicado de las células, lo que en el mundo Android puede entenderse como el duplicado de un proceso. En resumen, Zygote se encarga de la ejemplificación y duplicidad de los procesos en colaboración con la máquina virtual.

Cada vez que se ejecuta una aplicación Java dentro de Android, estás desencadenando el proceso de lanzamiento. In-

ternamente, el proceso de lanzamiento es un procedimiento multicapa sencillo, que implica una serie de componentes que hablan y se conectan entre sí. De un modo general, el proceso funciona como muestra la Figura 1.

Cuando inicias una aplicación, estás dando instrucciones a Android para crear/ dividir un proceso, Éste se encarga de enviar una solicitud de conexión a Zygote durante la ejecución del proceso init cuando inicias Android por primera vez. Consulta la edición de diciembre de ODDROID Magazine <http://bit.ly/1x2sg6z> para conocer más detalles del proceso init. Una de las principales tareas de init es poner en marcha Zygote, logrando que resida en memoria a la espera de una instrucción.

Zygote Init

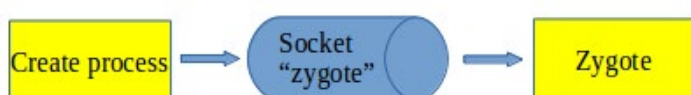
Este es el comando del proceso init que lanza Zygote:

```
service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server
    class main
    socket zygote stream 660 root system
    onrestart write /sys/android_power/request_state wake
    onrestart write /sys/power/state on
    onrestart restart media
    onrestart restart netd
```

El comando de servicio anterior ordena a Android que ejecute zygote con los permisos adecuados. Ten en cuenta que la aplicación realmente se llama app_main, pero el servicio se conoce como Zygote. A continuación se explican los diferentes parámetros que pasan por app_main:

--zygote: pide a la aplicación app_main que ejecute el programa en modo zygote, donde se activa el entorno y abre una conexión

Figura 1 : Proceso de creación de aplicaciones



--start-system-server: este pide a la aplicación app_main que inicie el servidor del sistema que requiere diferentes tipos de gestiones, más que las aplicaciones normales. El servidor del sistema contiene varios componentes que se ejecutan como parte del proceso init.

La aplicación app_main se utiliza para lanzar aplicaciones, y también se utiliza para poner en marcha los servicios internos de Android. Se puede decir que la aplicación app_main es la “app que se adapta a todo” para autofinanciar las aplicaciones dentro Android.

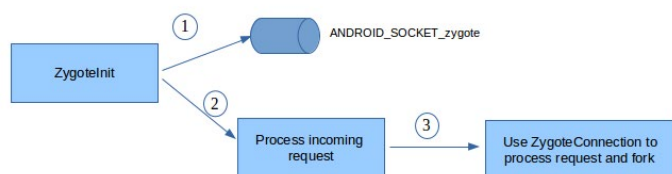


Figura 2 : Inicialización de ZygoteInit

La Figura 2 muestra la clase ZygoteInit activada durante el proceso init, con dos pasos principales:

1. Crear una conexión llamada “ANDROID_SOCKET_zygote”, que se utiliza para recibir peticiones entrantes.
2. Procesar Peticiones entrantes para lanzar nuevas aplicaciones y dividir procesos.

La clase zygoteinit es la clase principal que se encarga de todas las funciones relacionadas con zygote, incluida la preparación del entorno para la nueva aplicación que se va a utilizar, mientras que la clase ZygoteConnection se utiliza para manejar las peticiones entrantes.

Lanzamiento de Apps

Sabiendo que Zygote es el componente que se encarga de poner en marcha las aplicaciones de Android, podemos ir un

Figura 3 : Arquitectura de Android



Figura 4 : Flujo de lanzamiento de una App

poco más allá y observar las diferentes clases involucradas en el desarrollo de este suceso “mágico”.

Como vemos en la figura 3, vamos a analizar la parte resaltada de la arquitectura Android.

En concreto, vamos a revisar el Administrador de actividades, ya que este es el principal servicio que se encarga del ciclo de vida de la actividad de una aplicación en Android. El diagrama de flujo de la Figura 4 se muestra las clases que intervienen cuando se quiere ejecutar una aplicación:

- ActivityManagerService es el servicio principal de Android, que se encarga de las actividades dentro de las aplicaciones Android
- La clase Process es responsable de la distribución de aplicaciones y procesos creados dentro de Linux

Los pasos 2 y 3 de la figura 4 muestran la interacción con Zygote a través de la conexión ANDROID_SOCKET_zygote como se describe en la Figura 1. Como te puedes imaginar, sin Zygote no podrías ejecutar la aplicación, y todo el sistema

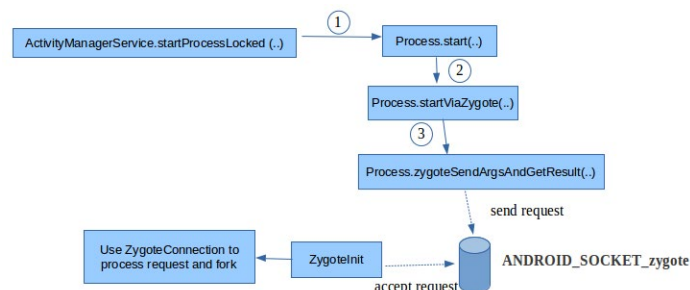


Figura 5: Flujo completo de Zygote

Android serían inservible. Zygote es sólo un pequeño componente en todo el entorno de trabajo de Android que ayuda a ActivityManager a iniciar aplicaciones en memoria.

En resumen, podríamos decir que toda la interacción entre las diferentes capas se parece a la figura 5. Con un mayor conocimiento de Zygote llegarías a comprender mejor cómo se ejecuta una aplicación dentro de Android.



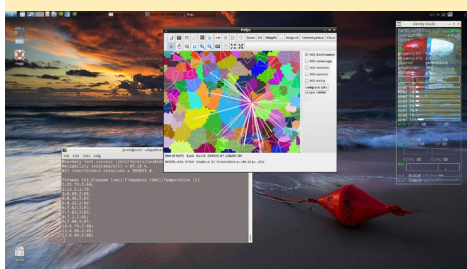
GRAFICOS DE LUJO CON JAVA: POIJU

por Jussi Opas

Para muchas aplicaciones, resulta muy valioso representar objetos usando gráficos; por ejemplo, el diseño de una planta, la ruta de un viaje en coche, el esquema de diseño de una placa, o la ruta de herramientas de un determinado programa. Para estos fines, se utilizan formas geográficas básicas, como líneas, rectángulos, elipsoides o formas libre. Otra forma para crear gráficos es mediante imágenes en mapa de bits, donde cada píxel contiene un color RGB y a veces también un valor de transparencia alfa.

Hemos creado una aplicación de ejemplo llamada Poiju, que se ejecuta en un ODROID-XU3, para que nos ayude con las representaciones gráficas de nuestros experimentos. En este artículo mostramos la capacidad gráfica de la aplicación, presentamos una visión general de sus funciones de imagen y de trazado y mostramos cómo su afinidad con el procesador octa-core del ODROID mejora el rendimiento de cálculo utilizando algoritmos de dominio.

Un escritorio con la aplicación Poiju y el monitor Conky en un ODROID-XU3



Introducción

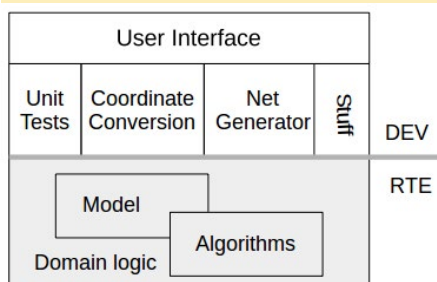
La palabra finlandesa Poiju significa boya. Es un marcador de mar que permanece en su ubicación sobre la superficie del agua, recoge el movimiento del viento y de las olas. La aplicación Poiju, tal como se utiliza en este artículo, configura los modelos informáticos y de señales para simular el comportamiento de la red. La aplicación ha sido desarrollada en parte para un uso real y en parte por diversión y para experimentar.

La lógica de dominios consiste en elementos espacialmente relevantes y algoritmos relacionados para usarse en un entorno de ejecución (RTE). Los algoritmos son:

- Modelo de propagación
- Cobertura combinada
- Dominación
- Área de servicio
- Interferencia
- Paralelización

La arquitectura informal de la aplicación Poiju se muestra en la Figura 2.

Figura 2 - Arquitectura de Poiju



El entorno de desarrollo se utiliza a nivel local para verificar la lógica de dominio así como para otros experimentos.

Swing y AWT

La visualización se puede utilizar para que un usuario de la aplicación o analista pueda verificar la integridad de los cálculos y los resultados intermedios. Java ofrece sus propios métodos para la visualización. Los paquetes Swing y AWT de Java son capaces de convertir dibujos lineales en contenido gráfico como una clase Graphics, y pintar la imagen en un contexto gráfico. Para Pintar en la interfaz de usuario se utiliza un sistema de coordenadas cuyo origen está en la esquina superior izquierda, con el eje Y tirás hacia abajo y con el eje X hacia la derecha.

La clase de Java AffineTransform se utiliza para traducir, rotar y escalar la vista de objetos a sus posiciones correctas. Mientras tanto, los objetos de dominio son delimitados utilizando otro sistema de coordenadas. Por ejemplo, la ubicación de los objetos geográficos se puede conseguir con la longitud y latitud o con coordenadas norte y sur en una zona UTM puntual. Las celdas espaciales de Poiju se crean primero como implementaciones de formas y se pintan después. Por ello, todas las formas están disponibles. Por ejemplo, es fácil determinar si un clic del ratón realiza algo en un mapa. Vamos ver cómo Java dibuja

sobre un ODROID-XU3 mientras configura redes multi-nodo.

Dibujar

La siguiente figura muestra el dibujo de celdas con cuatro métodos diferentes: con marco o con suavizado de bordes, y el contenido de los sectores con o sin relleno. Visualmente, uno seleccionaría el último método con el suavizado de bordes activado y el contenido vacío.



Figura 3 - 4 métodos para dibujar celdas

Intuitivamente, es de suponer que el método que no tiene relleno es el más rápido. También es de suponer que el relleno requiere más tiempo. Entonces, sería lógico pensar que el suavizado de bordes consume más tiempo y que el relleno con suavizado de bordes sería el método más lento. Para decidir cuál de los métodos usar se puede, por supuesto, probar con una aplicación real y recopilar los tiempos de trazado obteniéndose así información precisa. Los tiempos de trazado de 10.000 celdas se muestran en la siguiente tabla:

Line style and fill method	Time [sec]
Border aliased	0.41
Border aliased and fill	5.7
Border anti-aliasing	1.01
Border anti-aliasing and fill	1.7

La sorpresa es que dibujar líneas con contenido es básicamente más lento que cualquier otro método, se tarda 5,7 segundos en dibujar. Si el límite para usar un medidor de progreso está en 2 segundos, la aplicación debería usarlo mientras realiza el trazado. En cuanto al rendimiento, es factible dibujar con suavizado de bores con relleno, aunque también es interesante el suavizado de bordes sin relleno. También hemos registrado el tiempo para trazar 305.000 líneas. Los resultados se muestran en la siguiente tabla:

Line style	Time [sec]
Aliased	9.9
Anti-aliasing	19.4

Como se observa en la tabla, en un segundo se pueden dibujar 30.800 líneas alias o 15.700 líneas suavizadas. Ten en cuenta que el componente Swing no es seguro para los hilos de ejecución. Por ello, un único hilo puede dibujar líneas y no se pueden utilizar múltiples núcleos para acelerar el trazado.



Figura 4 - Líneas de interferencia

Como se muestra en la Figura 4, todas las líneas de interferencia se han pintado en una sola imagen. La interferencia se representa como probabilidad (CIP) y su magnitud se mueve en el intervalo [0, 100]%. Cuanto más fuerte es la interferencia más oscuro es el color rojo. Respectivamente, un color claro significa que la interferencia es baja entre dos celdas. Pintar todas las líneas de interferencia a la vez no tiene sentido, pero lo hemos hecho para mostrar la capacidad gráfica de la aplicación Poiju.

Todas las líneas de interferencia se

Figura 5 - Líneas FEP

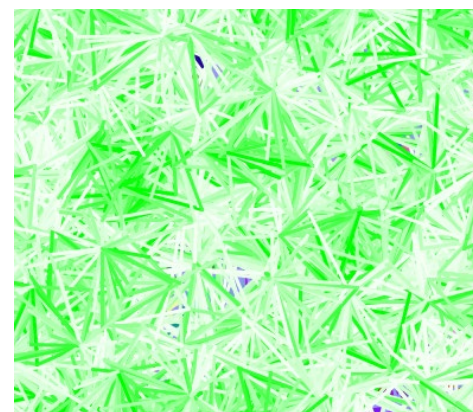
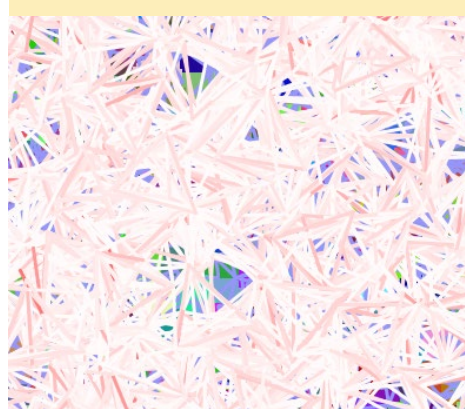


Figura 6 - Líneas ARP

calculan con otro método de interferencia en la Figura 5. Las líneas son más leves, porque el método matemático utilizado (FEP) es diferente al de la imagen anterior.

Las Líneas de interferencia de la Figura 6 se muestran como un promedio de potencia recibida (ARP), cuyo intervalo de valores es [0, 63] y la unidad es RXLEV. Se ha utilizado un blanco con un verde degradado.

Images

Con Java las otras formas de producir gráficos es crear imágenes. Una vez que una imagen ha sido representada, su trazado es muy rápido. No las podemos medir con el método System.currentTimeMillis, porque el trazado tarda menos de una milésima de segundo. Por eso, el

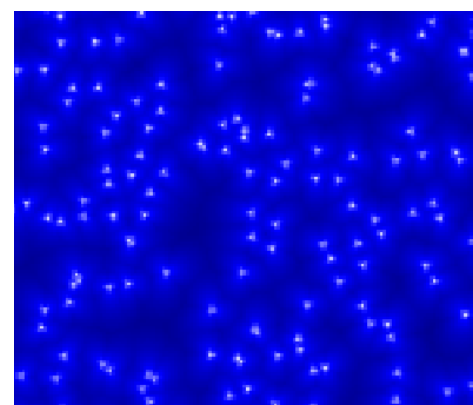


Figura 7 - Cobertura Combinada

tiempo sólo se tiene en cuenta cuando los algoritmos de dominio calculan el contenido de un mapa de bits. La figura 6 muestra qué tipo de gráficos podemos generar con la funcionalidad de imagen.

En la cobertura combinada parece

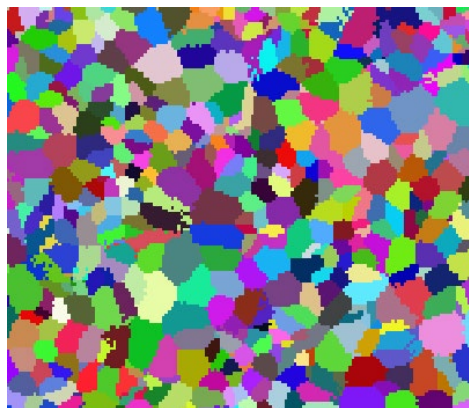


Figure 8 - Dominación

como si los focos se encontraran alrededor del espacio reservado. Cada píxel tiene el valor de intensidad del campo de la celda más fuerte. Alternativamente, se podría colorear con un color diferente.

La imagen Dominación contiene el identificador de la celda más sólida en cada píxel. Se colorear para asignar un color al azar a cada identificador.

La Figura 9 muestra cuántas celdas atienden a cada píxel. Se ha utilizado un

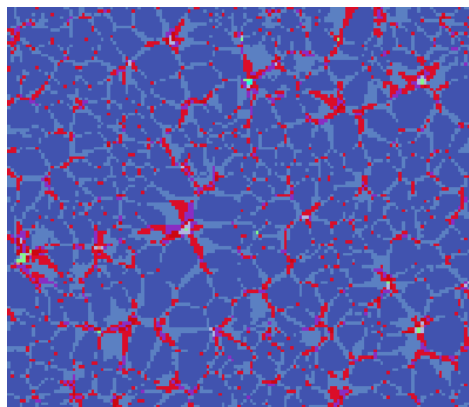


Figura 9 - Servidores

margen de traspaso de 2 dB para la imagen. Los píxeles azules son atendidos por una celda, la luz azul está atendida por dos células, los píxeles rojos por tres células y así sucesivamente. Una red de tecnología streaming con varios servidores implicaría un área de traspaso suave, donde los datos se transfieren a un móvil por varias células a la vez.

Capas

Es útil usar ambas imágenes y dibujar en la misma imagen. Se puede hacer trazando varias capas de datos en el mismo contexto gráfico. Por ejemplo, puede

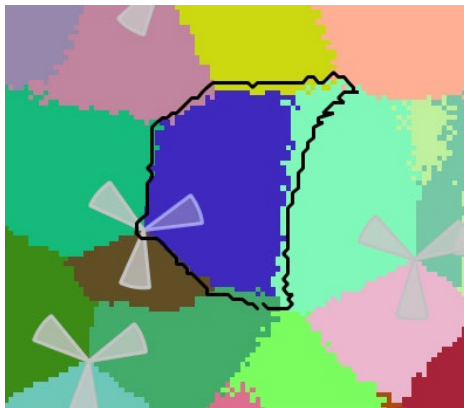


Figura 10 - Area de servicio y dominación

haber una imagen en segundo plano y luego se dibuja una o más capas encima.

Con los gráficos disponibles es po-

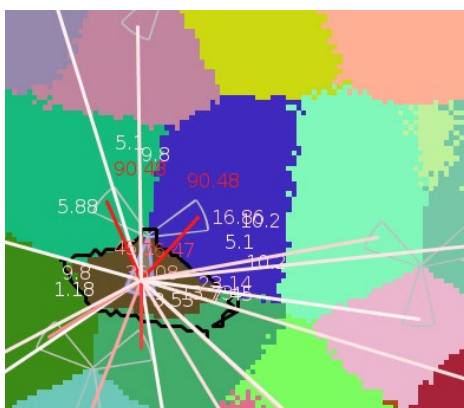
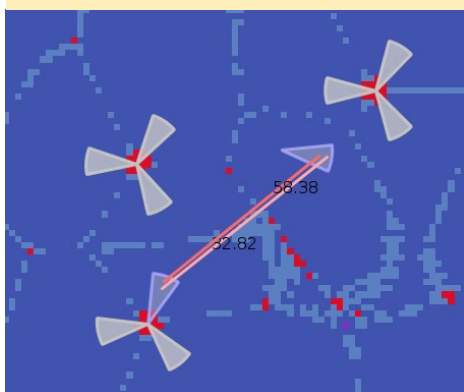


Figura 11 - Interferencia con valores

sible inspeccionar más de cerca una red. En la Figura 10, la zona de dominio de la celda seleccionada se muestra con el color azul. La línea del borde negro muestra que el área de servicio es mayor que el área dominio.

En la figura 11, las líneas de interferencia de la celda seleccionada se han pintado con valores de probabilidad de interferencia. Además, el color de los valores pintados está cambiando de acuer-

Figura 12 - Interferencia entre dos celdas



do a los valores de interferencia. También es posible inspeccionar valores de interferencia entre celdas seleccionadas. La figura 12 muestra una situación en la que dos celdas han sido seleccionadas.

En un segundo plano, el número de margen de traspaso es de 0 decibelios. Sólo hay unos pocos píxeles con varias señales que atender. La interferencia mutua entre las celdas está desequilibrada, ya que la interferencia en una dirección es más grande que en la otra (33% vs 58%). El color de la línea de interferencia cambia en consecuencia.

Muchas dimensiones

En una fase temprana de desarrollo, la aplicación solo permitía el dibujo lineal y tablas que mostraban los valores de interferencia. Sin embargo, usando sólo las tablas no es posible decir si los valores calculados son correctos. Por el contrario, con el trazado lineal sobre un mapa se hace evidente que los valores están a veces sesgados o son erróneos. No todas las relaciones de interferencia fueron contabilizadas, pero la función de especificación insistía en que las fórmulas eran correctas, se deduce pues que la función de ejecución cometió errores.

Una solución de software para este tipo de problemas es el desarrollo de gráficos que muestren la corrección o incorrección de las matemáticas aplicadas y sus implementaciones. Los gráficos deben ser creados dentro del sistema y usar el mismo lenguaje que se utiliza en la aplicación. Además, la verificación gráfica debe ser desarrollada bajo el entorno que implementa la fórmula. Esto se debe a que Matlab tiene algunas funciones que no existen en el lenguaje de producción. Otras cuestión es si es posible calcular razonablemente la relación de interferencia sin hacer recorridos de píxeles. La función de especificación quería evitar una implementación basada en píxeles, porque las implementaciones anteriores habían sido demasiado lentas y los tiempos de desarrollo habían sido demasiado largos.

En ese momento del desarrollo, los ingenieros de pruebas estaban inspeccionando las líneas de interferencia con un diseño de red estático. No fue posible mover las celdas o girar sus antenas. Sin embargo, en una prueba completa hay que tener en cuentas todas las configura-

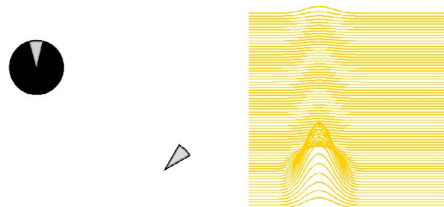


Figura 13 - Precoz gráfico de interferencia

ciones posibles. Para implementar esto, hicimos el siguiente experimento: se colocaron dos celdas cerca de las demás y la orientación de la antena principal se cambiaba de forma gradual un grado. Conseguimos 360 líneas y cada una tenía 360 valores. Los valores y el diseño de celda se muestra en la Figura 13.

El círculo negro en la esquina superior izquierda refleja que una celda entrometida ha sido trazada 360 veces, una vez por cada de grado de muestra. Basán-



Figura 14: Trazo de l punto opuesto al círculo

donos en la figura del lado derecho, dos celdas parecen interferir entre sí sólo cuando son dirigidas hacia el resto. Esto es erróneo, porque la interferencia es causada por la celda portadora cuando una celda entrometida está disparando desde atrás. Según esto, el cálculo debe ser rehecho y corregido.

Tras la corrección, la relación de interferencia mutua entre dos celdas se muestra en la Figura 12. La relación de interferencia entre dos celdas es continua, y una celda que dispara hacia tras y a los lados causa interferencia a la celda portadora. Otros proyectos antiguos o nuevos, también podrían utilizar este método de triple eje, <portador, interferencia, valor de interferencia>, para verificar su definición y aplicación.

Paralelización

El cálculo de interferencia puede ser paralelizado. XU3 tiene 4 núcleos grandes y 4 pequeños, lo que permite que un programa ejecute Java con 8 hilos simultáneamente. La dominación y la cobertura combinada de 10.000 cel-

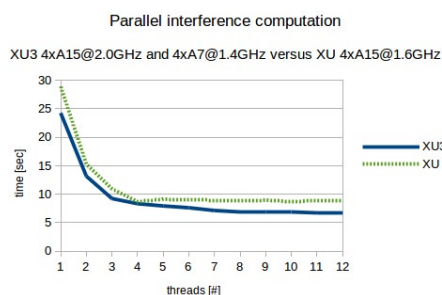


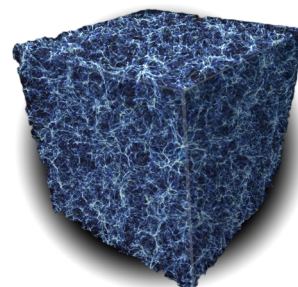
Figura 15 - Calculo de interferencia en paralelo en XU y XU3

das se calcula en 0,4 segundos. En la figura 15 aparece un perfil de prueba del cálculo de interferencia en paralelo con diversas configuraciones de hilos. La interferencia de 10.000 celdas se calcula en menos de 7 segundos. Trazar las líneas lleva más tiempo, 10 o 19 segundos dependiendo del estilo de bordes.

También podemos comparar la capacidad de multiprocesamiento heterogénea del XU3 con el XU. El XU3 puede funcionar con 8 hilos de ejecución al mismo tiempo mientras que el XU ejecuta 4 hilos. Puesto que el XU3 funciona a 2 GHz, es más rápido con un hilo porque el XU utiliza núcleos A15 que funcionan a 1,6 GHz. Con estas cifras, se puede decir que no hay razón para evitar las implementaciones basadas en píxeles.

Conclusiones

Hemos demostrado que con Java, se puede crear fantásticos gráficos. La visualización gráfica permite mostrar de una forma consistente lo que hace una aplicación. En la lógica de dominio, la visualización se utiliza para validar la funcionalidad de aplicaciones durante el desarrollo. La tendencia actual en el desarrollo de software son las pruebas por sección y la automatización de pruebas. Esto no justifica el trabajo que se muestra aquí, considerado como algo que se debe evitar. A pesar de ello, hemos escrito una aplicación independiente con varios propósitos: para verificar modelos y algoritmos, aprender y por diversión. Mientras tanto, también podemos asegurar que los modelos de dominio y algoritmos funcionan perfectamente usando gráficos. Hemos encontrado que la ODROIDXU3 octa-core es una excelente plataforma para la implementación y puesta en práctica de experimentos de paralelización.



SEAFILE

SOFTWARE EN LA NUBE PERSONAL

por @tlankford

Seafile es un servicio en la nube que te permite crear y compartir ficheros privados en la nube con los amigos o compañeros de trabajo. Los archivos se sincronizan para todos los usuarios, por lo que si una persona edita un documento o hace algún cambio, Seafile actualiza automáticamente los cambios para todo el grupo. Es seguro, porque utilizas tus propios servidores, y fiable porque Seafile lo guarda todo, incluso se puede restaurar los elementos que han sido borrados accidentalmente. También es seguro porque los archivos pueden ser cifrados con una contraseña.

Este tutorial ofrece información general para instalar y configurar un servidor Seafile en un ODROID-U3. Seafile es una gran plataforma para alojar un blog, un servidor de una pequeña empresa o el servidor multimedia de la familia cuando se combina con un disco duro USB de 250GB externo.

Componentes necesarios

La lista de elementos para este proyecto es bastante corta. Voy a describir qué es lo que estoy usando y qué componentes son intercambiables u opcionales. Podemos configurar el servidor con o sin monitor. Yo prefiero ver el primer arranque en el monitor.

- ODROID-U3 con fuente de alimentación y cable HDMI
- Cable Ethernet o un adaptador Wifi
- Módulo eMMC o tarjeta SD de 16 GB (o más grande) para el arranque
- Disco duro compatible con Linux de 250GB

- Monitor, teclado y ratón, que puede ser suprimidos tras la instalación inicial

Intalar la imagen

Suelo usar SD formatter para formatear las tarjetas y el Win32 disk imager de Hardkernel para escribir una imagen de Ubuntu. No voy entrar en detalles puesto que ya hay tutoriales sobre ello.

Primer arranque

Aunque el ODROID puede iniciarse sin monitor y conectarte a él por SSH en el primer arranque, me gusta tener el escritorio para hacer los primeros cambios. Inserta el módulo eMMC o tarjeta SD, conecta monitor, teclado y ratón, cable ethernet y el adaptador de corriente. Si tiene que introducir una contraseña y nombre de usuario, son los siguientes:

User = ODROID

Password = ODROID

ODROID arranca rápido y el escritorio aparece inmediatamente tras iniciar sesión. Lo primero que hay que hacer es ejecutar ODROID utility que es la versión de Hardkernel del comando Raspi-config. Normalmente elijo en primer lugar cambiar el nombre de host, a continuación, expando la partición raíz e instalo los archivos Xorg actualizados. Si usamos con frecuencia el escritorio, también deberíamos actualizar el gestor de ventanas y drivers de vídeo, pero como no es nuestro caso podemos saltar este paso. Cuando hayas terminado,

puedes reiniciar para que los cambios tengan efecto. Una vez reiniciado, inicia una ventana de Terminal para que podamos hacer algunos cambios. En primer lugar, ejecuta el siguiente comando para verificar si hay una conexión Ethernet:

```
$ sudo ifconfig
```

Estamos buscando eth0 y su dirección IP. Toma nota de la dirección IP. Luego, introduce este comando para personalizar la configuración del teclado:

```
$ sudo dpkg-reconfigure keyboard-configuration
```

Selecciona la configuración UTF-8 apropiada para tu ubicación usando la barra espaciadora para anular la selección por defecto y seleccionar la deseada. Luego, introduce el comando:

```
$ sudo dpkg-reconfigure tzdata
```

Este comando te permite elegir la zona horaria correcta, es sencillo. Después, introduce el comando:

```
$ sudo nano /etc/ssh/sshd.conf
```

Esto abrirá el editor nano para que puedas realizar cambios en el script. Me he dado cuenta que las imágenes por defecto de Hardkernel no tienen instalado nano, por lo que puedo sustituir vi por nano si te resulta más cómodo. Si prefieres el editor nano, ejecuta el comando:

```
$ sudo apt-get install nano
```

Tras iniciar el editor, desplázate hacia abajo hasta que veas las siguientes líneas y cámbialas para que coincida con:

```
Port 22
Protocol 2
PermitRootLogin yes
```

Por seguridad, tendrás que cambiar la opción PermitRootLogin a “no” tras la configuración inicial. Guarda el archivo. Después, vincula la configuración mediante los siguientes comandos tomando nota de la dirección IP:

```
$ ifdown eth0
$ ifup eth0
```

Por último, escribe lo siguiente para aplicar los cambios:

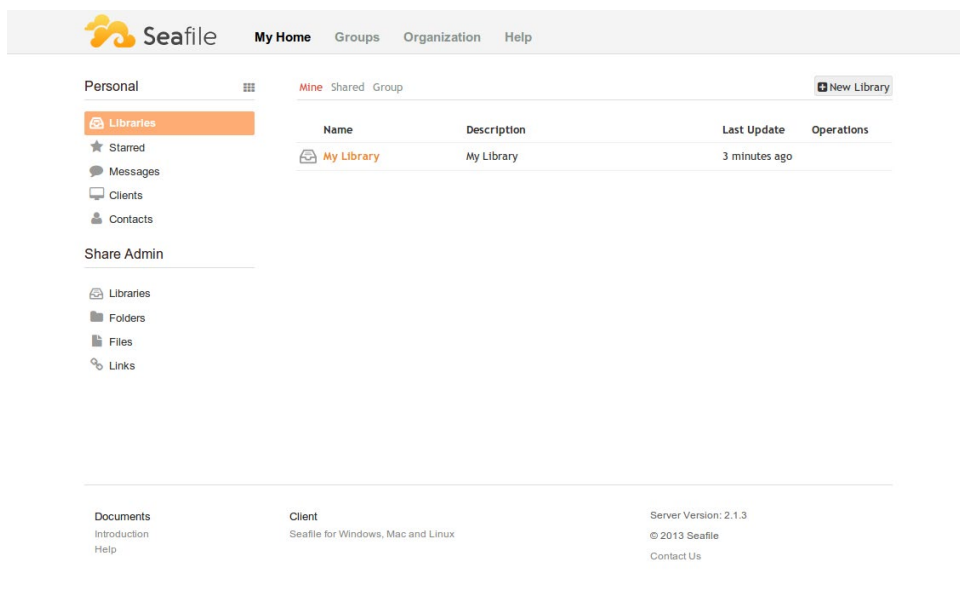
```
$ sudo reboot
```

Crear una muestra de prueba

Al conectar la unidad USB, en el escritorio aparecerá un mensaje para abrir la unidad con el administrador de archivos. Guardaremos la ruta de la unidad para después. En este ejemplo, el directorio es /media/ODROID/OneTouch 4 Mini. Puedes cambiar la ubicación en la cual se monta la unidad, pero a nosotros nos vale con la ruta por defecto. A continuación, abre una ventana de terminal y escriba el siguiente comando, aparecerá una lista de las unidades disponibles:

```
$ cat /proc/mounts
```

Ignora las unidades listadas que contiene las letras “mmc” y busca una unidad denominada /dev/sda *. El * probablemente sea la partición número 1. Si hay montadas varias particiones, entonces las veremos enumerar en orden. Asegúrate de que sólo tienes un duro externo conectado antes de ejecutar este comando. A continuación, queremos



ver si la unidad se puede utilizar como tal, o si vamos a tener que reescribirla. Para probar su funcionalidad, escribe el siguiente comando utilizando la ruta de la unidad proporcionada por el administrador de archivos en el paso anterior:

```
$ sudo nano mkdir /media/ODROID/
OneTouch 4\Mini/test
```

Si aparece un error de sólo lectura, tendrás que volver a formatear, lo cual se hace con unos sencillos pasos. Hay varias formas de hacer esto, nosotros vamos a utilizar un programa llamado parted. Ten mucho cuidado y presta atención a lo que haces, ya que la aplicación parted puede sobrescribir la unidad de arranque y dañar el sistema operativo, por lo que debes tener mucho cuidado cuando trabajes con unidades. En primer lugar, escribe el siguiente comando utilizando el nombre de la partición que puedes localizar con el comando “cat”:

```
$ umount /dev/sda1
```

Por último, escribe lo siguiente para aplicar los cambios:

```
$ sudo reboot
```

Esto desmonta la partición para no obtener un error de “drive in use”. A continuación, inicia la aplicación parted

y selecciona la partición apropiada:

```
$ parted
$ select /dev/sda
```

El comando “select” es muy importante para asegurarnos de que estamos utilizando la unidad correcta. No queremos tontear con la unidad de arranque en absoluto. A continuación, escribe el siguiente comando y te pedirá que asignes un nombre a la unidad. Para este ejemplo, la unidad se llamará “labserv”:

```
$ mklabel msdos
labserv
```

A continuación, escribe:

```
$ mkpart
```

Cuando se te solicite, responde con las siguientes respuestas:

```
primary
ext4
0%
100%
```

Por último, escribe lo siguiente para aplicar los cambios:

```
$ sudo reboot
```

Esto determina la distribución de la

partición, podrás cambiar estos valores si quieres hacer varias particiones, pero por simplicidad, sólo vamos a escribir una única partición. A continuación, escribe lo siguiente para comprobar que la partición se ha escrito correctamente, luego salte del editor de particiones:and you can change these values if you partición. Puedes cambiar estos valores si quieres hacer varias particiones, pero por simplicidad sólo vamos a escribir una única partición. A continuación, escribe lo siguiente para comprobar que la partición se ha escrito correctamente, luego salte del editor de particiones:

```
print
quit
```

Por último, designa un sistema de archivos para la partición escribiendo:

```
$ mke2fs -v -L labserv -t ext4 /
dev/sda1
$ umount /dev/sda
```

Una vez desmontada la unidad, desenchufala del ODROID, espera un momento y luego conéctala de nuevo. Comprueba el administrador de archivos para ver si la ruta de la unidad está montada correctamente. Como en mi ejemplo use la etiqueta “labserv”, el gestor de archivos muestra que está montada en el directorio /media/ODROID/labserv. En una ventana de terminal, escribe el siguiente comando, sustituyendo “labserv” por el nombre que elegiste para tu unidad:

```
$ mkdir /media/ODROID/labserv/
test
```

Si no hay errores, elimina la carpeta de prueba escribiendo:

```
rm -rf /media/ODROID/labserv/test
```

Si la carpeta de prueba ha sido eliminada, significa que la unidad está formateada correctamente y está lista para

su uso con el servidor. Para que puedas desenchufar el monitor y el teclado cierra ODROID escribiendo:

```
$ sudo shutdown -h now
```

Instalar MySQL

Para el resto de pasos, es necesario utilizar SSH. No voy a entrar en demasiados detalles en lo que respecta a SSH, puesto que hay muchos tutoriales disponibles con datos concretos. Use PuTTY para establecer la conexión que instale desde <http://bit.ly/1jsQjnt>. Tras lanzar PuTTY, te pedirá la dirección IP de ODROID. Asegúrate que el puerto es el 22. Pulsa Intro y esperar a que aparezca la ventana de Terminal. Después, inicia sesión como ODROID con la contraseña ODROID. Deberías ver el mismo símbolo de Terminal que verías con la aplicación nativa Terminal en el escritorio, algo así:

```
ODROID@ODROID~$
```

Si ves esto, significa que estás en el directorio principal. Nosotros hemos hecho algunos cambios. Ejecutaremos los siguientes comandos respondiendo “y” cuando lo solicite:

```
sudo apt-get update
sudo apt-get upgrade
```

A continuación, inicia la instalación de MySQL ejecutando el comando:

```
$ sudo apt-get install mysql-
server mysql-client php5-mysql
```

Se te pedirá que fijas una contraseña para el usuario “root”. Tras escribir la contraseña elegida, ejecuta dos comandos más escribiendo “y” para cada una de las respuestas, excepto con el cambio de contraseña de root. Asegúrate de recordar la contraseña de root, la usarás en la configuración del servidor Seafile.

```
$ sudo mysql_installation_db
```

```
$ sudo mysql_server_installation
```

Instalar Seafile

El software necesita dos dependencias durante la instalación, de modo que vamos a adelantarnos a este posible error con la preinstalación de las mismas, respondiendo “y” cuando lo solicite:

```
$ sudo apt-get install python-
setuptools python-simplejson
```

A continuación, instala la aplicación de servidor Seafile, que ofrece una interfaz amigable utilizando un navegador estándar. Crea un directorio temporal denominado seafile para almacenar los archivos:

```
$ mkdir seafile
$ cd seafile
```

Luego, descarga y descomprime el paquete en el directorio recién creado:

```
$ wget https://bitbucket.org/hai-
wen/seafile/downloads/\
seafile-server_3.0.4_pi.tar.gz
$ tar xzvf seafile_server*
$ cd seafile_server*
$ ./setup-seafile-mysql.sh
```

Sigue las indicaciones y elije los valores por defecto, excepto en dos casos: cuando se te pregunte por el nombre del servidor e introducir la dirección de IP, selecciona los puertos por defecto. Además, selecciona crear nuevas bases de datos MySQL. Tras responder a todas las preguntas, debería ver una lista que tendrá este aspecto:

```
Server Name: Your Server Name
IPAddress: <ODROID IP address>
ccnet server port: 10001
seafile data: /media/ODROID/lab-
serv/seafile-data
seafile server: 12001
HTTP server: 8082
MySQL server: 3306
ccnet database: ccnet-db
seafile database: ccnet-db
seahub database: seahub-db
```

Durante la instalación, se te pedirá que introduzcas la contraseña root de MySQL. Luego, iniciar el servidor Seafile y la aplicación de seahub escribiendo:

```
$ ./seafile.sh
$ ./seahub.sh
```

Cuando se te solicite introducir el correo electrónico de administración, escribe tu dirección de correo preferida, después fija una contraseña para el inicio de sesión. Una vez configurada la aplicación Seahub, puedes probarla utilizando un explorador local y navegando a <http://<dirección IP>:8000>, donde <dirección IP> debe coincidir con la del comando “ifconfig” anterior. Por último iniciar sesión y ¡Echa un vistazo a tu nuevo servidor de la nube!

Para más información sobre la configuración de Seafile, puedes consultar el artículo original en <http://bit.ly/1rIb9Te>, o la página web de Seafile en <http://seafile.com>.

El hombre en la nube nos dijo que disfruta pasando el tiempo navegando por tu colección de memes de gatos divertidos y selfies del gimnasio



HISTORIA DE LOS ODROIDS

por Rob Roy

Wikipedia mantiene un historial de los productos de Hardkernel, que incluye un gráfico de las especificaciones de todos los dispositivos ODROID desde 2009. Échale un vistazo en <http://en.wikipedia.org/wiki/Odroid>:

Name	CPU
Odroid 2009	Samsung S5PC100 833 MHz, Cortex-A8
ODROID-U2	1.7 GHz Exynos 4412
ODROID-X2	
ODROID-U3	1.7 GHz Exynos 4412 Prime
ODROID-XU	Exynos 5410 Octa big.LITTLE ARM Cortex-A15 @ 1.6 GHz quad-core and ARM Cortex-A7 @ 1.2 GHz quad-core CPUs
ODROID-XU3/XU3-Lite	Exynos 5422 Octa big.LITTLE ARM Cortex™-A15 @ 2.0Ghz (Lite @ 1.8Ghz) quad core and Cortex™-A7 quad core CPUs
ODROID-W ^[3] (discontinued)	Broadcom BCM2835 ARM11 @ 700 MHz
ODROID-C1 ^[4]	Amlogic S805, 4x Cortex-A5 @ 1.5 GHz

CONOCIENDO A UN ODROIDIAN

NANIK TOLARAM: UN JEDI JAVA

editado por Rob Roy

Por favor, háblanos un poco sobre ti.

Vivo en Sydney, Australia. Estoy casado con mi mejor amiga con la que tengo 2 hijos, adorables y muy traviesos de 5 y 10 años de edad. En el día a día, trabajo en el desarrollo de una ingeniosa plataforma de Android personalizada llamado ScreenerOS (<http://bit.ly/1wjixnr>) que es compatible con las plataformas x86 y ARM. (<http://bit.ly/1wjixnr>)

¿Cómo fueron tus inicios con los ordenadores?

Empecé a trabajar con los ordenadores cuando tenía 9 años. Mi primer ordenador fue un Apple (no recuerdo el modelo) y mi primer contacto con la programación fue con el lenguaje BASIC. Empecé a tener un gran interés por los ordenadores cuando, a los 11 años, se me infectó el sector de arranque con un virus llamado Denzuko. Ese fue el momento en el que empecé a aprender todo lo que necesitaba saber para limpiar virus. Desde entonces me he dado cuenta que tengo una gran pasión por aprender cómo funcionan los ordenadores por dentro. Escribí mi primer libro sobre ordenadores cuando tenía 17 años, era un libro sobre programación C++.

¿Qué te llevó a la plataforma ODROID?

Me encontré con ODROIDs cuando estaba investigando sobre los sistemas de hardware de código abierto integrados, descubrí que podía ejecutar Android y el precio era razonable. Al mismo tiempo vi



Nanik surrounded by his family. Kudos for happiness above all else!

por casualidad un post en el foro donde aparecía que estabais buscando un columnista Android, así que aproveche la oportunidad ya que también estaba buscando una forma de compartir mis conocimientos Android con la comunidad. Supongo que es mi manera de devolver a la comunidad todo lo que he aprendido del mundo open source.

¿Qué ODROID es tu favorito?

El ODROID-U3 es mi favorito, porque es pequeño y potente.

¿Cómo llegaste a ser tan hábil con el Java?

Empecé a aprender Java en 2004. Me encontré con Java por casualidad cuando estaba investigando el funcionamiento interno de las máquinas virtuales y el potencial que podían aportar. Descubrí

mucho sobre Java y J2EE con la lectura del código fuente de aplicaciones de diversos documentos de Java Specification Request (JSR). He estado trabajando con Java desde entonces, tanto a nivel de hobby como profesionalmente.

¿Qué aficiones e intereses tienes aparte de los ordenadores?

Desde joven, siempre he tenido un gran interés por los peces tropicales y su cría, sobre todo por los peces de colores y el pez disco. Un nuevo hobby al que últimamente estoy dedicando tiempo es el hecho de hacer proyectos electrónicos con mis hijos, así como de carpintería, que me encanta diseñar cosas desde cero.

¿Está involucrado con otros proyectos informáticos ajenos a ODROID?

Estoy participando de forma activa en la comunidad Sydney Android haciendo presentaciones (<http://bit.ly/1EbknRo>) y además, formo parte de las comunidades MinnowBoard Max.

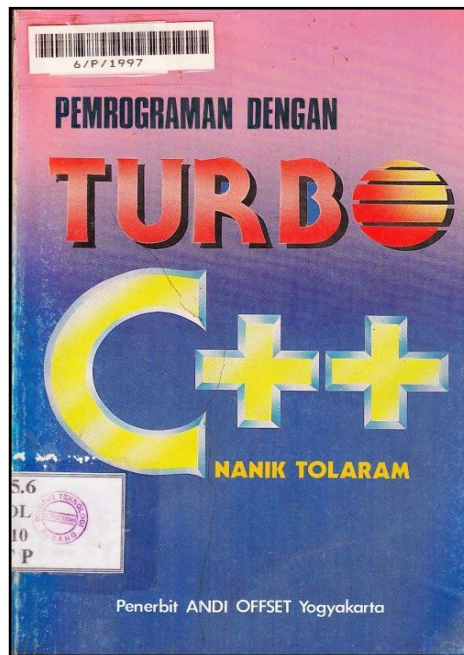
¿Qué tipo de innovaciones de hardware te gustaría ver en futuras placas Hardkernel?

Una cosa de la que me dado cuenta es que carecen de los sensores para placas específicas con Android. Sabemos que la mayoría de los dispositivos Android como teléfonos y tabletas tienen variedades de sensores que no existen para muchas de las placas desarrolladas.

¿Qué consejo le darías a alguien que quiere aprender programación?

Aprender a programar nunca ha sido tan fácil como hoy día, ya que hay muchos y muy buenos sitios web y comunidades que pueden ayudarte con cualquier tipo de problema. El código abierto ha cambiado la forma en la que la gente está aprendiendo. Es una buena oportunidad para que alguien aprenda lo que realmente le interesa.

Siempre digo a la gente que el peor enemigo al que tienes que vencer es a ti mismo, porque aprender es un proceso. El aprendizaje es fácil, pero el proceso que hay detrás que te empuja



Además de tener una familia feliz, Nanik es también un hábil escritor.

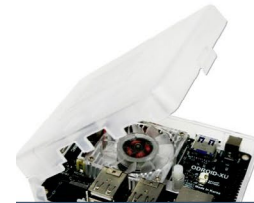
en sí mismo a dar lo mejor de ti es la parte difícil. Crear la motivación y el deseo de aprender a programar, puesto que la programación es una combinación de ciencia y arte. Analizar montones de código fuente, porque la mejor documentación está dentro de la fuente. Cometer errores es la mejor manera de aprender, si no cometes errores es que no lo has conseguido. Comprender cómo funcionan ciertas cosas es más importante que conocer un determinado lenguaje de programación. Piensa en la programación como la herramienta que usas para conducir, porque si no sabes conducir no serás capaz de llegar a tu destino.

“El peor enemigo al que tienes que vencer es a ti mismo, porque aprender algo es un proceso. El aprendizaje es fácil, pero el proceso que hay detrás que te empuja en sí mismo a dar lo mejor de ti es la parte difícil”.



Por si acaso pones en duda que Nanik es un Jedi, acabas de leer su cita.

OFFICIAL US DISTRIBUTOR OF HARDKERNEL PRODUCTS



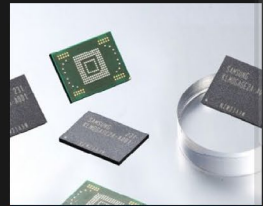
SINGLE-BOARD COMPUTERS



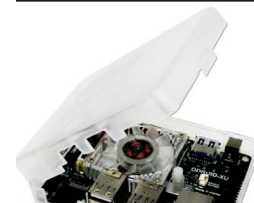
DISPLAYS



DEVELOPMENT



FLASH STORAGE



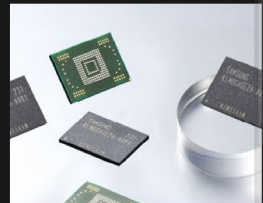
SINGLE-BOARD COMPUTERS



DISPLAYS



DEVELOPMENT



FLASH STORAGE

Big excitement, small packages

Thrill your inner geek

**ODROIDS
AHORA ESTA
DISPONIBLE EN LOS
ESTADOS UNIDOS**

WWW.AMERIDROID.COM

ENVIOS ECONOMICOS