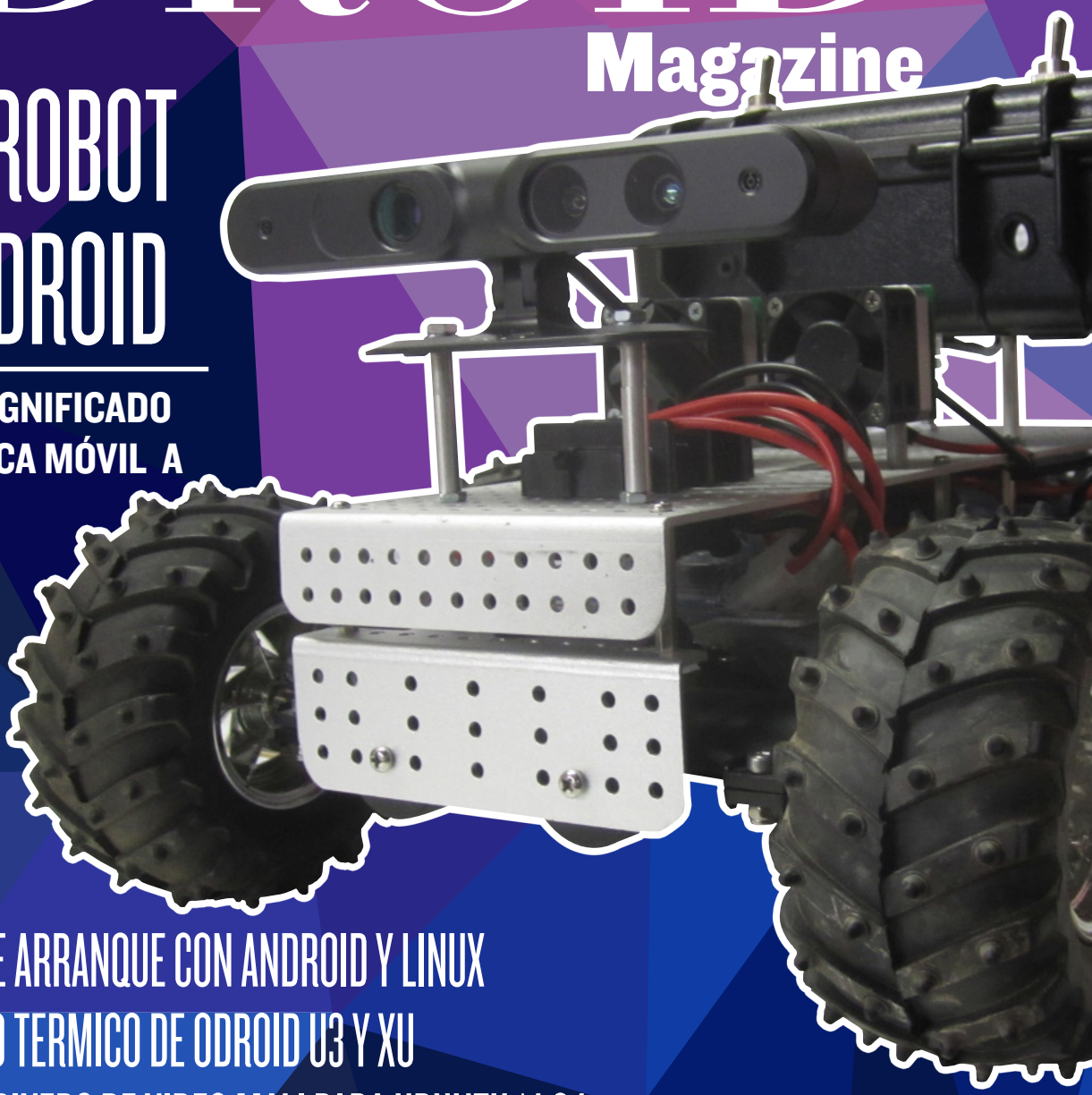
Proceso de arranque de Android • Compila y juega al clásico DOOM

ODROID Magazine

Año Uno
Num #5
May 2014

CREA UN ROBOT CON TU ODROID

EL VERDADERO SIGNIFICADO DE LA INFORMÁTICA MÓVIL A TU ALCANCE!



Y ADEMAS:

- DOBLE SISTEMA DE ARRANQUE CON ANDROID Y LINUX
- COMPORTAMIENTO TERMICO DE ODROID U3 Y XU
- RECOMPILA LOS DRIVERS DE VIDEO MALI PARA UBUNTU 14.04
- SISTEMA OPERATIVO DESTACADO: FULLY LOADED

¡HARDKERNEL LO HACE DE NUEVO! AMPLIA TU ODROID U3 CON:

ODROID-UPS: ALIMENTACIÓN ININTERRUMPIDA

ODROID-SHOW: COMPATIBLE CON ARDUINO

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 para alcanzar todo lo que puedas imaginar.

HARDKERNEL

Pollin ELECTRONIC

Ahora estamos enviando los dispositivos ODROID U3 a los países de la UE! Venga y visite 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://www.pollin.de/shop/suchergebnis.html?S_TEXT=odroid&log=internal`

EDITORIAL

Nuestra edición "Hazlo Tú Mismo!" del mes pasado fue uno de los números más populares que hemos tenido hasta la fecha, y realmente puso de manifiesto lo flexible y útil que pueden llegar a ser las placas ODROID para realizar grandes proyectos de calidad a un precio muy asequible en tu propia casa. Las últimas y emocionantes noticias proceden de Chris McMurrough, que publica Ubuntu Robotics Edition en los foros ODROID y cuenta con un gran talento para la robótica y la automatización de hardware. Su robot vehículo todoterreno automático que parece sacado directamente del programa espacial de la NASA, demuestra lo fácil que es construir tu propio robot usando un ODROID y algunas piezas fáciles de obtener.

Hardkernel ha vuelto a crear una par de periféricos muy útiles para el ODROID-U3: el ODROID-Show y ODROID-UPS. El ODROID-SHOW es un panel LCD de 2.2" con 320x240 píxeles que se conecta a cualquier ordenador y es capaz de visualizar texto, estadísticas, imágenes y otra información útil en tiempo real. Encaja perfectamente en la parte superior del ODROID U3. Con un precio razonable se presenta como una muy buena alternativa a la pantalla de texto de 2 líneas de Arduino One.

El ODROID-UPS (Sistema de Alimentación Ininterrumpida) está diseñado para mantener tus aplicaciones imprescindibles funcionando si hay un corte eléctrico. En lugar de usar un costoso y voluminoso sistema de baterías, ODROID-UPS coge en la palma de tu mano y puede mantener el sistema encendido durante un máximo de 4 horas sin necesidad de recarga, dependiendo de la carga de trabajo. Nuestro artículo también incluye un simple script de ejemplo para apagar ODROID con seguridad cuando la corriente eléctrica no esté disponible. Tanto el ODROID-SHOW y ODROID-UPS están disponibles en la tienda Hardkernel en `http://hardkernel.com/main/shop/good_list.php`.

También se incluye en este número una guía de principiantes para grabar archivos de imagen con sistemas operativos pre-compilados, lo cual pueden ser un reto para aquellos que están acostumbrados al típico método de PCs que incluye "disco de instalación". Los archivos de imagen permiten instalar de un modo simple un sistema operativo completo en un solo paso, y ahorrar horas de configuración para conseguir que tu ODROID arranque y funcione con rapidez. Para los usuarios más avanzados, presentamos una cuestión muy solicitada: Cómo configurar un doble sistema de arranque con Android y Ubuntu en un solo disco.

Nuestros habituales columnistas han estado trabajando duro para crear guías paso a paso para los aficionados al software. Tobías presenta una completa guía sobre la compilación de sus juegos favoritos, tales como Doom, Yo describo las características de la popular imagen de la comunidad Fully Loaded, Jussi nos muestra una comparación de ODROID-U3 y ODROID-XU cuando se ejecutan con una elevada carga de trabajo, y Nanik nos ayuda a entender cómo funciona el proceso de arranque de Android . Vamos a continuar con los códigos de colores en los títulos del artículo con el fin de clasificarlos según su nivel: Principiante, Intermedio o Experto, para asegurarnos que siempre hay algo para todos.

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

Robert Hall, Editor Jefe

Soy un programador informático que vive y trabaja en San Francisco, CA, en el diseño y desarrollo sitios web. 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 cluster de 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**.

Bo 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 mías ya que tengo demasiadas. Eso ayudaría a que tuviese más tiempo para estar con mi maravillosa esposa de 23 años y mis cuatro hijos estupendos.

Bruno Doiche, Editor Artistico

Estoy pensando acudir a un centro de rehabilitación de abstinencia al café después de descubrir el café Airpress H20 y beberlo por cubos. Además de gestionar un cinturón negro en reorganizaciones multinacionales. Pero no le pregunte cuál era su trabajo en ese momento pues que no podía ni siquiera explicarlo. Sólo que necesita gran cantidad de dispositivos de almacenamiento SAN y toneladas de burocracia.

Manuel Adamuz, Traductor

Me gusta tanto el mundo ODROID que últimamente tengo hasta pesadillas por la noches.

Novedades de esta Edición:

Y seguimos haciendo cambios! Los foros realizan peticiones, nosotros las tenemos en cuenta siempre que tengas sentido. Tenemos un montón de solicitudes para eliminar las pantallas de terminal en color verde con el fin de facilitar la impresión de la revista. Ahora hemos unificado el código monoespaciado en los bloques amarillos. Junto con nuestro nuevo sistema de columnas, viene bastante bien. Además, para que quede aún más claro el nivel de cada artículo, colocaremos una estrella para diferenciar la habilidad exi-gida para cada artículo.

También hay mayor flexibilidad en el espaciado de la fuente en algunos bloques de código, donde ahora estamos usando una fuente más pequeña para que las cosas se vean un poco más claras (aunque más difíciles de editar) y fácil de leer. Créame, a mi es al que le crea más frustración estos pequeños problemas gráficos.

Para terminar, y para que veáis que existe cierta sincronización con el foro, vi una sugerencia de un usuario que utiliza cowsay para saludar a los usuarios. Esto estaba reservado para la columna de trucos y consejos de linux de este mes, pero no se ha incluido en esta edición. Voy proponer uno este mes:

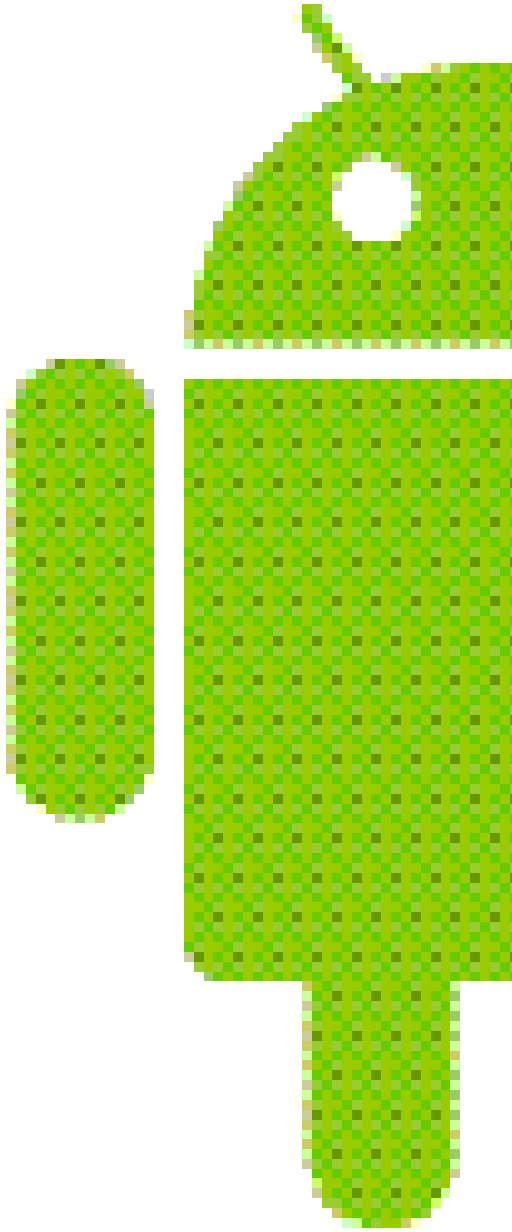
```
 _____________________________________
/ Te gusta hacerlo instantaneamente,  \
\ pero es demasiado lento.            /
 -------------------------------------
  \
   \____
  /@    ~-.
  \/ __ .- |
   // //  @
```

INDICE

PROCESO DE INICIO DE ANDROID

ENTENDER LOS ENTRESIJOS DE COMO ARRANCA ANDROID EN TU ODROID

por Nanik Tolaram



Todas las plataformas informáticas, incluido ODROID-U3, tienen alguna forma predefinida de arrancar y cargar el sistema operativo. Por ejemplo, en un PC convencional, la BIOS será la primera en ejecutarse y la siguiente serie de acontecimientos en la secuencia de arranque es controlada por el microprocesador. Las diferentes arquitecturas de procesadores tienen sus propios modos de arranque y los procesadores ARM arranca de un modo distinto al de un procesador x86. En este artículo, vamos a ver lo que sucede desde que se conecta el ODROID U3 hasta que aparece en pantalla Android.

La Figura 1 muestra de forma general lo que sucede durante el proceso de arranque. Usaremos este esquema para describir cada uno de los pasos que tienen lugar.

ROM de Arranque

Este es el primer programa que ejecuta el microprocesador cuando se inicia, reside "dentro" del procesador y es instalado habitualmente por el fabricante. Este programa ejecuta decenas de kilobytes y su principal función es la preparar el hardware para que el proceso de arranque pueda continuar con el siguiente paso. Esta ROM es algo "misterioso" para los desarrolladores, ya que su código fuente por lo general no suele estar disponible por los derechos de propiedad. La ROM es responsable de iniciar y preparar los componentes de arranque para su posterior uso, del mismo modo que la siguiente fase comenzará con la lectura de la memoria externa (tarjeta SD, eMMC, etc) con el fin de continuar con el proceso de arranque.

Gestor de Arranque

Activar el hardware en la fase de ROM es decisivo para el gestor de arranque, ya que éste reside en un dispositivo de almacenamiento. En el caso del ODROID U3 este dispositivo es la SD o módulo eMMC. Durante la ejecución del código de la ROM, se leerán los ficheros binarios del gestor de arranque del dispositivo de almacenamiento y comenzará a ejecutarse el código cargado.

Como se puede ver en la Figura 2, para la plataforma ODROID-U3 existen un par de archivos que se usan como gestores de arranque. Los ficheros bl1.bin son gestores de arranque propiedad de Samsung que contienen una serie de tareas que activan aún más el hardware. El código fuente del gestor de arranque u-boot está disponible para ODROID-U3 y se puede modificar y volver a compilar fácilmente. Los 2 principales ficheros binarios (bl1.bin y bl2.bin) son archivos que han sido firmados y son necesarios para el arranque. El archivo bl1.bin será el primero en buscar y ejecutar cuando la ROM de arranque complete su ejecución.

Al finalizar bl1.bin, el código buscará bl2.bin y continuará desde este punto. No podemos saber qué es exactamente lo que hace bl1.bin, puesto que no conocemos a nadie de Samsung. El otro archivo bl2.bin se genera como parte del

Figura 1 : Proceso general de Arranque

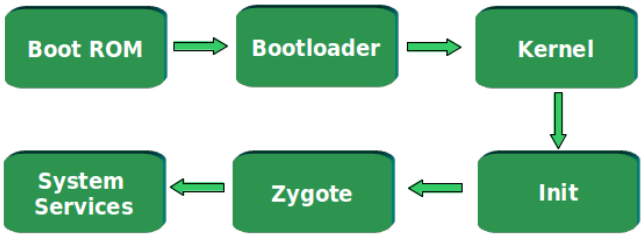
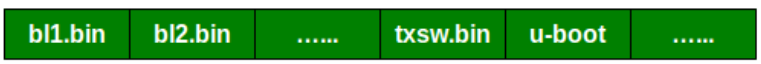
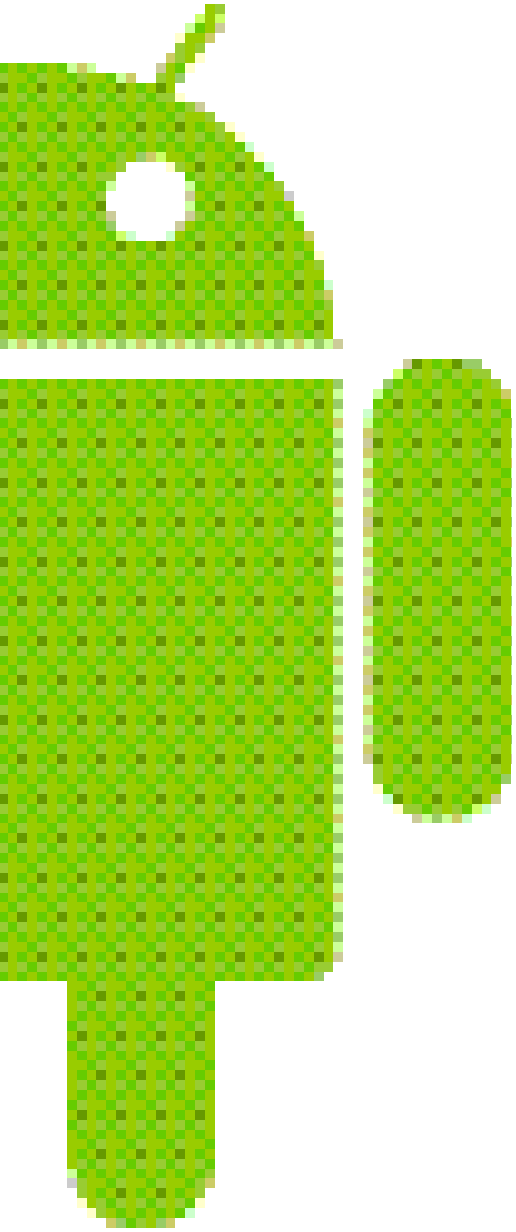


Figura 2 : Esquema del Gestor de Arranque de la tarjeta SD





desarrollo de u-boot. Este archivo tiene que ser firmado por Hardkernel para que funcione con el ODROID-U3.

Por último, el archivo tzsw.bin es un archivo propiedad de Samsung/ARM que permite que el código sea ejecutado de forma segura.

Kernel

Una vez que u-boot se ha completado correctamente, se carga el kernel de Linux y se ejecuta. Este kernel es el mismo que se usa para arrancar Ubuntu o cualquier otra distribución de Linux, pero contiene controladores específicos de Android necesarios para su ejecución. Este paso no tiene nada de especial, puesto que es el mismo kernel al que estamos acostumbrados a ver cuando arranca Linux.

Figura 3 - La Aplicación Init

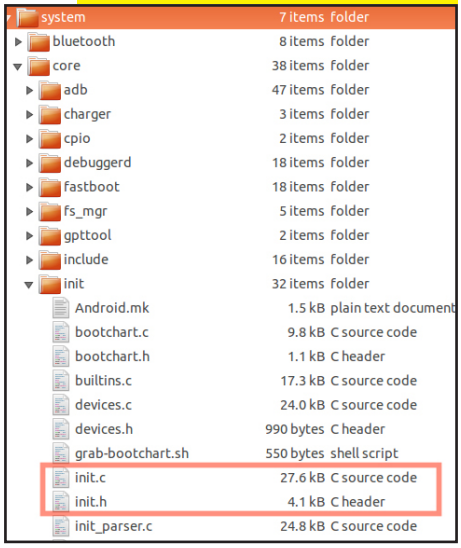


Figura 4 - Otros ficheros .rc dentro de init.rc

```
import /init.${ro.hardware}.rc
import /init.usb.rc
import /init.trace.rc
```

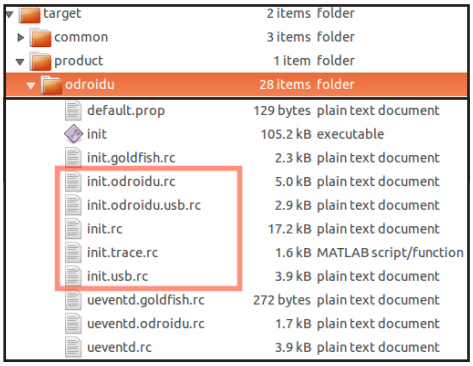


Figura 5 - Lista completa de ficheros .rc de init del ODROID-U3

Init

Cuando el kernel de Linux ha completado todas las tareas de iniciación y todos los servicios están ejecutados, se ejecuta un programa denominado init. De aquí en adelante todo lo que se inicia está relacionado con Android o lo que se conoce en el mundo Linux como la capa de espacio de usuario, desde la carga de controladores HAL (capa de abstracción de hardware) hasta ejecutar aplicaciones. La aplicación init en Adroid es distinta a la de un entorno normal de Linux y reside en la carpeta /system del código fuente de Android.

La principal función de la aplicación init es activar todos los directorios, permisos, servicios, propiedades y variables de entorno necesarios. La aplicación init funciona con un archivo de configuración llamado init.rc dentro del directorio out/target/product/odroidu/root. El init.rc simplemente contiene otros archivos .rc.

El init.rc contiene lo que se conoce como lenguaje Init de Android. La descripción de los diferentes comandos disponibles puedes localizarla en **`http://bit.ly/1kfCibb`**. Vamos a analizar el contenido de los archivos .rc en detalle.

```
on boot
    setprop ro.build.product odroidu
    setprop ro.product.device odroidu
    setprop ro.radio.noril yes
    setprop ro.kernel.android.ril 0
    setprop ril.work 0

# Run sysinit
    start sysinit
```

El fragmento anterior esta sacado del archivo init.odroid.rc con el comando *on boot*. Su finalidad es establecer diferentes propiedades de entorno para ro.build.product, ro.radio.noril y otros servicios. Estas diferentes variables de entorno son usadas internamente por el entorno de trabajo de Android como parte del proceso de iniciación. También se inicia un servicio denominado sysinit. La siguiente lista muestra la secuencia de comandos de ejecución cuando init completa la lectura de init.rc:

```
early-init
init
early-fs
```

PIPE VIEWER
CONSIGUE MÁS INTERACTIVIDAD CON TUS PROGRESOS DE DATOS

por Bruno Doiche

Cansado de no conseguir ningun resultado con *dd* siempre que necesitas hacer backup o grabar en tu eMMC? Entonces instala pv, es un programa que se puede colocar entre 2 procesos para gestionar la entrada y salida estándar visualizando el progreso de la operación. En primer lugar, debes instalar:

```
sudo apt-get install pv
```

y úsalo de esta forma:

```
dd if=/dev/sdX bs=1M | pv | dd of=/dev/sdY
```

Obtendrás un resultado del progreso, ¡mira!

```
Output:
1,74MB 0:00:09 [198kB/s] [<=>]
```

Ahora ya no tiene que adivinar si tu *dd* funciona o no y cuánto ya se ha copiado.

LA FUEZA
ES PODER CON TRACEROUTE

por Bruno Doiche

Nada que hacer en tu Terminal? Bueno, a continuación, pasar un buen rato ejecutando el siguiente comando:

```
traceroute 216.81.59.173
```

Espere unos pocos de saltos, y verá el texto de una extraña película familiar.


```
fs
post-fs
post-fs-data
charger
early-boot
boot
```

La función principal de la aplicación init es iniciar los diferentes servicios de Android que son necesarios para ejecutar el entorno de trabajo de Android al completo. Echemos un vistazo a algunos de los servicios incluidos dentro de init.rc.

```
service servicemanager /system/bin/servicemanager
    class core
    user system
    group system
    critical
    onrestart restart zygote
    onrestart restart media
    onrestart restart surfaceflinger
    onrestart restart drm
```

El comando *service* anterior inicia el servicio ServiceManager. Esta aplicación y otros ficheros binarios residen dentro de Android en el directorio /system/bin, y guardan un registro de los diferentes servicios que Android arranca durante el proceso init. Los parámetros que aparecen después del comando *service* describen las características de ServiceManager

Hay una línea en particular que quiero destacar: *onrestart restart zygote*. El comando *onrestart* indica cuando la aplicación ServiceManager es reiniciada, también necesita reiniciar el servicio zygote. Esto es importante porque zygote es la aplicación clave necesaria para que tus aplicaciones Java funciones, de modo que si el ServiceManager falla o se reinicia, la aplicación ejecutada se detendrá.

El comando *service* es el método preferido para lanzar servicios del sistema durante el proceso de arranque, por lo que cuando se ejecuta el comando *ps* desde el interprete de comando de Android verás un resultado como el que aparece en la Figura 6.

Si abres init.rc y compruebas el comando *service*, verás que la mayoría de los servicios necesarios están ejecutados en el sistema. El init.rc es el lugar donde todos los

Figura 6: Aplicación ejecutada con el comando service

```
system    1376  1     904    176   c048c344 4007ab64 S /system/bin/servicemanager
root      1377  1     5672   1096  ffffffff 4010d2f0 S /system/bin/vold
root      1383  1     9684   932   ffffffff ffff0520 S /system/bin/netd
root      1384  1     956    224   c04c5ed0 400d25f4 S /system/bin/debuggerd
system    1385  1     58064  5752  ffffffff 400a5b64 S /system/bin/surfaceflinger
root      1386  1     453652 36688 ffffffff 400fbc88 S zygote
drm       1387  1     11116  3280  ffffffff 4017eb64 S /system/bin/drmserver
media     1388  1     27808  6400  ffffffff 4008ab64 S /system/bin/mediaserver
bluetooth 1389  1     1416   732   c0148060 40109ab8 S /system/bin/dbus-daemon
root      1390  1     916    200   c0580254 400b392c S /system/bin/installd
keystore  1391  1     1932   680   c04c5ed0 400135f4 S /system/bin/keystore
root      1394  1     832    328   c031b7dc 400d092c S /system/bin/sh
root      1395  1     4468   4     ffffffff 00013380 S /sbin/adbd
root      1588  2     0      0     c015e794 00000000 S flush-179:0
root      1592  2     0      0     c009dd90 00000000 S kworker/0:3
root      1603  1     1088   384   c052dc64 400415f4 S /data/gatord/gatord
system    1607  1386  539504 41136 ffffffff 400fbb64 S system_server
u0_a26    1684  1386  480536 56148 ffffffff 400fca40 S com.android.systemui
u0_a30    1732  1386  465928 32900 ffffffff 400fca40 S com.android.inputmethod.latin
u0_a14    1744  1386  468620 32492 ffffffff 400fca40 S android.process.media
radio     1766  1386  475188 31536 ffffffff 400fca40 S com.android.phone
u0_a6     1783  1386  525588 44204 ffffffff 400fca40 S com.android.launcher
```

servicios vitales están definidos, junto con las dependencias de las que depende cada servicio, incluyendo las características (usuario, grupo, reinicio, etc) de cada servicio. Un fallo en la ejecución de cualquiera de los servicios definidos dará como resultado un Android inoperativo así como cualquier aplicación de usuario.

Zygote

Como hemos visto en el apartado anterior Android se inicia con una serie de servicios de los que depende, incluyendo Zygote. Es importante señalar que Zygote es el nombre del servicio que se da en Android a una aplicación que se encarga de "ejectuar" aplicaciones de usuario a través de la máquina virtual Dalvik, como puede verse en el servicio que se muestra a continuación:

```
   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
```

Cuando el servicio arranca, se creará una conexión local que es usada por el entorno de trabajo interno para lanzar aplicaciones. En resumen, zygote es una capa de conexión que se encarga de la ejecución de aplicaciones de usuario. Todas las aplicaciones Android que usas en tu dispositivo (teléfono, tablet, etc.) son "lanzadas" mediante zygote, si éste no está operativo tu aplicación no podrá ser ejecutada.

Vamos a ver qué sucede si apagamos zygote desde la línea de comandos. Usa el intérprete de comando adb (adb shell) para conectarte a tu ODROID-U3 y ejecutar el comando *stop zygote*. Inmediatamente verás como se apaga el sistema de Android. Para iniciar de nuevo zygote, sólo tienes que escribir *start zygote*.

Servicios del Sistema

Este es el paso final en el proceso de arranque, y además es esencial para hacer la vida más fácil a los desarrolladores. Estos servicios son una combinación de código nativo y Java que existen para cubrir las necesidades de las aplicaciones de usuario y servicios, tales como USB, acelerómetro, Wifi y muchos más. Al escribir una aplicación para Android, es inevitable encontrarse con estos servicios y su uso, ya sea directa o indirectamente.

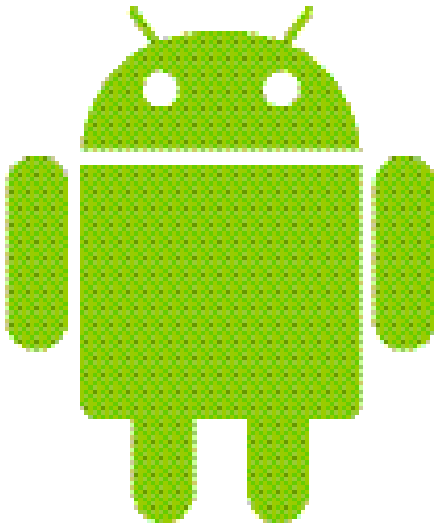
```
root@android:/ # service list
Found 70 services:
0       sip: [android.net.sip.ISipService]
1       phone: [com.android.internal.telephony.ITelephony]
2       iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
3       simphonebook: [com.android.internal.telephony.IIccPhoneBook]
4       isms: [com.android.internal.telephony.ISms]
5       commontime_management: []
6       samplingprofiler: []
7       diskstats: []
8       appwidget: [com.android.internal.appwidget.IAppWidgetService]
9       backup: [android.app.backup.IBackupManager]
10      uimode: [android.app.IUiModeManager]
11      serial: [android.hardware.ISerialManager]
12      usb: [android.hardware.usb.IUsbManager]
13      audio: [android.media.IAudioService]
14      wallpaper: [android.app.IWallpaperManager]
15      dropbox: [com.android.internal.os.IDropBoxManagerService]
16      search: [android.app.ISearchManager]
17      country_detector: [android.location.ICountryDetector]
18      location: [android.location.ILocationManager]
19      devicestoragemonitor: []
20      notification: [android.app.INotificationManager]
21      updatelock: [android.os.IUpdateLock]
22      throttle: [android.net.IThrottleManager]
23      servicediscovery: [android.net.nsd.INsdManager]
24      connectivity: [android.net.IConnectivityManager]
25      ethernet: [android.net.ethernet.IEthernetManager]
26      wifi: [android.net.wifi.IWifiManager]
27      wifip2p: [android.net.wifi.p2p.IWifiP2pManager]
28      netpolicy: [android.net.INetworkPolicyManager]
```

Figura 6 : Aplicación ejecutada desde el comando service

Sin servicios, se necesitaría mucho tiempo y esfuerzo para escribir aplicaciones en Android. Imagina que deseas escribir una aplicación basada en USB, pero no hay servicio disponible. Tendrías que escribir un montón de código tanto en Java como en capa nativa para que la aplicación pudiera acceder a los puertos USB. Puedes ver los servicios disponibles actualmente usando el comando *service list* desde el intérprete de comandos ADB.

La clase se encarga de la correcta ejecución de los servicios que se encuentran en frameworks/base/services/java/com/android/server/SystemServer.java. Si tienes un proyecto de hardware que necesita proporcionar servicios a los desarrolladores, es mejor tenerlo funcionando como un servicio de Android para que el código de la aplicación cliente no necesite ser reescrito si la interfaz necesita actualizaciones o cambios.

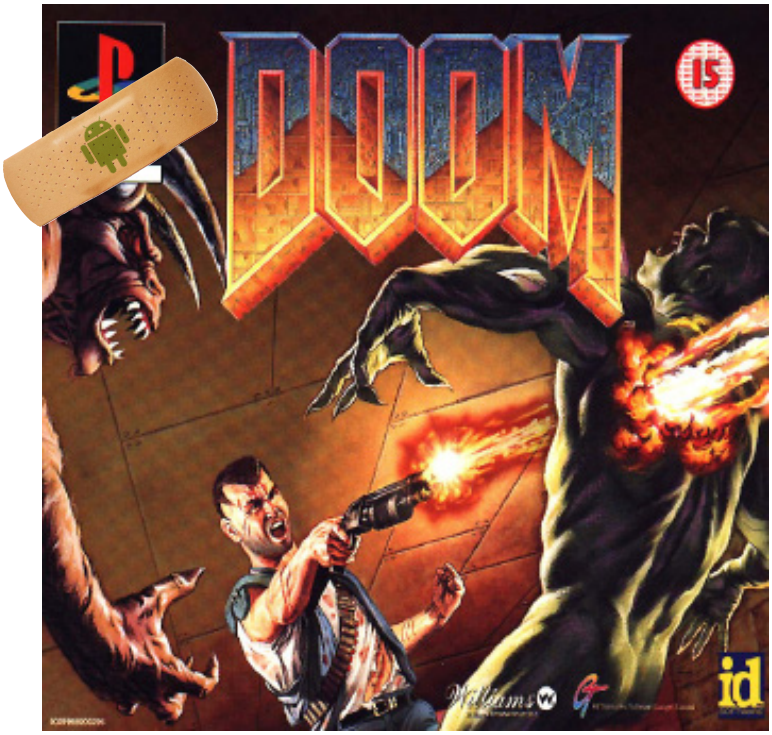
Nanik Tolaram vive en Sydney con su esposa y 2 niños. Su trabajo diario es hacer frente al código fuente de Android, personalizarlo, solucionar problemas y mejorarlo para asegurarse de que funciona con el hardware (ARM y x86). Sus pasatiempos incluyen la cría de peces, dar clases de Android y electrónica y el bricolaje. También se encarga de los sitios web de Android **www.ozandroid.info** *y* **kernel.ozandroid.info**



COMO COMPILAR DOOM EN TU ODROID

JUEGA A ESTE CLASICO DE SIEMPRE ADAPTADO Y COMPILADO PARA ODROID

por Tobias Schaaf



Ahora incluso tienen una porada para tu reciente desarrollo de DOOM en tu ODROID

He compilado y publicado muchos juegos para ODROID, y a menudo recibo peticiones de usuarios con información de cómo hacer lo mismo con sus juegos favoritos. Como ejemplo de cómo compilar tus propios juegos y aplicaciones, te presento una completa guía en la que compilo el juego Doom de id Software para las series U o X de ODROID.

Para empezar debes tener al menos dos copias de ODROID Game Station Turbo, descárga la imagen desde los foros ODROID y pásala a las tarjetas SD. Aunque no es completamente necesario tener dos imágenes, probablemente quieras reutilizar lo que ya has hecho en caso de tener que reinstalar el sistema operativo. Además, a pesar de que tienes todo lo necesario para ejecutar ciertos juegos en la imagen que compiles, necesitarás otra imagen de "reserva" para probar el script de instalación con el fin de asegurar que todas las librerías que necesitas están incluidas en la pack final.

No recomiendo compilar sobre una eMMC, ya que reducirás bastante la vida útil de tu módulo eMMC con el paso del tiempo. Ya he estropeado 3 o 4 tarjetas SD y una memoria USB usándolas en esta compilación tan pesada, y he tenido que cambiar a un disco duro estándar de 1 TB.

WGET

Wget es una herramienta bastante fácil de usar. Permite descargar de una forma sencilla archivos individuales desde Internet al introducir una URL como argumento de comando. En este caso, le proporcionamos la dirección de la versión SDL de Doom:

```
wget http://www.libsdl.org/projects/doom/src/sdl-doom-1.10.tar.gz
```

Este comando descarga el archivo sdldoom-1.10.tar.gz en el directorio actual. Recuerda que wget no permite descargar una carpeta o varios archivos al mismo tiempo. Únicamente funciona con archivos individuales.

Configurar un entorno de desarrollo

Esta es una lista de programas que te recomiendo instalar antes de comenzar la compilación:

```
apt-get install build-essential cmake automake autoconf git subversion checkinstall
```

Es probable que necesites muchas más aplicaciones, pero es un buen comienzo. A continuación crea una carpeta en la cual desarrollarás tus ficheros binarios.

```
mkdir sources
```

Oh!!! Recuerdo siendo un adolescente en apuros intentando matar cyberdemons! Durante mucho tiempo pensé que era el jefe final del juego porque he muerto en tantas ocasiones.

Esta carpeta puede ubicarse en un dispositivo externo, como una memoria USB o disco duro, lo cual facilita su uso en una imagen diferente, un ODROID distinto, o incluso otro PC.

Comienza con simple desarrollo

Para empezar a desarrollar la aplicación SDL Doom escriba lo siguiente:

```
cd sources
wget http://www.libsdl.org/projects/doom/src/sdldoom-1.10.tar.gz
tar xzvf wget sdldoom-1.10.tar.gz
cd sdldoom-1.10
```

En la carpeta sdldoom-1.10 encontrarás un archivo muy importante llamado "configure", que muchos programas incluyen en su código fuente.

Configure

Configure es un programa que analiza tu sistema y comprueba que tu entorno de desarrollo disponga de las piezas necesarias para compilar el programa. Normalmente te informa de los archivos que faltan y a menudo, te permite añadir parámetros especiales para ajustar tu desarrollo. Para ver qué parámetros ofrece el programa que deseas compilar puedes iniciar *configure* con el parámetro *--help*.

```
./configure --help
```

Hay una gran cantidad de parámetros pero no te preocupes, la mayor parte no son necesarios para los desarrollos estándar. Por defecto, al escribir *make install* copiarás todos los archivos compilados a los directorios del sistema, tales como /usr/local/bin y /usr/local/lib. Si deseas cambiar el directorio de instalación, puedes especificar un prefijo de instalación diferente a /usr/local usando la opción *--prefix*, por ejemplo, --prefix=$HOME.

Compilando el código fuente de SDL Doom

Normalmente, hay un archivo README que indica las dependencias que son necesarias. Esta versión de Doom en realidad no tiene este tipo de archivo README. Sin embargo, el comando configure destacará las dependencias que faltan. En el siguiente ejemplo, falta una librería llamada "libsdl1.2-dev":

```
./configure
loading cache ./config.cache
checking for a BSD compatible install... (cached) /usr/bin/install -c
[...]
checking for sdl-config... (cached) /usr/bin/sdl-config
checking for SDL - version >= 1.0.1...
./configure: 1: ./configure: /usr/bin/sdl-config: not found
./configure:1:./configure:/usr/bin/sdl-config: not found
no
*** Could not run SDL test program, checking why...
*** The test program failed to compile or link. See the file config.log for the
*** exact error that occured. This usually means SDL was incorrectly installed
*** or that you have moved SDL since it was installed. In the latter case, you
*** may want to edit the sdl-config script: /usr/bin/sdl-config
configure: error: *** SDL version 1.0.1 not found!
```

Aquí es donde el proceso de compilación es un poco más complicado y se necesita algo de experiencia para entender cier-

tas cosas. En este caso, *configure* informa que no pudo encontrar sdl-config y también muestra que la versión 1.0.1 de SDL no fue localizada. Puesto que los mensajes de código fuente tienden a ser muy genéricos y no muestran EXACTAMENTE lo que necesitamos instalar, sino que más bien ofrecen una orientación de como debemos trabajar.

En este caso, indica que el programa se llama SDL, pero en realidad el archivo exigido es un programa llamado "libsdl1.2-dev". En la mayoría de los casos, necesitarás algo de experiencia para entender lo que realmente hace falta, pero existen un par de herramientas que pueden ayudarte en este sentido.

```
apt-cache search
```

Con el comando *apt-cache search* puedes localizar determinados paquetes de los que no conoces el nombre completo.

```
apt-cache search SDL
```

Dependiendo de los términos de búsqueda estas listas pueden ser bastante largas, ya que se busca por nombre de archivos y descripción. Es mejor buscar algo que no sea muy común como el término "libsdl":

```
apt-cache search libsdl
```

Al usar el prefijo "lib" la lista es más corta y muestra, en efecto que hay unas cuantas librerías que empiezan por libsdl. Es importante recordar que al compilar un paquete, las librerías dependientes son siempre librerías de desarrollo que contiene los archivos cabecera. Así que cuando ejecutamos *apt-search*, sólo noss interesa las librerías que terminan en "-dev". Podemos buscar justamente estas librerías dev agregando otra palabra clave a la búsqueda:

```
apt-cache search libsdl dev
```

Con los términos de búsqueda adicionales, la lista de resultados es más breve. Muestra una librería libsdl1.2-dev que coincide con la versión obtenida con el comando c*onfigure*, el cual requiere la versión 1.0.1 o superior. La instalamos escribiendo:

```
apt-get install libsdl1.2-dev
```

Ahora que el paquete libsdl1.2-dev está instalado, vamos a probar *configure* de nuevo:

```
  ./configure
  loading cache ./config.cache
  checking for a BSD compatible install...
(cached) /usr/bin/install -c
  checking whether build environment is
sane... yes
  checking whether make sets ${MAKE}...
(cached) yes
  checking for working aclocal... found
  checking for working autoconf... found
  checking for working automake... found
  checking for working autoheader... found
  checking for working makeinfo... missing
  checking whether make sets ${MAKE}...
(cached) yes
  checking for gcc... (cached) gcc-4.7
  checking whether the C compiler (gcc-4.7
) works... yes
  checking whether the C compiler (gcc-4.7
) is a cross-compiler... no
  checking whether we are using GNU C...
(cached) yes
  checking whether gcc-4.7 accepts -g...
(cached) yes
  checking for a BSD compatible install...
/usr/bin/install -c
  checking for sdl-config... (cached) /usr/
bin/sdl-config
  checking for SDL - version >= 1.0.1...
yes
  creating ./config.status
  creating Makefile
```

Rostros sonrientes, armas en llamas - Esto es lo que lo convierte en uno de los mejores juegos de la historia! Jugar toda la noche... No me arrepiento!

Ahora que no hay errores, ¡Ya estamos listos! Observa la última línea: "creating Makefile". Un Makefile es algo que siempre necesitas casi todo el tiempo. Si tienes un Makefile en tu carpeta simplemente puedes escribir "*make*" y comenzará automáticamente la compilación de tu programa. También puede agregar el parámetro *make-j4* que utiliza 4 hilos de ejecución para crear un programa, lo cual aumenta la velocidad de desarrollo al utilizar los 4 núcleos del ODROID.

Información avanzada sobre configure

A veces configure muestra que falta algo así como "-lSDL_mixer". "-l" nos dice que se trata de una librería y el resto te dice que busques algo con mixer.

```
apt-cache search libsdl mixer dev
libsdl-mixer1.2-dev - Mixer library for Simple DirectMedia Layer 1.2, development files
```

Otras veces, la fase de configuración puede reclamar algunos archivos cabecera (archivos que terminan en. h) como "SDL/SDL_net.h". Hay sitios en la red para Debian y Ubuntu donde puedes buscar archivos dentro de determinados paquetes y averiguar los nombres de los paquetes que faltan:

Ubuntu: `http://bit.ly/PSihOu`
Debian: `http://bit.ly/1rQEbzW`

Completando el desarrollo

Ahora deberías tener un programa totalmente funcional llamado "doom" en tu carpeta y se puede ejecutar desde aquí si pinchar en el archivo .wad. Todo hecho, ¿verdad?

Bueno, no del todo. Doom está desarrollado y se puede jugar copiando tus archivos .wad en la misma carpeta, pero aún no se ha "instalado". La mayoría de los programas te permiten hacer un simple *make install* el cual copiará todos los archivos necesarios a la carpeta correcta para completar la instalación.

```
make install
make[1]: Entering directory `/home/odroid/sources/sdldoom-1.10`
/bin/sh ./mkinstalldirs /usr/local/bin
  /usr/bin/install -c doom /usr/local/bin/doom
make[1]: Nothing to be done for `install-data-am`.
make[1]: Leaving directory `/home/odroid/sources/sdldoom-1.10`
```

doomsdl es el único archivo binario (doom) que se instala, pero dependiendo de lo que estés desarrollando, la instalación puede ser mucho mayor e instalar miles de archivos.

Entonces, ¿cómo lo instalas en otro sistema? Puedes copiar los archivos manualmente pero si son muchos, puede llevarte tiempo y no incluir todas las dependencias y librerías necesarias para ejecutar la aplicación. Aquí es donde *check install* resulta muy útil, el cual crea archivos portátiles .deb que se pueden instalar en otro sistema. En el artículo del próximo mes, explicaré cómo usar *check install* para copiar Doom a una nueva instalación de Linux.

RECOMPILA LOS DRIVERS DE VIDEO MALÍ

CORREGIR LOS PROBLEMAS DE LA "PANTALLA EN BLANCO" Y "VENTANAS RETARDADAS" AL ACTUALIZAR A UBUNTU 14.04

por Rob Roy

Al actualizar a Ubuntu 14.04, es probable que la versión actual del Xorg Server 1.14, el software responsable de crear la interfaz gráfica de usuario, ya no sea compatible con los controladores de vídeo Mali que funcionaban con Ubuntu 13. Para compilar la última versión del software Malí, escriba lo siguiente en una ventana de terminal o una sesión SSH:

```
wget http://malideveloper.arm.com/downloads/drivers/DX910/r3p2-01rel4/DX910-SW-99003-r3p2-01rel4.tgz
tar xzvf DX910-SW-99003-r3p2-01rel4.tgz
cd DX910-SW-99003-r3p2-01rel4/x11/xf86-video-mali-0.0.1/
sudo apt-get install autoconf xutils-dev libtool xserver-xorg-dev
cd src/
rm compat-api.h

wget http://cgit.freedesktop.org/~cooperyuan/compat-api/plain/compat-api.h
cd ..
./autogen.sh
./configure --prefix=/usr
make
sudo make install
```

La versión más popular de los drivers de Mali 400 de abril 2014 es r3p2-01rel4. Puedes comprobar si hay una versión actualizada del software Malí visitando el sitio del desarrollador en **http://bit.ly/1f5Jk51.**

Después de reiniciar, la versión del Xorg Server (14) coincidirá con la versión del controlador de Malí y la señal HDMI deberá funcionar correctamente. Si la señal HDMI está funcionando, pero las ventanas de escritorio se mueven lentamente, escribe las líneas de abajo para restaurar los controladores de Malí tras el reinicio. La configuración de Malí se almacena en /etc/X11/xorg.conf. Cambiar el nombre del archivo de configuración evita que se reemplaza la versión de Mali.

```
cd /etc/X11/xorg.conf.d
sudo mv exynos.conf exynos.conf.original
sudo reboot
```

2 SISTEMAS, 1 ODROID, PURA DIVERSION!

CÓMO HACER UN DOBLE SISTEMA DE ARRANQUE CON ANDROID Y UBUNTU

por Yong-Oh Kim, Desarrollador Hardkernel

Ubuntu y Android son dos sistemas operativos muy populares para ODROID y se pueden combinar para ejecutarse juntos de varias formas. Puedes crear un sistema "multi-placa" instalando cada sistema operativo en distintos ODROIDs y conectarlos a través de protocolos de comunicación, ejecutar una versión de Linux dentro de Android con el comando *chroot*, conectarlos mediante un cable USB para la transferencia de archivos, o crear una conexión cliente-servidor mediante un servidor web y un entorno de desarrollo. En este artículo se describe otro método para combinar Ubuntu y Android: mediante la creación de un doble sistema de arranque flexible que permite a cualquiera de los sistema operativo ejecutarse desde el mismo disco duro.

Android ofrece muchas aplicaciones orientadas al consumidor como Netflix, Hulu, Skype, Google Hangout, modernos juegos 3D/2D, XBMC y otras aplicaciones destinadas al entretenimiento e interacción social. Por otro lado, Ubuntu ofrece la experiencia de un PC con muchas aplicaciones de productividad y utilidades para desarrolladores como LibreOffice, GIMP, servidor Apache, Eclipse, Arduino IDE, librerías OpenMP/CV, herramientas de programación C/C++/Java/Python y muchas otras aplicaciones técnicas.

Sin embargo, Android y Ubuntu son muy diferentes y el usuario puede desperdiciar mucho almacenamiento, así como no poder compartir contenidos entre ambos sistemas. Hay 5 pasos principales para poder arrancar con varios sistemas operativos:

-Modificar el código fuente de Android para MTP y desarrollar el código fuente de la plataforma.
-Coger el sistema de archivos raíz Xubuntu y modificar odroid-config para la nueva tabla de particiones.
-Crear nueva tabla de particiones y formatear particiones en tu eMMC.
-Copiar el sistema de archivos raíz de Xubuntu y cambia el UUID.
-Transferir del sistema de archivos raíz de Android via fastboot.

Para un mejor rendimiento, se recomienda usar un módulo eMMC o tarjeta SD de alta velocidad (20 MB/s), el sistema operativo Android tiene un mal funcionamiento cuando se utiliza una tarjeta SD demasiado lenta.

Modificar el código fuente de Android para el MTP

Descarga el archivo odroidu.zip desde http://bit.ly/PX-pjkR, luego descomprímelo en el directorio device/hardkernel/odroidu/ del código fuente de Android. También es necesario modificar el archivo/app/Utility/src/com/ hardkernel/odroid/MainActivity.java para acceder al MTP en Android en lugar de VFAT, como se muestra a continuación.

```
$ svn diff packages/apps/Utility/
Index:    packages/apps/Utility/src/com/
hardkernel/odroid/MainActivity.java
@@ -20,6 +20,7 @@
import android.app.Activity;
import android.content.Context;
import  android.content.SharedPreferenc-
es;
+import android.os.Environment;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
@@ -44,7 +45,7 @@
 public final static String MIN_FREQ_NODE
=    "/sys/devices/system/cpu/cpu0/cpufreq/
scaling_min_freq";
  //private final static String BOOT_INI
= "/storage/sdcard1/boot.ini"; //"/mnt/sd-
card/boot.ini";
- private final static String BOOT_INI =
"/mnt/sdcard/boot.ini";
```

```
   + private String BOOT_INI = "/mnt/sdcard/
boot.ini";
    public int mCurrentMaxFreq;
    public int mCurrentMinFreq;
   @@ -371,6 +372,14 @@
      tv.setVisibility(View.GONE);
     }

   +        File sdcard1 = new File("/stor-
age/sdcard1");
   +        if (sdcard1.exists()) {
   +            Log.e(TAG, "MTP");
   +          BOOT_INI = "/storage/sdcard1/
boot.ini";
   +        } else {
   +            Log.e(TAG, "Mass Storage");
   +        }
   +
     File boot_ini = new File(BOOT_INI);
     if (boot_ini.exists()) {
      try {
```

Tras compilar del código fuente de Android por completo, dispondrás del system.img qué contiene todo el sistema de archivos raíz de Android. Si no quiere compilar el código fuente de Android, puedes usar una imagen pre-compilada: `http://bit.ly/1i5ZJB3` o `http://bit.ly/1rWMMB9`.

Copiar el sistema de archivos raíz de Xubuntu

Se debe usar un PC con Linux para almacenar y transferir los archivos necesarios para el doble sistema de arranque.

```
   mkdir boot
   sudo   cp   -a   /media/codewalker/BOOT/*
boot/
   mkdir rootfs
   sudo   cp   -a   /media/codewalker/rootfs/*
rootfs/
```

Modificar el script odroid-config

Actualiza el script ubicado en rootfs/usr/local/sbin/odroid-config y amplia las funciones del sistema de archivos al cambiar "mmcblk0p2" por "mmcblk0p3".

```
   40 do_expand_rootfs() {
   42    p2_start=`fdisk -l /dev/mmcblk0 |
grep mmcblk0p3 | awk '{print $2}'`
   43   fdisk /dev/mmcblk0 <<EOF
   44 p
   45 d
   46 3
   47 n
   48 p
   49 3
   50 $p2_start

   72 case "$1" in
   73   start)
   74    log_daemon_msg "Starting resize2fs_
once" &&
   75     resize2fs /dev/mmcblk0p3 &&
   76     rm /etc/init.d/resize2fs_once &&
   77     update-rc.d resize2fs_once remove
&&
   78     log_end_msg $?
   79     ;;
   80   *)
```

Comprueba la versión de u-boot

Debes utilizar el último u-boot (12 de enero de 2014 o posterior), de lo contrario el comando *fdisk* no funcionará correctamente. Para ello, instala la versión 2.6 de Android en tu eMMC desde **`http://bit.ly/PXriWq`**. A continuación, compruebe la versión usando la consola USB-UART (el cable se vende por separado), deberías ver algo parecido a esto:

```
   OK

   U-Boot 2010.12-svn (Jan 27 2014 - 15:07:10)
for Exynox4412

   CPU: S5PC220 [Samsung SOC on SMP Platform
Base on ARM CortexA9]
   APLL = 1000MHz, MPLL = 880MHz
   DRAM:  2 GiB
```

Generar una nueva tabla de particiones

Usando la consola USB-UART, entra en u-boot y usa el comando *fdisk* para crear la tabla de particiones. Formato del comando: *fdisk -c* [boot device:0] [system] [userdata] [cache] [vfat]

```
   Exynos4412 #  fdisk -c 0 512 -1 128 300
   Count: 10000
   NAME: S5P_MSHC4
   fdisk is completed

   partion #     size(MB)      block start #
block count    partition_Id
      1              306            1462846
626934          0x0C
```

```
     2              517              134343
1059817          0x83
     3             6362             2089780
13031271         0x83
     4              131             1194160
268686           0x83
```

Item	Descripción	Sistema	Tamaño
Boot zImage ramdisk	BL1/BL2/U-boot/ENV Android Kernel Android Ramdisk(not used)	RAW	66MB
SYSTEM (mmcblk0p2)	Android System Gapps size is considered	EXT4	512MB
CACHE (mmcblk0p4)	Android Cache	EXT4	128M
BOOT (mmcblk0p1)	Linux Kernel boot.scr or boot.ini	FAT16	300M
USERDATA (mmcblk0p3)	Android Userdata Ubuntu rootfs	EXT4	All the rest

Esta tabla describe las diferentes partes de la tabla de particiones, y en cada parte se indicar el sistema operativo que se utiliza (Cómun, Android, Ubuntu).

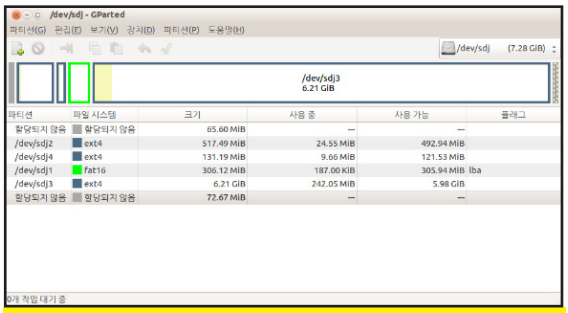
Formatear las particiones

Para preparar las particiones, apague ODROID y retire el módulo eMMC. Luego, conéctelo a cualquier PC Linux usando el adaptador que se incluye con eMMC y un adaptador de tarjeta SD USB. Tenga en cuenta que las particiones listadas en la tabla están por orden lógico y no por orden físico en el que aparecen en el disco.

```
  $ sudo fdisk -l
  Disk /dev/sdX: 7818 MB, 7818182656 bytes
  253 heads, 59 sectors/track, 1022 cylin-
ders, total 15269888 sectors
  Units = sectors of 1 * 512 = 512 bytes
  Sector size (logical/physical): 512 bytes
/ 512 bytes
  I/O size (minimum/optimal): 512 bytes /
512 bytes
  Disk identifier: 0x00000000

    Device Boot      Start         End
Blocks   Id  System
  /dev/sdX1            1462846      2089779
313467    c  W95 FAT32 (LBA)
  /dev/sdX2             134343      1194159
529908+  83  Linux
  /dev/sdX3            2089780     15121050
6515635+  83  Linux
  /dev/sdX4            1194160      1462845
134343   83  Linux
```

Luego, formatea el móculo eMMC para prepararlo para los sistemas de archivos raíz de Ubuntu y Android.

```
  [~]$ sudo mkfs.vfat /dev/sdX1
  mkfs.vfat 3.0.16 (01 Mar 2013)
  [~]$ sudo mkfs.ext4 /dev/sdX2
  mke2fs 1.42.8 (20-Jun-2013)
  Filesystem label=
  OS type: Linux
  Block size=4096 (log=2)
  ...

  [~]$ sudo mkfs.ext4 /dev/sdX3
  mke2fs 1.42.8 (20-Jun-2013)
  Filesystem label=
  OS type: Linux
  Block size=4096 (log=2)
  ...

  [~]$ sudo mkfs.ext4 /dev/sdX4
  mke2fs 1.42.8 (20-Jun-2013)
  Filesystem label=
  OS type: Linux
  Block size=1024 (log=0)
  ...
```



Verificar las particiones con GParted

Instalar rootfs de Ubuntu

```
  $ sudo cp -a boot/* /media/[user]/[mount_
point]/
  $ sudo cp -a rootfs/* /media/[user]/
[mount_point]/
```

[user]/[mount_point]] corresponde al directorio donde el adaptador eMMC está montado en el pc Linux. La partición de arranque corresponde a /dev/sdX1 (FAT) y la partición

rootfs corresponde a la partición dev/sdX3 / (EXT4 datos de usuario).

Reemplaza UUID en boot.scr

Revisa el archivo boot.scr en la partición "boot" y actualiza la partición /dev/sdX3 para que el UUID coincide con el que aparece en boot.scr:

```
  $ cat /media/codewalker/5145-2E60/boot.
scr

                              '^E^YVOÚ<9f>7R}-
>^@^@^A<^@^@^@^@^@^@^@^E^?ß9^E^
B^F^@boot.scr for X with HDMI auto-
pr^@^@^A4^@^@^@^@setenv initrd_high
"0xffffffff"
  setenv fdt_high "0xffffffff"
  setenv bootcmd "fatload mmc 0:1 0x40008000
zImage; fatload mmc 0:1 0x42000000 uInitrd;
bootm 0x40008000 0x42000000"
  setenv bootargs "console=tty1
console=ttySAC1,115200n8
root=UUID=e139ce78-9841-40fe-8823-
96a304a09859 rootwait ro mem=2047M"
   boot

  $ sudo tune2fs /dev/sdX3 -U e139ce78-
9841-40fe-8823-96a304a09859
  tune2fs 1.42.8 (20-Jun-2013)
```

Instalar los archivos del sistema Android

Puesto que la imagen original ya contiene los archivos de Android, las particiones de Android se pueden habilitar introduciendo en u-boot los siguientes comandos fastboot:

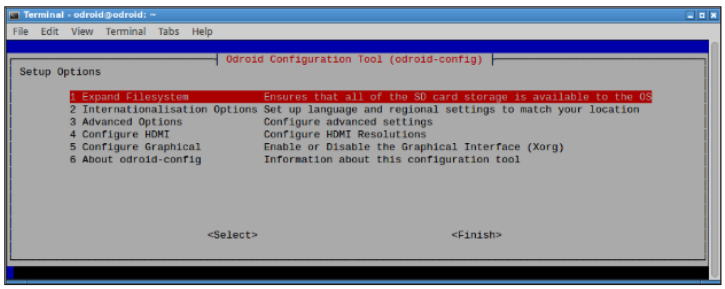
```
# fastboot
$ fastboot flash system system.img
$ fastboot reboot
```

Ahora que tanto Ubuntu como Android están instalados, el módulo eMMC está listo para usarse.

Usando la imagen pre-compilada

Si no eres desarrollador de plataformas Android o deseas utilizar el doble sistema de arranque de inmediato, te ofrecemos una imagen pre-compilada, que se puede descargar desde **http://bit.ly/1i6bSWQ** y pásala a tu eMMC.

La imagen pre-compilada utiliza Xubuntu como sistema operativo por defecto. Amplia el sistema de archivos y reinicia el sistema operativo en el primer arranque. La partición de archivos raíz ampliada de Ubuntu será accesible desde Android como una partición de datos.



Herramienta de configuración del ODROID. Un desarrollo, instalación y ejecución limpia.

Cambiar entre Android y Ubuntu

Cuando deseas cambiar la opción de arranque por defecto a Android, abre una ventana de terminal en Xubuntu y ejecutar el script bootA.sh utilizando permisos de superusuario tecleando *sudo/media/boot/bootA.sh*. Si has creado tu propio sistema de arranque, deberías agregar el script bootA.sh manualmente:

```
#!/bin/sh
cd /media/boot
mv boot.ini.android boot.ini
mv boot.scr boot.scr.ubuntu
sync
reboot
```

Si desea cambiar la opción de arranque por defecto de nuevo a Ubuntu desde Android, abra la aplicación Terminal Android y ejecutar el script bootL.sh, que también requerirá permisos de superusuario:

```
# su
# sh /storage/sdcard1/bootL.sh
```

Para los sistemas creados manualmente, el script bootL.sh deberá contener lo siguiente para volver a Xubuntu:

```
#!/bin/sh
cd /storage/sdcard1
mv boot.scr.ubuntu boot.scr
mv boot.ini boot.ini.android
sync
reboot
```

Para ver una demostración del doble sistema de arranque, por favor echa un vistazo a nuestro video formativo en **http://youtu.be/osERBvQN5ME**.

PRIMEROS PASOS CON TU ODROID

CÓMO COPIAR UN ARCHIVO DE IMAGEN A UNA TARJETA SD O EMMC

Muchas de las imágenes oficiales pre-compiladas de la comunidad están disponibles para su descargar en los foros ODROID, http://forum.odroid.com, pero a veces resulta difícil para los nuevos poseedores de un ODROID aprender a usar estas imágenes para crear un disco de arranque. En este artículo se describe el proceso de descarga, verificación e instalación de un archivo .img o .img.xz usando Linux, Mac OSX o Windows.

Requisitos Generales

1. Cualquier equipo ODROID, con el adecuado adaptador de alimentación.
2. Tarjeta MicroSD Clase 10+ (con lector de tarjeta SD) o un eMMC de 8 GB+
3. Un archivo de imagen descargado cuyo nombre termina en .img o .img.xz

Imagen y archivo de verificacion

Para descargar la imagen, primero crea una carpeta en la que colocarás la imagen en un ordenador con Linux, OSX o Windows. Si vas a usar una imagen oficial de Ubuntu pre-compilada de Hardkernel puedes descargar el archivo comprimido .img.xz desde **http://bit.ly/1iPCvzf**. Cualquier imagen para U2 también funciona en el U3 (y viceversa). Para garantizar la integridad de los archivos asegúrate de descargar el archivo de verificación checksum correspondiente (.xz.md5sum) desde el mismo lugar. Los archivos descargados y usados en este artículo son:

```
xubuntu-13.10-desktop-armhf_odroidu_20140211.img.xz"
"xubuntu-13.10-desktop-armhf_odroidu_20140211.img.
xz.md5sum
```

Si tiene la intención de utilizar una imagen de otra parte, ten en cuenta que necesitas asegurar la autenticidad de la imagen y que es segura para su posterior uso. Descarga los archivos de imagen sólo de una fuente fiable como los foros ODROID, repositorios de la comunidad o el sitio de Hardkernel.

Linux

El procedimiento indicado aquí usa el comando Linux *disk-duplicate* (*dd*). Al igual que numerosos comandos de Linux, se debe utilizar con cuidado. Sin darte cuenta puedes inutilizar el sistema Linux, ya que las particiones vitales de disco puedes ser sobrescritas. Algunos de los parámetros de los comandos manejados pueden necesitar modificarse para utilizar información específica en tu instalación.

En una ventana de terminal, navega hasta la carpeta donde has descargado la imagen con el comando *cd*. Luego, utiliza md5sum en la imagen descargada, escribiendo:

```
md5sum xubuntu-13.10-desktop-armhf_odroidu_20140211.
img.xz
```

Compara el resultado con el contenido del archivo md5sum descargado del servidor. En este caso particular, la coincidencia entre md5sum y la imagen era:

```
605ac6805feb2867d78c45dd660acc80
```

Si coinciden la integridad de los archivos está garantizada y se puede continuar con el siguiente paso. Si no, el archivo de imagen puede estar dañado y se debe volver a descargar. Un desajuste en el md5sum indica una imagen alterada o corrupta, especialmente cuando la autenticidad de la página web de descargas es cuestionable.

Una vez que md5sum ha sido verificado, descomprima la imagen comprimida utilizando el comando *xz*:

```
xz -d xubuntu-13.10-desktop-armhf_odroidu_20140211.
img.xz
```

Esto reemplazará el archivo comprimido por un archivo de imagen con extensión .img. El siguiente paso es determinar qué etiqueta ha asignado el PC Linux a la SD o módulo eMMC para escribir la imagen. En la ventana de terminal ya ejecutada y sin necesidad de insertar la tarjeta SD ejecute el comando *df-h*. Anota el resultado que refleja las diversas unidades montadas. El resultado puede ser algo similar a esto:

```
$ df -h
Filesystem    Size   Used    Avail  Use%  Mounted on
/dev/sda1     46G    3.4G    40G    8%    /
none          4.0K   0       4.0K   0%    /sys/fs/cgroup
udev          2.0G   4.0K    2.0G   1%    /dev
tmpfs         396M   880K    395M   1%    /run
none          5.0M   0       5.0M   0%    /run/lock
none          2.0G   152K    2.0G   1%    /run/shm
none          100M   76K     100M   1%    /run/user
```

En este caso, /dev/sda1 refleja el sistema de ficheros correspondiente a la primera partición del primer dispositivo de almacenamiento, que en este ejemplo es de 50 GB.

Ahora, inserte la tarjeta SD y ejecute el mismo comando:

```
$ df -h
Filesystem  Size  Used  Avail Use% Mounted on
/dev/sda1   46G   3.4G  40G   8%   /
none        4.0K  0     4.0K  0%   /sys/fs/cgroup
udev        2.0G  4.0K  2.0G  1%   /dev
tmpfs       396M  880K  395M  1%   /run
none        5.0M  0     5.0M  0%   /run/lock
none        2.0G  152K  2.0G  1%   /run/shm
none        100M  76K   100M  1%   /run/user
/dev/sdb1   15G   32K   15G   1%   /media/terrapin/XFER
```

En este caso, /dev/sdb1 es la única nueva entrada, refleja que la tarjeta SD ha sido insertada y que la primera partición se ha montado. Aunque el número de partición se añade al nombre, el nombre del dispositivo de la tarjeta SD en realidad es /dev/sdb (sin el número 1). Si se utiliza una eMMC, el sistema puede asignar una etiqueta similar a /dev/mmcblk0 en su lugar, y la primera partición será montada como /dev/mmcblk0p1.

Si tu tarjeta SD es nueva y sin formatear, puedes saltarte el siguiente paso, en el cual se limpia la tarjeta SD escribiendo ceros en la misma. Esto asegura que no se queden datos previos en el disco, pudiendo alterar el nuevo esquema de particiones, así como hacer cualquier copia de backup más grande de lo necesario cuando se comprima.

En este ejemplo, hemos visto que la tarjeta SD contiene una partición con formato. Tiene que ser desmontada en primer lugar, con el comando *umount*.

```
sudo umount /dev/sdb1
```

Luego, pon a cero la partición anterior con el comando:

```
sudo dd if=/dev/zero bs=4M of=/dev/sdb
&& sync
```

Merece la pena reiterar que, puesto que se están borrando las particiones y formateando la tarjeta SD, se debe extremar la precaución. Si tienes dudas sobre cómo usar estos potentes comandos, puede que sea más seguro crear una máquina virtual de Linux (VM) usando VirtualBox de Oracle (**https://www.virtualbox.org/**) y luego ejecutar los comandos dentro de la máquina virtual. En el peor de los casos, la máquina virtual se estropeará, dejando intacto el sistema Linux real.

Espera que se complete el proceso de formateo antes de pasar al siguiente paso, que puede durar desde 15 minutos a 2 horas, dependiendo de la velocidad de la tarjeta SD o el módulo eMMC. Una vez finalizado, la ventana de Terminal informará que la tarjeta está sin espacio en disco (que es normal), lo que indica que los ceros se han escrito correctamente.

Desde la carpeta que tiene la imagen extraída, escribe la imagen en la tarjeta SD formateada utilizando el nombre del dispositivo de la siguiente forma:

```
sudo dd bs=4m if=xubuntu-13.10-desktop-
armhf_odroidu_20140211.img \
of=/dev/disk2
```

El nombre del dispositivo debe especificarse cuidadosamente en este comando como ya se ha señalado, dejando cualquier numero entero que corresponda a las particiones individuales en lugar de todo el disco.

Este proceso tardará un tiempo (incluso 2 horas) hasta completarse. Si todo sale bien, el resultado incluirá el número de registros (entrada y salida), bytes copiados, velocidad y duración de la copia de datos. El comando *sync* vacía la caché de escritura, lo que garantiza que la imagen ha sido completamente grabada en el disco.

En caso de fallo, siga las indicaciones si las hubiese. Puede valer la pena volver a formatear la tarjeta SD y repetir el proceso. Si vuelve a fallar, es preferible usar otra tarjeta SD de capacidad similar.

Al finalizar el comando *dd* con un resultado satisfactorio, la SD se volverá a montar automáticamente. Vuelva a ejecutar el comando *df* para asegurarte que la SD se ha montado correctamente y luego extraiga la tarjeta con el comando:

```
sudo eject /dev/sdb
```

La imagen ya está lista para arrancar. Si ODROID se está ejecutando, apáguelo y luego inserte la tarjeta SD y vuelva a encenderlo. Ahora debería arrancar usando la nueva imagen con el sistema operativo y está lista para que la disfrutes.

OSX

Además de los requisitos generales mencionados anteriormente, OSX también debe tener instalado Unarchiver que te prermitirá comprimir y descomprimir de archivos de imagen, disponible en **http://bit.ly/1iLr5m3**. Ten en cuenta que

CONSIGUE
UN POCO MÁS DE PERSONALIDAD EN TU SUDO

por Bruno Doiche

A nadie le gusta ser insultado, por supuesto, pero a veces tu Linux parece una máquina sin alma cuando se introduce un sudo su - y por error una contraseña incorrecta:

```
odroid@goonix:~$ sudo su -
[sudo] password for odroid:
Sorry, try again.
```

Qué aburrido es eso, ¿no? Nada que no se pueda arreglar introduciendo:

```
sudo visudo
```

Agregue la siguiente línea:

```
Defaults        insults
```

Y ahora, cuando hagas sudo con un comando y ponga una contraseña incorrecta verás algo como esto:

```
odroid@goonix:~$ sudo su -
[sudo] password for odroid:
Hold it up to the light --- not
a brain in sight!
[sudo] password for odroid:
You can`t get the wood, you
know.
[sudo] password for odroid:
There must be cure for it!
```

Es como tener tu propio editor gráfico viviendo en tu terminal!

OTRO TRUCO
DE SEGURIDAD SUDO

Una buena práctica de seguridad es no fijar tu usuario sudo de forma automática en una máquina a la que otra persona pueda acceder. Y después de salir de sudo, no pedir inmediatamente tu contraseña. Para tu tranquilidad escriba el siguiente comando:

```
sudo -K
```

Unarchiver tiene varias versiones específicas para Macintosh, así que asegúrate de descargar e instalar la versión OSX apropiada para su sistema.

El procedimiento para comprobar el md5sum del archivo descargado es similar a Linux, pero usa el comando *md5* en lugar de *md5sum*. Un buen atajo para realizar la suma de verificación checksum es abrir una ventana de terminal y escribir *md5* seguido de un carácter [SPACE]. Luego, usando el ratón arrastre el archivo de imagen comprimido (*.img.xz) a la ventana de terminal. La línea de comandos se actualizará con el nombre del archivo comprimido. Ahora pulse la tecla [ENTER]. El md5sum del archivo de imagen comprimido es devuelto como resultado. Compare el resultado con el contenido del archivo md5sum para asegurarte que el archivo se ha descargado correctamente. Para más información sobre cómo comprobar md5sum, consulta la página de ayuda de md5sum para OSX en **http://bit.ly/1nTVz7q**.

Suponngamos que la utilidad Unarchiver en su versión 3.9.1 (la última a raíz de este artículo), ha sido instalada en tu Mac y se ha fijado como utilidad por defecto para descomprimir archivos. Inicie el programa y configuralo para:

1. Conservar el archivo original descargado (después de descomprimirse).
2. Colocar la imagen sin comprimir en la misma carpeta en la que se encuentra el archivo comprimido.
3. Guardar la fecha de modificación del archivo comprimido (mantener un registro de información de la imagen).

Descomprimir el archivo con estas opciones debe dar lugar a la creación de un archivo con extensión .img en la misma carpeta que el archivo original img.xz.

Aunque *df-h* también se puede utilizar para comprobar las unidades montadas disponibles, OSX cuenta con un comando personalizado denominado *diskutil* que puede ser utilizado para tal fin proporcionando un resultado más simple. En la ventana de terminal, escriba el siguiente comando antes de insertar la tarjeta SD:

```
diskutil list
```

Observa que, en OSX las unidades montadas son nombradas como /dev/diskX, en lugar de /dev/sdX del sistema de Linux. Si la tarjeta SD es nueva, omita el siguiente paso ya que no es necesario poner a cero una tarjeta nueva.

Para preparar la SD o el módulo eMMC, inicie la aplicación OSX Disk Utility y haga clic en la tarjeta SD de destino en la parte izquierda de la ventana. Pulse el botón "Security Options " en la parte inferior central y seleccione la opción "Zero Out Data" en la ventana emergente. Pulse OK y a continuación, haga clic en el botón "Erase" y espere hasta que la barra de progreso llegue al 100%. Una vez que el disco se ha puesto a cero está listo para aceptar la nueva imagen. .

Debido al auto-montaje en OSX de cualquier medio, la unidad debe estar primero sin montar usando el comando:

```
sudo diskutil unmountdisk /dev/disk2
```

Luego, escriba la imagen en la tarjeta SD usando el nombre de dispositivo con el comando *dd*. Tenga en cuenta la minúscula "1m" diferente a la sintaxis de mayúsculas en Linux:

```
sudo dd bs=1m if=odroidu2_20130125-linaro-ubuntu-desk-
top_SDeMMC.img of=/dev/disk2
```

El nombre de dispositivo de disco debe especificarse cuidadosamente en este comando como se ha señalado anteriormente. Espere a que finalice el comando.

Una vez que el comando *dd* haya finalizado correctamente, la tarjeta SD volverá a ser automáticamente montada. Expulsa la tarjeta mediante el siguiente comando:

```
sudo diskutil unmountdisk /dev/disk2
```

Espera hasta que el icono de disco desaparezca del escritorio, retire la tarjeta SD o módulo eMMC del Macintosh, insértela en el ODROID y enciéndalo para comenzar a utilizar el nuevo sistema operativo.

Windows

Windows no soporta de forma nativa las partición de archivos EXT3/EXT4 de Linux, asi que son necesarias varias utilidades para copiar una imagen al disco:

1. 7-Zip (`http://www.7-zip.org/`) utilidad de compresión de archivos para extraer la imagen de la tarjeta SD desde el archivo descargado .xz.
2. Win32DiskImager mejorado (`http://bit.ly/1q1HTsW`) utilidad para escribir el fichero .img en tu tarjeta SD.
3. MD5sums (`http://bit.ly/1ukeVUZ`) utilidad para evaluar la suma de verificación (integridad) de un archivo descargado. Esto es opcional, pero útil para asegurarte que la imagen coincide con la versión del servidor.

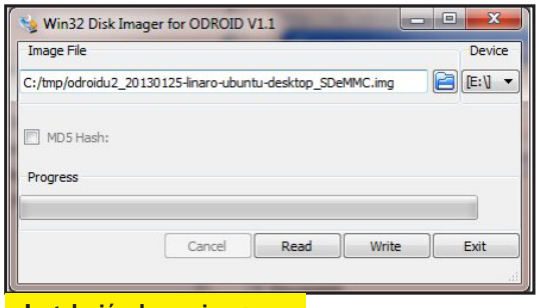
Después de descargar el archivo .img.xz como se ha descrito anteriormente, comprueba md5sum usando el comando:

```
c:\Program Files (x86)\md5sums-1.2\md5sums xubuntu-13-
.10-desktop-armhf_odroidu_20140211.img.xz
```

Compara el resultado con el contenido del archivo md5sum y si coinciden continua con el siguiente paso. Una vez que el archivo ha sido comprobado y está en perfecto estado, utiliza la utilidad 7-Zip para extraer la imagen del archivo comprimido:

```
c:\Program Files (x86)\7-zip-7z920 -z xubuntu-13.10-
desktop-armhf_odroidu_20140211.img.xz
```

Para comodidad de los usuarios de Windows, Hardkernel publica una utilidad especifica Win32DiskImager, que escribe automáticamente ceros a la tarjeta SD antes de copiar la imagen, de modo que todo se puede hacer en un único paso. Su interfaz es similar a la mostrada en la captura de pantalla de la izquierda.



Instalación de una imagen usando WinDiskImager

Selecciona la imagen y el dispositivo de destino, e inicia la instalación de la imagen haciendo clic en el botón "Write". El tiempo extra que se necesita para escribir ceros suele añadir unos 30 minutos o más al proceso de escritura.

Por último, expulse el disco haciendo clic derecho sobre la tarjeta SD con el Explorador de archivos y seleccionando la opción "Expulsar". Inserte la tarjeta SD en el ODROID, enciéndalo, espera a que el proceso de arranque se complete y disfrutar de tu nuevo sistema operativo.

Para obtener más información sobre cómo copiar archivos de imágenes en una tarjeta SD, consulte la wiki ODROID de Osterluk en **`http://bit.ly/1rQgqWH`**.

LOCALIZA LOS ARCHIVOS MÁS GRANDES DE UN DIRECTORIO

por Bruno Doiche

Quieres saber de un directorio específico cuales son los archivos más grandes, ejecute el siguiente comando:

```
find . -type f -exec ls -s {} \;
| sort -n -r | head -5
```

Esto es útil cuando necesitas hacer un poco de limpieza, ¿Necesitas conocer los archivos de más de un determinado tamaño? Entonces, escriba este comando para archivos mayores de 100 MB:

```
find ~ -size +100M
```

COMO DIVIDIR UN GRAN ARCHIVO

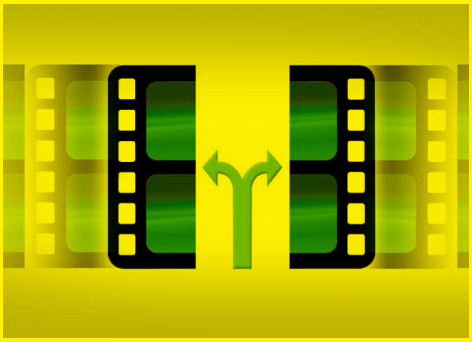
Por fin tiene almacenado ese maravilloso espectáculo grabado en HD para tu amigo, pero te das cuenta que ocupa 7GB y lo único que tienes es dos unidades de memoria flash de 4 GB ¡No temas!

```
split -b 1GB [yourvideofile.
mkv] [yoursplitvideofile]
```

Se divide en trozos de 1 GB que se pueden copiar a las unidades flash. Cuando estés en casa, copiarlos al disco duro y ejecutar el comando *cat*:

```
cat [yoursplitvideofile*] >
[yourvideofile.mkv]
```

Y ahí lo tienes!



SOBRE EL COMPORTAMIENTO TERMICO DE ODROIDS

LA DIFERENCIA DE RENDIMIENTO ENTRE EL XU Y EL U3 CON MAYOR DETALLE

por Jussi Opas

Dos ODROIDs, el XU y el U3, se han probado al mismo tiempo con el fin de evaluar su diferencia de rendimiento, temperatura y comportamiento al ajustar su frecuencia. Podemos presuponer con toda seguridad que el XU es más rápido que el U3, pero la pregunta es: ¿cuánto más rápido? Para que podamos tener una respuesta con fundamento y no sólo basada en la intuición, hemos realizado algunas mediciones con las dos máquinas. Para estas pruebas, la placa XU cuenta con un disipador de calor y ventilador conectados, mientras que el U3 incorpora un disipador de calor sin ventilador.

Los dos equipos tienen especificaciones muy distintas, tal y como se muestra en la tabla de abajo. El U3 tiene un procesador QuadCoreARM Cortex A9, mientras que el XU tiene un procesador big.LITTLE con dos clusters separados: uno con cuatro núcleos A7 ARM cortex y otro con cuatro núcleos A15. Ambas placas cuentan con 2GB de memoria PoP, pero el tipo de memoria incluida en el XU es más rápida que la que viene en el U3. Al ejecutar las distribuciones de Ubuntu oficiales de Hardkernel, el regulador de ajuste de frecuencia en el U3 está definido en "performance" por defecto, mientras que el XU usa la configuración "ondemand". La frecuencia de reloj de fábrica del U3 a 1,7 GHz es 100 MHz más alta que la frecuencia del XU a 1,6 GHz. La capacidad para overclock de cada placa puede variar y sus valores están indicados en la tabla anterior. El U3 fue probado con el Kernel 3.8 de Linux y el XU fu testeado con un Kernel 3.4 reajustado.

Cada ODROID potencialmente se comporta de forma distinta cuando los procesadores son usados al máximo por una potente aplicación. Ambos SoCs también incluyen una GPU, pero su comportamiento no nos es relevante, puesto que nuestra aplicación de pruebas no realiza cálculos gráficos.

	ODROID U3	ODROID XU
SoC	Exynos 4412 Prime	Exynos5 5410
CPU	ARM 4xA9	ARM 4xA15 and 4xA7
Memory	2 GB, LPDDR2	2 GB, LPDDR3
Default governor	performance	ondemand
Default max frequency	1.7 GHz	1.6 GHz
Overclockability	1.92 GHz	1.8 GHz or more
Cooling	Heat sink	Heat sink with embedded fan
Kernel	3.8.13.18	3.4.74 (customized)

Sabemos qué haces esta comparación una docena de veces al día cuando estás pensando en comprar uno, de modo que te hemos preparado esta pequeña tabla.

-Todos los cálculos se hacen por CPU y la RAM no es un fator limitador.
-Los cálculos se hacen con múltiples hilos de ejecución usando Java con una aplicación real, que no ha sido escrita para la pruebas de rendimiento en cuestión.
-La misma prueba se ha realizado con diferentes números de hilos de ejecución, de modo que los núcleos han sido probados al 100%.
-No se han usado dispositivos E/S.
-Los cálculos se componen de números enteros, duplicados, restas, divisiones, multiplicaciones, raíces cuadradas, algunos procesos matemáticos de Java, accesos y asignaciones, y la creación y eliminación de objetos.

Esperamos que la aplicación utilizada en las pruebas no presente fallos internos y podamos confiar en los resultados de mi anterior artículo [OP14].

La aplicación de prueba, tal y como se ha utilizado lleva al equipo al borde de su capacidades, lo cual rara vez ocurre cuando tiene un uso diario normal. Los ensayos se realizaron a temperatura ambiente, aproximadamente 22ºC. La comparación se realizo con las frecuencias por defecto (1,6 y 1,7 GHz).

En base los resultados, podemos concluir que el XU es aproximadamente un 25% más rápido que el U3. Sin embar-

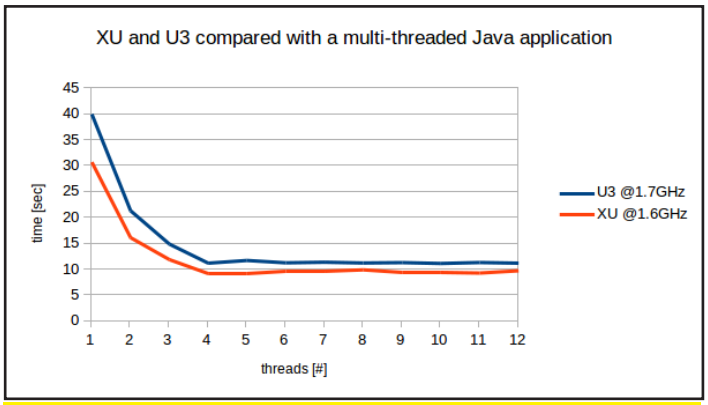


Figura 1 - XU y U3 comparados con una aplicación Java con múltiples hilos de ejecución.

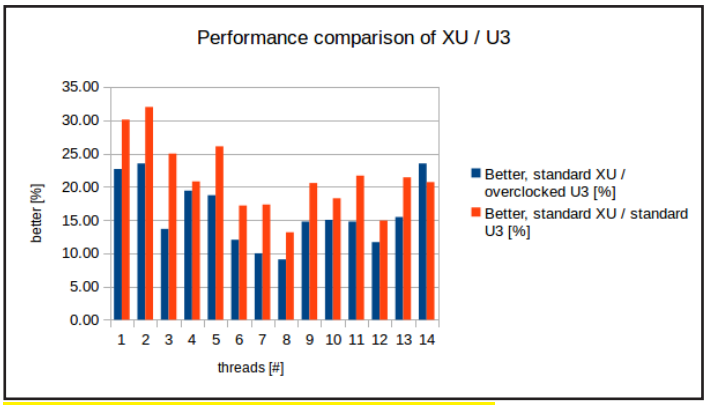


Figura 2 - Comparativa de rendimiento del XU/U3

go, al U3 se le puede aplicar fácilmente overclock añadiendo la siguiente línea al archivo /etc/rc.local:

```
echo 1920000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_max_freq
```

También realizamos una prueba similar con overclock a 1.92 GHz y observamos cuánto más rápido es XU al comparase con el U3, tal y como se observa en la Figura 2.

Ahora podemos concluir que con U3 en overclock, XU es un 15 - 20% más rápido. Sin embargo, aún no hemos terminado. Si repetimos la prueba y elaboramos una gráfica con varios ensayos con el U3 en overclock, obtenemos la curva plana que se muestra en la Figura 3.

El rendimiento disminuye cuando las pruebas se repiten de forma continuada (sin enfriamiento) a una frecuencia con overclock. El rendimiento es constante con los tres primeros núcleos cargados, pero luego disminuye cuando usamos el cuarto núcleo al máximo. El XU en cambio ofrece un rendimiento similar en todos los ensayos realizados con su frecuencia de 1,6 GHz por defecto. Por lo tanto, debemos también considerar el comportamiento térmico y la escala de frecuencia de cada plataforma. La frecuencia utilizada puede ser fácilmente comprobada manualmente con el comando:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

La temperatura de los núcleos en XU se almacena en un archivo en Linux y se puede mostrar de la siguiente forma:

```
cat /sys/devices/platform/exynos5-tmu/temp
```

La temperatura del U3 se puede ver consultando un archivo similar:

```
cat /sys/class/thermal/thermal_zone0/temp
```

El valor 50000 significa una temperatura de 50 grados. Incluimos la Lectura de la temperatura y la frecuencia de reloj actual en nuestra aplicación de pruebas, de modo que hemos sido capaces de recoger también los datos de frecuencia de reloj y térmicos correspondientes tras la ejecución de cada sub-proceso con distintas configuraciones de hilos de ejecución. La figura 4 incluye un gráfico con las frecuencias de reloj y temperatura superpuestas.

Cuando la temperatura del chip aumenta la frecuencia de reloj disminuye, pero aumenta de nuevo a la máxima frecuencia cuando la temperatura baja. Por lo tanto, la placa U3 mantiene la temperatura a un nivel de alrededor de 80 °C. Este comportamiento es muy constante, y evita el sobrecalentamiento de manera muy eficaz manteniendo la placa estabilizada. Por lo tanto, el regulador en "performance" es la configuración por defecto más adecuada para gestionar la frecuencia de reloj del U3, y no necesitas pararte a probar el regulador en "ondemand" con el U3.

Figura 3 - Prueba del U3 repetida y ejecutada a una frecuencia de 1,92 GHz (overclock)

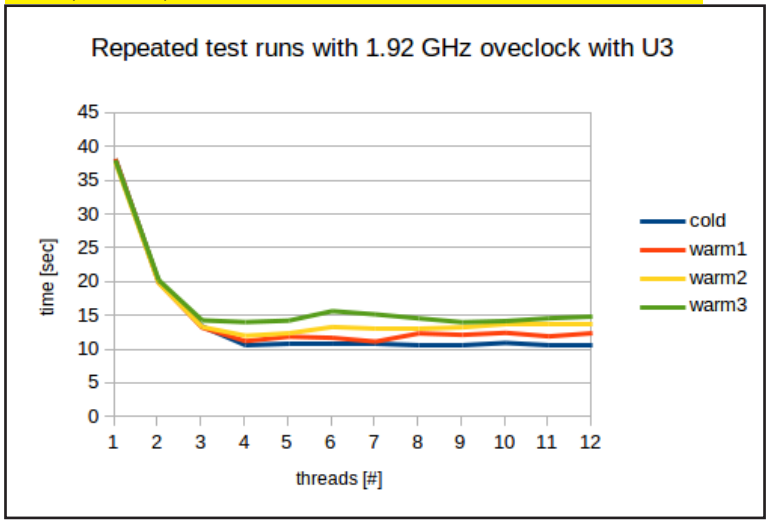
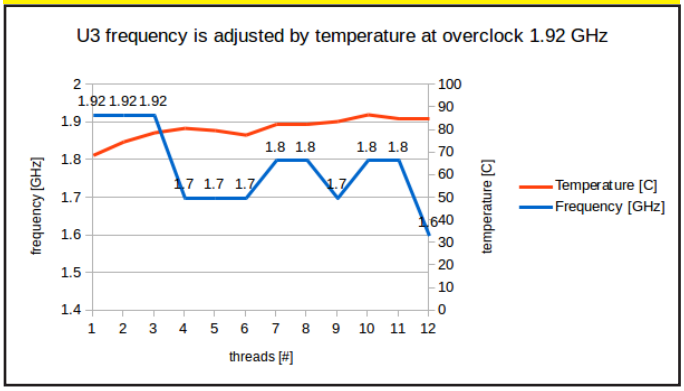


Figura 4 – Frecuencia del U3 ajustada por temperatura a 1.92GHz con overclock.



Para realizar una comparación justa entre los procesadores A9 y A15, ambas placas se ejecutaron a 1.7 GHz. El XU muestra un mejor rendimiento con todas las configuraciones de ejecución desde uno hasta 12 hijos de ejecución. La temperatura de U3 aumenta en toda la ejecución de la prueba, partiendo de 60 grados hasta llegar a los 78 grados. Sin embargo, la temperatura de XU aumenta más rápido que el U3, desde los 62 grados hasta 88 grados, lo que demuestra que XU funciona a más temperatura que el U3 con la misma frecuencia. La temperatura del XU disminuye más rápidamente tras finalizar la prueba. Esto se debe al ventilador que tiene incorporado que se mantiene funcionando hasta que la temperatura baje por debajo de los 55 grados. Puesto que el U3 sólo tiene un disipador de calor sin ventilador, la temperatura se mantiene más alta durante más tiempo tras haber finalizado la prueba.

El XU se puede configurar para que utilice sólo los núcleos PEQUEÑOS de dos formas posibles: 1) configurando el intercambio del clúster, o 2) limitando la escala de frecuencia máxima a un máximo de 600 MHz [ME13]. En ambos casos, se debe utilizar el regulador de frecuencia en "ondemand".

El primer método se hace introduciendo los siguientes comandos como root:

```
  echo ondemand > /sys/devices/system/cpu/cpu0/cpufreq/
scaling_governor
  # only LITTLE
  echo 01 > /dev/b.L_operator
  # it is better to wait a while and inspect that the only
A7 cores are in use
  cat /dev/bL_status
  # the output will be as follows
        0 1 2 3 L2 CCI
  [A15]  0 0 0 0  0
  [A7]   0 1 0 0  1
  # to disable cluster switching
  echo 00 > /dev/b.L_operator
```

Cuando queramos volver activar los núcleos A15 y el intercambio del clúster podemos escribir lo siguiente:

```
echo 11 > /dev/b.L_operator
```

El segundo método para aislar el clúster A7 consiste en modificar la frecuencia del procesador como root:

```
  echo ondemand > /sys/devices/system/cpu/cpu0/cpufreq/
scaling_governor
  echo 600000 > /sys/devices/system/cpu/cpu0/cpufreq/scal-
ing_max_freq
```

La baja frecuencia de la placa deben entenderse como duplicada, por lo que el valor usado realmente es de 1,2 GHz, aunque aparece 600 MHz. Vea, por ejemplo, el resultado del siguiente comando *cat*:

```
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
```

El valor 600.000 significa que la frecuencia actual es de 2 x 600 MHz = 1,2 GHz. Realizamos una prueba con los clúster A7 y A15 ejecutándolos a 1,2 GHz para ver la diferencia en el rendimiento y la temperatura puramente basada en la arquitectura. La figura 5 muestra el rendimiento y la Temperatura del clúster PEQUEÑO. El rendimiento mejora tal y como se esperaba cuando se agregan más hilos de ejecución para hacer el mismo trabajo. Al mismo tiempo, la temperatura se mantiene estable a 52-54 grados y el ventilador no se activo durante la prueba.

Con el regulador en "ondemand" la frecuencia es estable a su máximo inicial de 1,2 GHz. El comportamiento del cluster A15 se muestra en la parte derecha de la figura 5 y su rendimiento es alrededor de un 40-50% mejor que el clúster A7. La temperatura empieza a subir desde el inicio de la prueba hasta que se añaden 4 o más hilos de ejecución y los 4 núcleos se activan. En este punto, el ventilador empieza a girar y la temperatura se mantiene estable a unos 63 grados. La frecuencia de reloj permanece a 1,2 GHz, tanto en la opción de "ondemand" como "performance".

La placa XU es completamente silenciosa cuando sólo se usa el clúster A7 y continúa siendo silenciosa si se utilizan los núcleos A15, puesto que el ventilador gira muy lentamente. Obviamente, al XU se le puede aplicar overclock con frecuencias más altas como 1.4, 1.6 y hasta 1,7 GHz. La figura 6 muestra los resultados de las pruebas con 1,4 y 1,7 GHz.

La parte izquierda de la figura 6 muestra la temperatura y rendimiento usando 1,4 GHz, y la parte derecha muestra las mismas pruebas a 1,7 GHz. A 1,4 GHz, la temperatura se mantiene por debajo de 70 ºC y la frecuencia se mantiene constante a 1,4 GHz. El rendimiento a 1,7 GHz muestra que es más rápida, pero la temperatura es mayor. Durante la prueba con 4 hilos de ejecución el ventilador empieza a girar más rápido al mismo tiempo que aumenta la temperatura, por eso la temperatura desciende al trabajar con 5 hilos de ejecución.

Hemos preparado dos gráficos diferentes con el fin de comparar el regula-

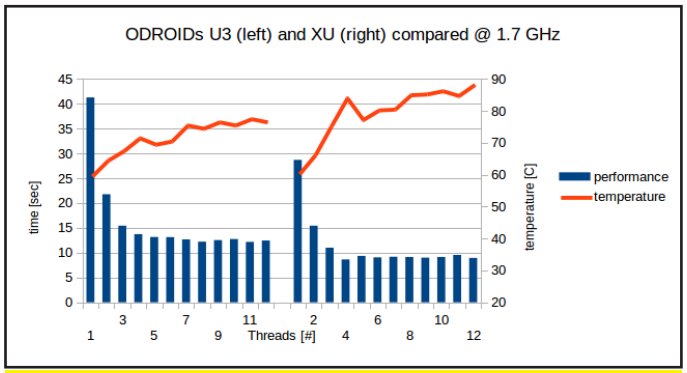


Figura 5 - Clusters XU big.LITTLE @ 1.2GHz, 4xA7 (izq.), 4xA15 (der.)

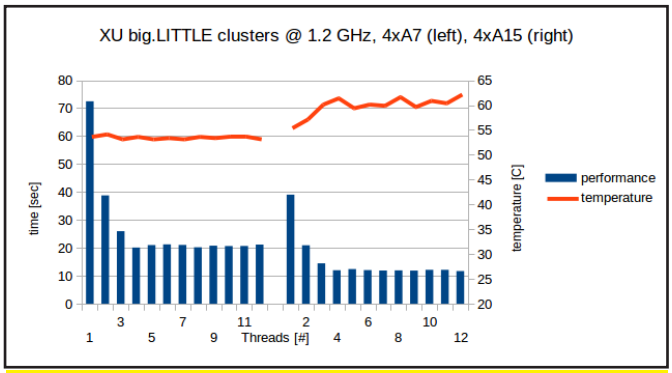


Figura 6 - A15 @ 1.4GHz (izquierda) y @ 1.7 GHz (derecha)

dor "ondemand" frente a una alta frecuencia constante en XU. La primera gráfica muestra la relación de la temperatura y el rendimiento, y la segunda muestra la frecuencia y la temperatura juntas. Con una frecuencia constante de 1,7 GHz, la temperatura aumenta hasta los 90 ºC con 12 hilos de ejecución. El rendimiento es estable cuando los 4 núcleos se utilizan con pruebas que usan 4-12 hilos. Cuando el regulador "OnDemand" se usa con una frecuencia máxima de 1,8 GHz, la temperatura aumenta más despacio. Sin embargo, el rendimiento tiene cierta degradación cuando se utilizan 6 y 9 hilos de ejecución.

La figura 8 muestra la relación entre la temperatura y la frecuencia utilizando el regulador "OnDemand" frente al uso de una frecuencia constante de 1,7 GHz.

En el gráfico de la izquierda, el regulador "ondemand" mantiene la temperatura baja con frecuencias variables. La frecuencia más alta utilizada ha sido 1.8, registrada desde un archivo justo después de la sub-ejecución de un hilo. Después, han sido usadas diferentes frecuencias; 1.6, 1.3 y 1.2 GHz. El gráfico de la derecha muestra que la frecuencia es constante a 1,7 GHz. La temperatura alcanza su máximo en toda la prueba. Además, sabemos que los PEQUEÑOS núcleos A7 no se usan cuando el regulador de rendimiento está activado o cuando es asignada como frecuencia máxima una alta frecuencia constante (> 1,2 GHz). La recomendación, en base a estos resultados, es usar regulador "ondemand" del ODROID-XU para un ajuste óptimo. La frecuencia constante puede ser asignada o usar el regulador "performance" cuando sea necesario, por ejemplo, para estudiar el comportamiento de una aplicación en desarrollo.

Conclusiones

Ambos equipos toleraron muy bien las diferentes configuraciones aplicadas. Con ODROID es seguro experimentar aplicando overclock, ya que tiene protección térmica contra sobrecalentamientos.

- El procesador del XU se calienta más que el procesador del U3 a idénticas frecuencias.

- En base a estas pruebas, podemos concluir que el XU es un 25-30% más rápido que el U3. Sin embargo, la E/S de archivos y el rendimiento de la GPU no se han tenido en cuenta.
- Si quiere tener un ordenador completamente silencioso, la mejor opción es un U3 estándar sin ventilador.
- Si se necesitas más potencia, la XU es tu opción. Para hacer frente a cualquier problema de ruido del ventilador, el XU se puede configurar para que el ventilador esté siempre apagado o que gire a una baja velocidad.
- Con el XU, dispones de dos equipos diferentes en una sola placa.
- Mejorando la refrigeración, incluso a altas frecuencias con overclock, se puede lograr un rendimiento aún mejor.

El comportamiento de la frecuencia del U3 con el regulador "performance" es fácil de entender y es perfecto. El comportamiento de frecuencias del XU es más complicado y no ha sido totalmente tratado en este artículo. Por ejemplo, no emos descritp cuándo y por qué un núcleo se enciende o se apaga durante la ejecución.

Figura 7 - ODROID-XU temperatura vs rendimeinto usando el regulador en "ondemand"

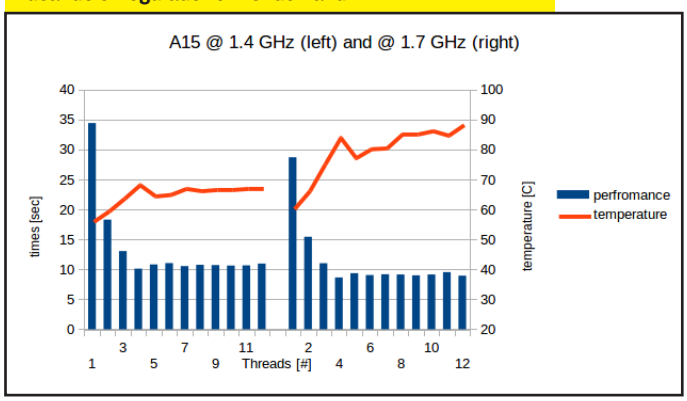
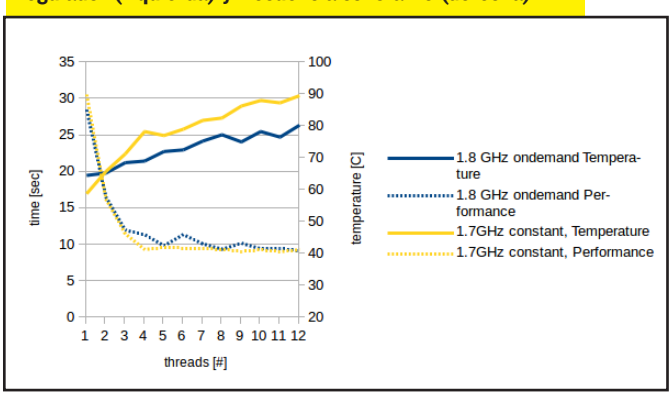


Figure 8 - ODROID-XU temperatura vs frecuencia usando regulador (izquierda) y frecuencia constante (derecha)



Da la impresión que hay más potencial de desarrollo. Digamos que XU puede cargar completamente los 4 núcleos a 1,6 GHz, de manera la frecuencia de 100 MHz podría ser utilizada por cada núcleo de forma decreciente. Por lo tanto, un regulador "turbo" podría funcionar a 1,9 con un núcleo, a 1,8 con dos núcleos, a 1,7 con tres núcleos y 1,6 GHz con 4 núcleos usados al completo. Veremos a ver qué posibilidades se podrán aprovechar en el futuro.

Referencias

[ME13] Memeka. Get to Know and Control big.LITTLE, ODROID Forum, 2013. **http://bit.ly/1oOODGP**

[HK14] Hardkernel product pages, 2013. **http://bit.ly/1hD2dIn.**

[OP14] Opas J. Estimating Radio Network Interference with Multi-threaded Java. ODROID Magazine, Issue #2, 2014.

DI ADIOS A NANO Y PÁSATE AL EDITOR VI

por Bruno Doiche

Incluso de vez en cuando, necesitas editar tu archivador "sudoers", y la forma por defecto para hacer esto es usar el comando visudo y recurrir a Vim para editar un archivo. Entonces ¿por qué no estableces por defecto el editor de texto de una vez por todas?
Sólo tiene que usar el comando update-alternatives escribiendo esto:

```
update-alternatives --config editor
```

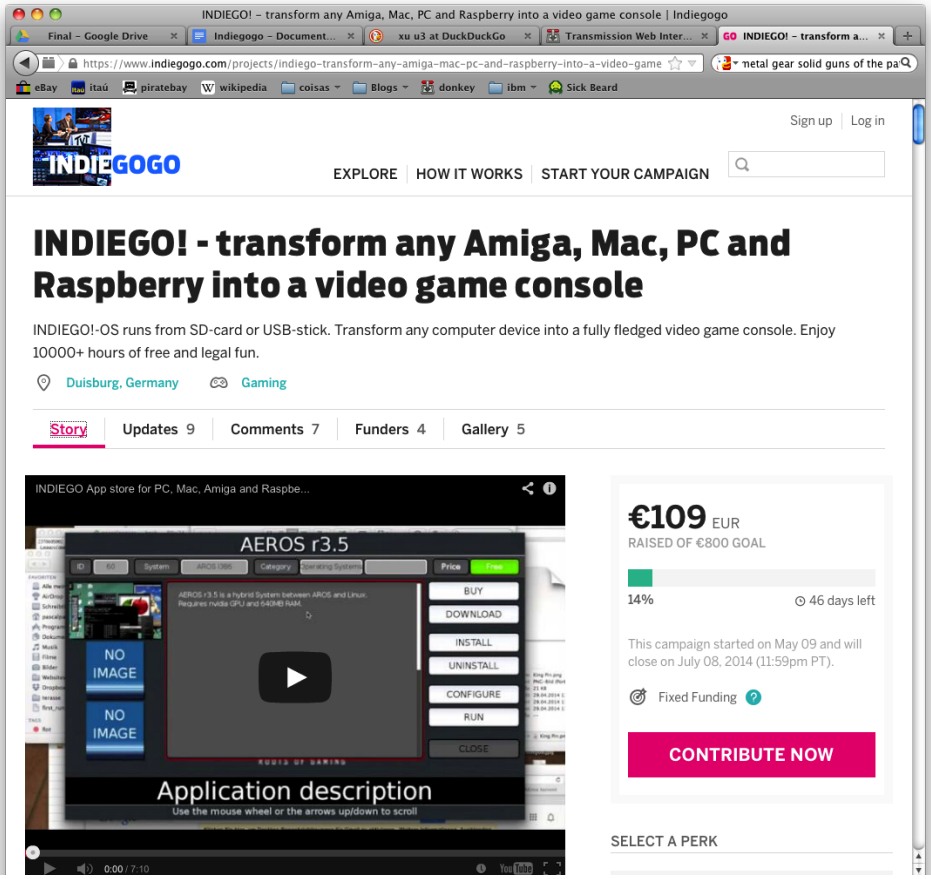
Ahora, te puedes sentir como un verdadero hacker mientras aprendes a usar Vim para editar tus archivos. Comienza usando la hoja de trucos que te dimos en el número 2, página 27.

UNA CAMPAÑA DE INDIEGGO PROMETE COMPATIBILIDAD ODROID CON UN AMBICIOSO OBJETIVO

CIRCUITOS INTEGRADOS PARA EL RESTO

por Bo Lechnowsky

Pascal Papara, desarrollador del sistema operativo Aeros, ha puesto en marcha una nueva campaña de Indiegogo que finaliza el 8 de julio de 2014. Tiene un objetivo de 800€ relativamente bajo ($ 1,100 USD), para la versión ODROID desarrollada y liberada en 1500€ ($ 2.000 USD). La campaña promete convertir cual-quier dispositivo, incluyendo el ODROID-U3, en un sistema de bajo coste similar a Steam Box. Los partidarios que donen 20€ ($ 27 USD) recibirán una distribución compa-tible con ODROID-U3 para tarjetas microSD, si el objetivo es alcanzado. Puedes leer más sobre la campaña en **http://bit.ly/1nppPXT**.



Stretch goal II at 1500€:

Make a SteamOS-like Distribution for Odroid U3

includes AEROS for Odroid as well

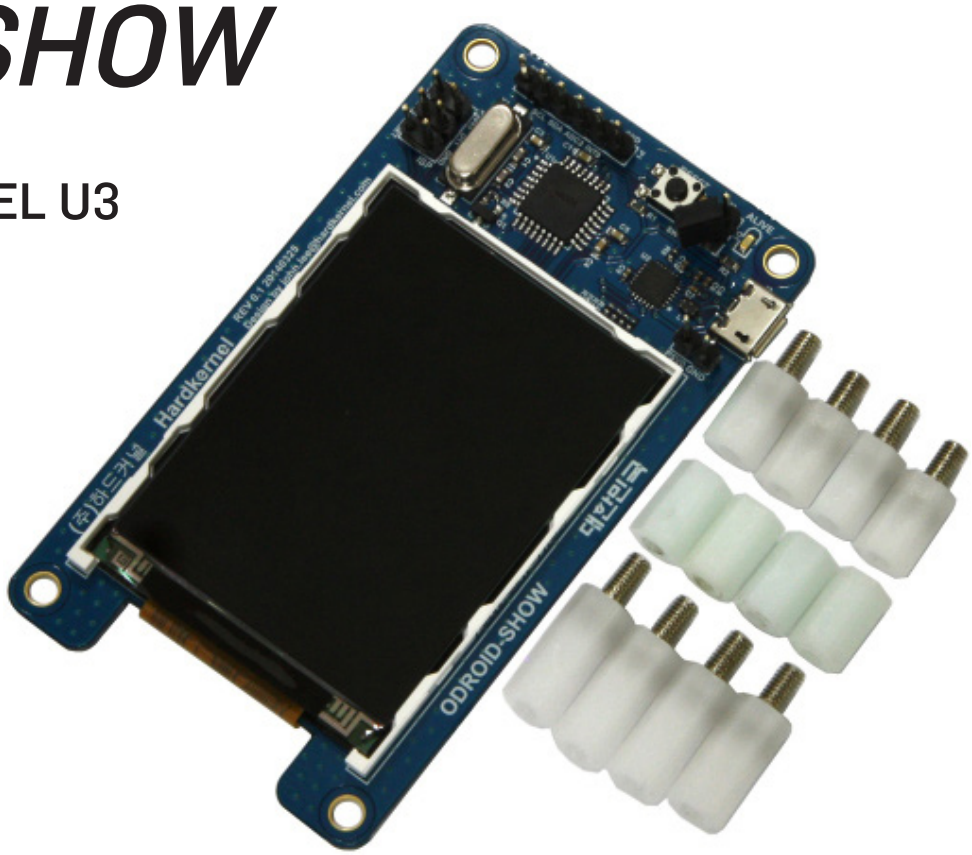
Make a SteamOS-like Distribution for SAM440/460

includes AEROS for SAM as well (SAM version will include lx command even if the last stretch goal is not reached)

ODROID-SHOW

UNA POTENTE MINI PANTALLA LCD PARA EL U3

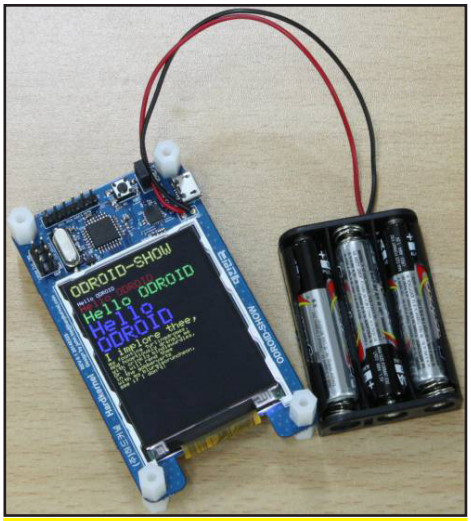
por Justin Lee y John Lee



El ODROID-SHOW es un nuevo dispositivo de Hardkernel compatible con Arduino que te permite ver lo que está pensado tu ODROID o PC mediante una pequeña LCD TFT de 2,2". Con precio de 25$ está diseñado para ser montado sobre ODROID-U3. Incluye separadores PCB y un cable USB.

Puedes mostrar textos de colores e imágenes vía interfaz USB con comandos VT100/ANSI, eliminando la necesidad de usar un monitor HDMI. No sólo puede conectar esta pequeña pantalla a tu ODROID, sino también a un Mac, PC Linux, PC Windows, o incluso a un servidor de empresa.

El ODROID-SHOW viene con clavijas GPIO, I2C y ADC para una mayor expansión y está pensado para insertar una placa adicional con sensores específicos para aplicaciones de robótica. Se puede convertir en un dispositivo completamente portátil añadiendo 3 o 4 pilas alcalinas. Debido a su bajo consumo de energía, es perfecto para proyectos portátiles.



ODROID-SHOW con pilas (baterias)

ODROID-SHOW - Especificaciones

MCU	ATmega328P a 16Mhz
LCD	2.2" 240x320 TFT-LCD (Interfaz SPI 8Mhz)
Interfaz Host	USB a UART via CP2104 en placa
Voltaje de entrada	3.7 ~ 5.5 Voltios
Consumo de energía	60mA @ 5 Voltios
Puerto Serie	Velocidad : 500,000 bps (0.5Mbps)
	Bits parada: 8-N-1
	Sin control de flujo H/W, S/W
Voltaje MCU/LCD	3.45 V desde el regulador voltaje en chip CP2104

Arquitectura de Hardware

El ATmega328 es el "cerebro principal" del ODROID-SHOW, que puede analizar el flujo desde UART y mostrar los datos en la pantalla TFT-LCD. El UART está conectado al PC o ODROID vía CP2104, que convierte el UART a una interfaz USB. La CP2104 también tiene un regulador de voltaje de 3.45V para suministrar la energía a la pantalla LCD. El regulador sobre chip permite un diseño de placa simple.

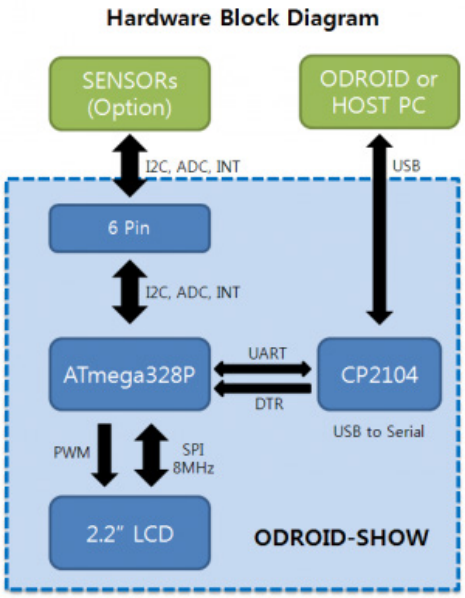
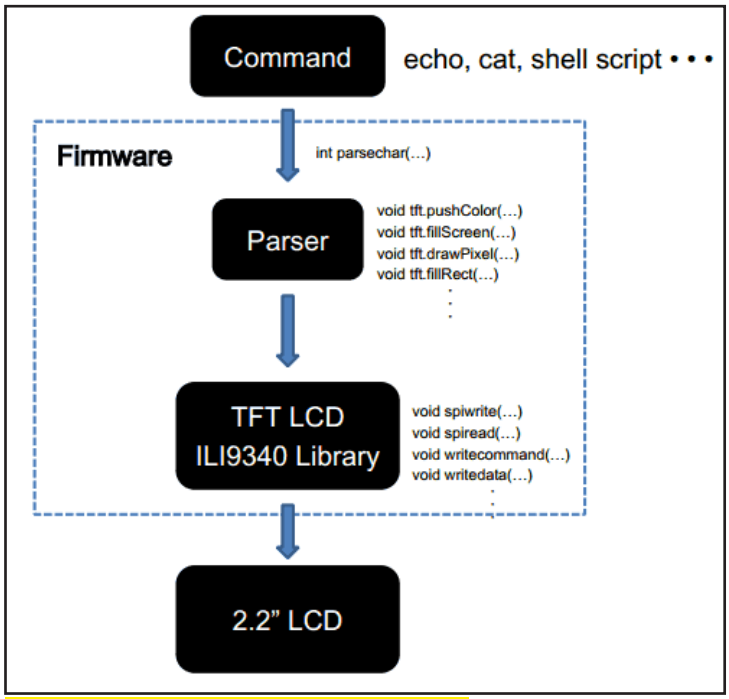


Diagrama por bloques del Hardware

El MCU contiene el gestor de arranque (Optiboot) en su memoria flash que es totalmente compatible con IDE Arduino. La memoria flash permite cambiar y mejorar el firmware con facilidad.

Arquitectura del Firmware



Arquitectura del Firmware

El firmware por defecto en el ODROID-SHOW fue compuesto con el analizador de comandos ANSI/VT100 y la librería TFT-LCD. El código fuente completo del firmware se puede descargar desde nuestro repositorio GitHub **http://www.github.com/hardkernel/ODROID-SHOW**. El código de la librería TFT-LCD fue desarrollado originalmente por Adafruit y nosotros lo perfeccionamos.

Cómo usarlo

Para enviar información (como cadenas de texto) a ODROID-SHOW, necesitas conocer los comandos Escape de ANSI/VT100.

Commanos Escape ANSI

Se necesitan códigos de terminal para enviar comandos específicos a tu ODROID-SHOW. Estos están relacionado con el cambio de colores o movimiento el cursor.

Nombre decimal	octal	hex	Descripción
ESC 27	033	0x1B	Escape character
CR 13	015	0x0D	Carriage return
LF 10	012	0x0A	Linefeed (newline)

Colores en primer plano

ANSI	Descripción
Esc [3 0 m	Set foreground to color #0 - black
Esc [3 1 m	Set foreground to color #1 - red
Esc [3 2 m	Set foreground to color #2 - green
Esc [3 3 m	Set foreground to color #3 - yellow
Esc [3 4 m	Set foreground to color #4 - blue
Esc [3 5 m	Set foreground to color #5 - magenta
Esc [3 6 m	Set foreground to color #6 - cyan
Esc [3 7 m	Set foreground to color #7 - white
Esc [3 9 m	Set default color as fg color - black

Colores de fondo

ANSI	Descripción
Esc [4 0 m	Set background to color #0 - black
Esc [4 1 m	Set background to color #1 - red
Esc [4 2 m	Set background to color #2 - green
Esc [4 3 m	Set background to color #3 - yellow
Esc [4 4 m	Set background to color #4 - blue
Esc [4 5 m	Set background to color #5 - magenta
Esc [4 6 m	Set background to color #6 - cyan
Esc [4 7 m	Set background to color #7 - white
Esc [4 9 m	Set default color as bg color - black

Commandos Escape VT100

(Pn = Numeric Parameter)

VT100	Descripción
Linefeed(\n)	Cursor Down
Esc D	Cursor Down
Esc E	Cursor Down to row 1
Esc M	Cursor Up
Esc c	Resets LCD
Esc [Pn A	Keyboard UP Arrow
Esc [Pn B	Keyboard Down Arrow
Esc [Pn C	Keyboard Right Arrow
Esc [Pn D	Keyboard Left Arrow
Esc [Pn ; Pn H	Cursor Position
Esc [H	Cursor to Home
Esc [2 J	Erase entire screen
Esc [6 n	Reports cursor position(serial port)

Commandos Escape VT100 Extendidos para ODROID-SHOW

Extended VT100	Descripción
Esc [s	Save cursor pos
Esc [u	Restore cursor pos
Esc [Pn s	Set text size (width = textsize*6, height = textsize*8)
Esc [r	Set rotation 0 to 3 (rotate to 90 in a clockwise)
Esc [0 q	Turn off LED backlight
Esc [1 q	Turn on LED backlight
Esc [Pn;Pn , Pn;Pn iStart image-drawing mode	

En primer lugar, crea y ejecuta el servicio demonio descrito al final de este artículo antes de pasar a los tutoriales.

Tutorial #1: Resultados de texto

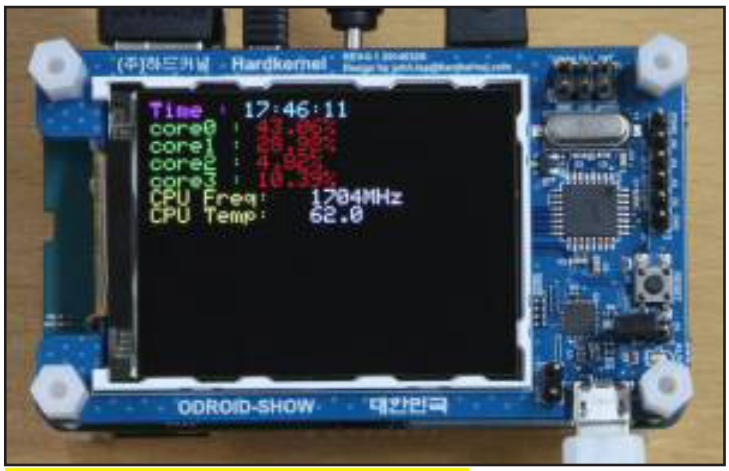
El script que se muestra a continuación puede visualizar 2 cadenas de texto con diferentes colores y tamaños de fuente. Para probarlo, abra el puerto "/ dev/ttyUSB0" y enviar comandos VT100/ ANSI con un par de cadenas de caracteres:

```
#!/bin/bash

flag=0
trap "flag=1" SIGINT SIGKILL SIGTERM
./port_open &
subppid=$!
```

ODROID SHOW con una muestra de texto

```
serialPort="/dev/ttyUSB0"

DATA[0]="ODROID"
DATA[1]="SHOW"

echo -ne "\e[5s\e[0r\ec" > $serialPort

sleep 0.1

while true
do
  if [ $flag -ne 0 ] ; then
    kill $subppid
    exit
  fi
  for ((j=1; j<8; j++)); do
    echo -ne "\e[25;100H" > $serialPort
    for ((i=0; i<6; i++)); do
                  echo   -ne   "\e[3"$j"m\
e[3"$j"m${DATA[0]:$i:1}" > $serialPort
      sleep 0.02
    done
     echo -ne "\eE\e[55;150H" > $serial-
Port
    for ((i=0; i<4; i++)); do
                  echo   -ne   "\e[3"$j"m\
e[3"$j"m${DATA[1]:$i:1}" > $serialPort
      sleep 0.02
    done
  done
done
```

Tutorial #2: Mostrar las estadísticas de tu ODROID



ODROID-SHOW con estadisticas

Este script muestra las principales estadísticas de ODROID, tales como el estado de carga de los 4 núcleos, frecuencia y temperatura de la CPU, junto con un reloj de tiempo real. Para ejecutar este script, primero necesita instalar sysstat usando *sudo apt-get install sysstat.*

```
!/bin/bash

flag=0

trap "flag=1" SIGINT SIGKILL SIGTERM

./port_open &
subppid=$!

function cpu_state {
cpuFreqM=$(echo "scale=0; " `cat \
/sys/devices/system/cpu/cpu0/cpufreq/
scaling_cur_freq` "/1000" | bc)
cpuTempM=$(echo "scale=1; " `cat \
/sys/class/thermal/thermal_zone0/temp`
"/1000" | bc)
}

echo -ne "\e[2s\e[3r\ec" > /dev/ttyUSB0
sleep 0.1

while true
do
```

```
    if [ $flag -ne 0 ] ; then
      kill $subppid
      exit
    fi
    echo -ne "\e[0;0H\e[35mTime : \e[36m" >
/dev/ttyUSB0
    date +"%T" > /dev/ttyUSB0
    sleep 0.1
    echo -ne "\eE\eM\e[32mcore0 : \e[31m" >
/dev/ttyUSB0
    sleep 0.1
    mpstat -P 0 | grep -A1 "usr" | grep -v
"usr" | awk '{print ""$4"%   "}` > \
      /dev/ttyUSB0
    sleep 0.1
    echo -ne "\eE\eM\e[32mcore1 : \e[31m" >
/dev/ttyUSB0
    sleep 0.1
    mpstat -P 1 | grep -A1 "usr" | grep -v
"usr" | awk '{print ""$4"%   "}` > \
      /dev/ttyUSB0
    sleep 0.1
    echo -ne "\eE\eM\e[32mcore2 : \e[31m" >
/dev/ttyUSB0
    sleep 0.1
    mpstat -P 2 | grep -A1 "usr" | grep -v
"usr" | awk '{print ""$4"%   "}` > \
      /dev/ttyUSB0
    sleep 0.1
    echo -ne "\eE\eM\e[32mcore3 : \e[31m" >
/dev/ttyUSB0
    sleep 0.1
    mpstat -P 3 | grep -A1 "usr" | grep -v
"usr" | awk '{print ""$4"%   "}` > \
      /dev/ttyUSB0
    sleep 0.1
    cpu_state
    echo -ne "\eE\eM" > /dev/ttyUSB0
    sleep 0.1
        echo   -ne   "\e[33mCPU   Freq:
\e[37m"$cpuFreqM"MHz   \eE" > /dev/ttyUSB0
   echo -ne "\e[33mCPU Temp: \e[37m$cpuTempM\e
" > /dev/ttyUSB0
    sleep 1
  done
```

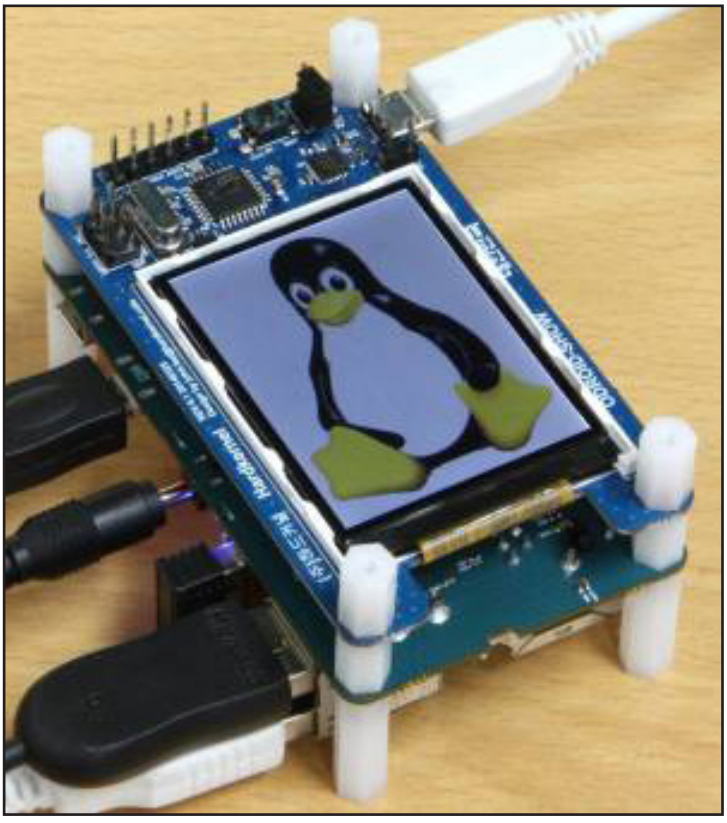
Tutorial #3: Mostrar un imagen

Además de texto, también puedes visualizar una imagen gráfica en ODROID-SHOW. Para ello, se recomienda usar ffmpeg para convertir un archivo PNG normal en un archivo raw RGB-565 (RGB-565 es el formato compatible). Para obtener los mejores resultados, primero se debe cambiar de tamaño del archivo PNG a 240x320 píxeles para ajustarse a la pantalla.

```
   ffmpeg -vcodec png -i penguin.png \
   -vcodec rawvideo -f rawvideo -pix_fmt
rgb565 penguin.raw
```

El archivo penguin.raw resultante estará listo para visualizarse en ODROID-SHOW. El modo de carga de la imagen se puede ajustar usando los parámetros de coordenadas de pixel.

```
   #!/bin/bash

   flag=0
   serial="/dev/ttyUSB0"

   trap "flag=1" SIGINT SIGKILL SIGTERM

   ./port_open &
   subppid=$!

   echo -ne "\e[0r\ec" > $serial

   while true
   do
```

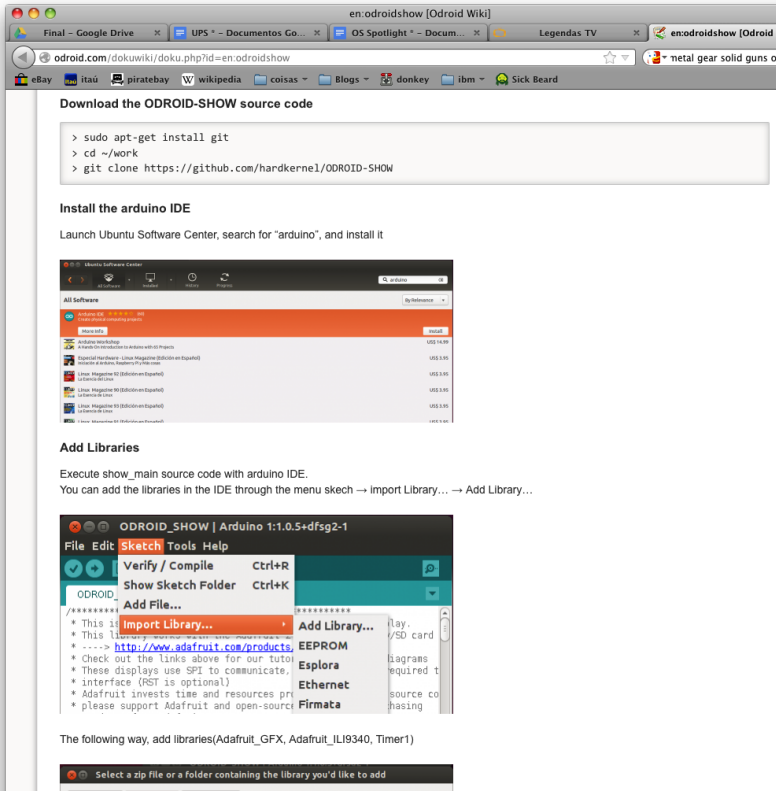


```
    if [ $flag -ne 0 ] ; then
      kill $subppid
      exit
    fi
    echo -ne "\e[0r" > $serial
    sleep 0.2
    echo -ne "\e[0;0,240;320i" > $serial
    cat penguin.raw > $serial
    sleep 0.2
    echo -ne "\e[1r" > $serial
    sleep 0.2
    echo -ne "\e[0;0,320;240i" > $serial
    cat butterfly.raw > $serial
    sleep 0.2
    echo -ne "\e[0r" > $serial
    sleep 0.2
    echo -ne "\e[0;0,240;320i" > $serial
    cat woof.raw > $serial
    sleep 0.2
  done
```

Debido a que los comandos "cat" y "echo" con redireccionamiento a "/ dev/ttyUSB0" siempre abren y cierran el puerto serie de forma automática, los datos que pasan hacia ODROID-SHOW se pueden dañar. Para evitar este problema, hemos escrito un pequeño programa que actúa como un demonio para gestionar la comunicación con el puerto serie.

```
  #include <stdio.h>
  #include <fcntl.h>
  #include <termios.h>
  #include <errno.h>

  #define baudrate    B500000

  const  char  serialPort[]  =  "/dev/tty-
USB0";

  int main(void)
  {
    int usbdev;
    struct termios options;

    usbdev = open(serialPort, O_RDWR | O_
NOCTTY | O_NDELAY);

    if (usbdev == -1)
        perror("open_port  :  Unable  to
open:");

    tcgetattr(usbdev, &options);
     cfsetispeed(&options, baudrate);
    cfsetospeed(&options, baudrate);

    options.c_cflag |= CS8;
    options.c_iflag |= IGNBRK;
    options.c_iflag &= ~( BRKINT | ICRNL |
IMAXBEL | IXON);
    options.c_oflag &= ~( OPOST | ONLCR );
     options.c_lflag &= ~( ISIG | ICANON |
IEXTEN | ECHO | ECHOE | ECHOK | ECHOCTL |
ECHOKE);
    options.c_lflag |= NOFLSH;
    options.c_cflag &= ~CRTSCTS;

    tcsetattr(usbdev, TCSANOW, &options);

    while(1)
      sleep(0.2);

    return 0;
  }
```

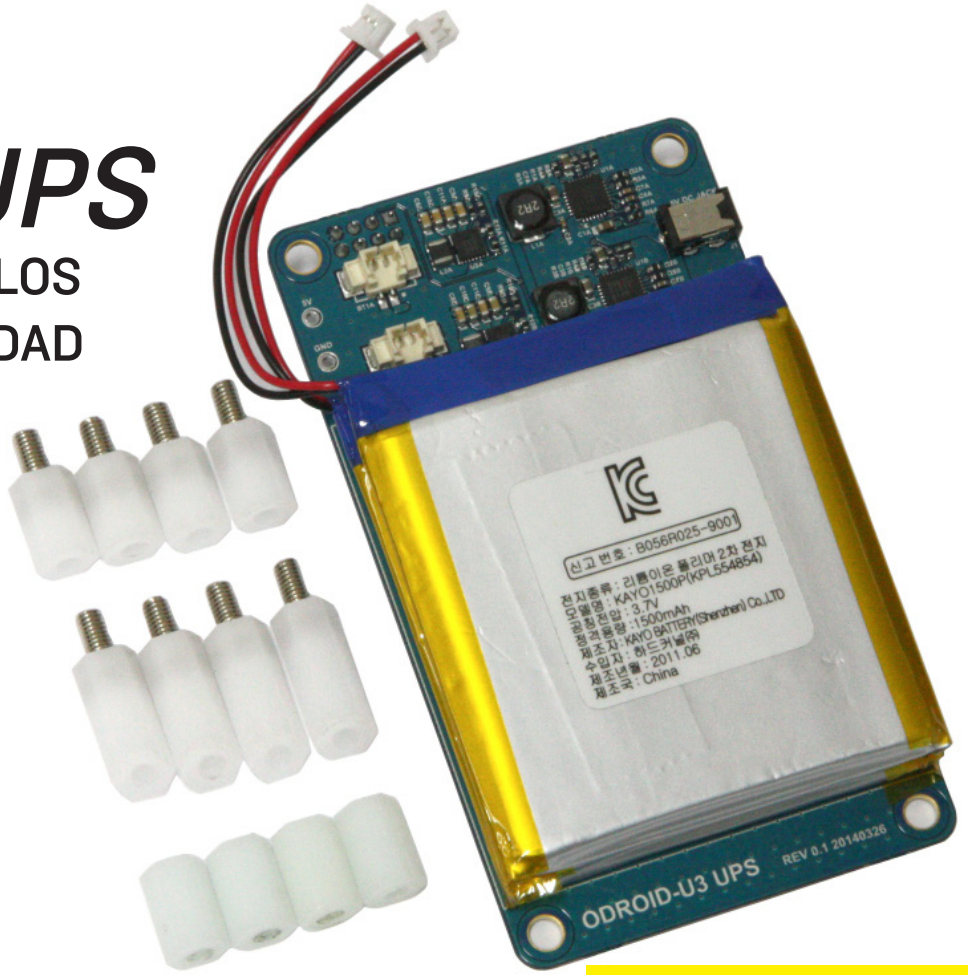
En primer lugar, modifica el número del puerto serie en el código fuente de arriba. A continuación, compila el demonio escribiendo gcc -o port_open port_open.c. Lance el ejecutable resultante port_open antes de ejecutar cualquiera de los scripts de ejemplo para evitar que se dañen los datos durante su transferencia al ODROID-SHOW. Para obtener información más detallada sobre la configuración de su ODROID-SHOW, visite **http://odroid.com/dokuwiki/doku.php?id=en:odroidshow**.



KIT ODROID-UPS

UNA SOLUCIÓN PARA LOS TIEMPOS DE INACTIVIDAD DE TU ODROID

por Justin Lee



Este pequeño puede hacer que tu ODROID funcione durante meses.

Hardkernel está orgulloso de anunciar el componente más reciente de la familia de periféricos del U3: el Sistema de Alimentación Ininterrumpida ODROID (ODROID-UPS). En aplicaciones industriales esenciales es importante asegurarse que la energía se mantiene constante en el caso de que la fuente de alimentación principal falle o se desconecte. Puesto que tiene el mismo formato que ODROID-U3, el ODROID-UPS se acopla correctamente en la parte superior de la placa con los separadores PCB, y se conecta a la toma de 8 pines en el U3.

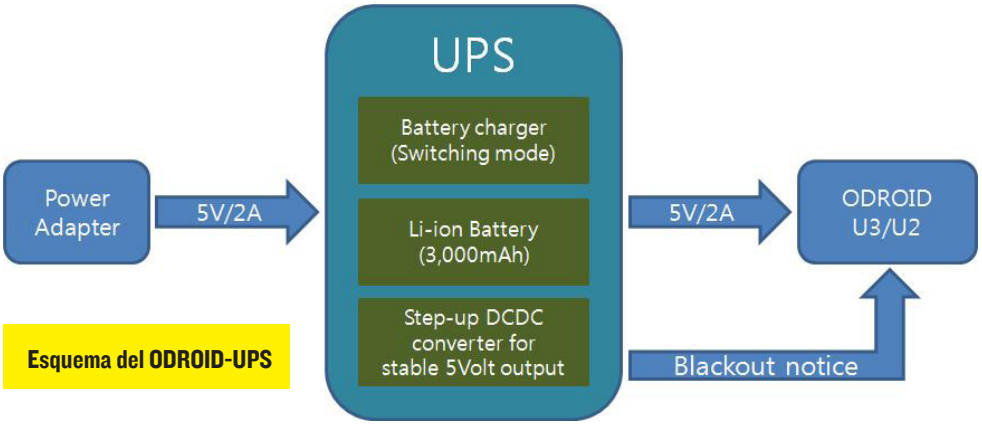
El kit UPS contiene el circuito del cargador, baterías y un circuito DC-DC con salida de 5V. El esquema completo se puede descargar desde http://bit.ly/1fDb3ds. Con una capacidad de 3000mA, ODROID-U3 puede funcionar alrededor de 1 o 2 horas con alta carga de trabajo sin necesidad de una fuente de alimentación.

Cargador de bacteria MAX8903C de IC Maxim

El MAX8903C está integrado por cargadores Li+ de 1 célula y reguladores de potencia inteligentes con entrada de energía variable. El cargador en modo switch utiliza una alta frecuencia para eliminar el calor y proteger los componentes externos. Todos los parámetros de potencia para cargar y desplazar la carga entre la batería y la fuente de alimentación externa esta incluidos en el chip de manera que no se requieren MOSFETs externos, diodos de bloqueo, o resistencias de corriente.

El MAX8903C ofrece un control de potencia inteligente permitiendo un mejor uso de la energía. La corriente del cargador de batería y la corriente de la salida SYS están configuradas de forma independiente (hasta 2A), y el sistema carga la batería con cualquier resto de potencia desde el adaptador de corriente del ODROID. La selección de entrada automática cambia el sistema de la batería a la alimentación externa, y la entrada DC funciona de 4.15V a 16V con una protección de hasta 20V.

El MAX8903C internamente bloquea la corriente desde la batería y el sistema a la entrada DC cuando no hay alimentación de entrada. Otras características incluyen el temporizador y la precarga, temporizador de carga rápida, protección contra sobretensiones, estado de carga, fallos de salida y monitoreo de potencia. Además, La limitación térmica sobre chip reduce la tasa de carga de la batería y la corriente de la fuente de alimentación para evitar el sobrecalentamiento del cargador.



Esquema del ODROID-UPS

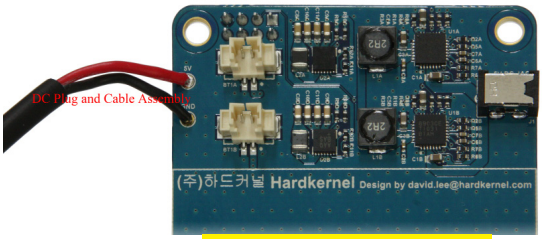
Batería Li-Ion de Polímero

El kit UPS tiene dos baterías de Li-ion de polímero que están conectados en paralelo. La capacidad de cada batería es de 1500mA con una capacidad total de 3000mA. La tensión de carga máxima es de 4.2V

Convertidor DC-DC S MAX8627 de IC Maxim

Hemos añadido un convertidor de refuerzo puesto que la tensión de salida de la batería Li-ion varía de 3,6 a 4,2 V pero ODROID-U3 necesita una entrada de 5V. MAX8627 es un convertidor de alta eficiencia, con baja corriente en reposo, sincronizado con True Shutdown ™ y limita la corriente de entrada. El MAX8627 genera 5V usando una batería de polímero Li +/Li de una sola célula.

Si la tensión normal es de 3.8V, la capacidad media de la batería es de aproximadamente 11.5Wh. Si el componente quimico y la eficiencia eléctrica funciona a aproximadamente 70% la capacidad real es de 8Wh. Si tu sistema consume 2W, el kit UPS puede funcionar durante unas 4 horas antes de necesitar ser recargado.



Primer plano del circuito de la placa

Si los puertos host USB de ODROID-U3 no funcionan, soldar el cable DC a la placa UPS y conéctelo a la clavija DC en ODROID U3. Esto garantiza una fuente de alimentación estable para los dispositivos USB.

Ejemplo de apagado automático

Las señales AC_OK están conectadas a GPIO199/GPIO200 en la toma principal de 8 pines del ODROID-U3. Cuando se produce un apagón o una desconexión repentina del suministro eléctrico, el sistema se apagará automáticamente después de 1 minuto usando el siguiente script, que comprueba continuamente el estado de la alimentación eléctrica.

```
#!/bin/bash
echo 199 > /sys/class/gpio/export
echo 200 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio199/direction
echo in > /sys/class/gpio/gpio200/direction

get_ac_status() {
  ac1=`cat /sys/class/gpio/gpio199/value`
  ac2=`cat /sys/class/gpio/gpio200/value`

  if [ "0$ac1" -eq 1 -o "0$ac2" -eq "1" ]; then
    export ACJACK="off"
  else
    export ACJACK="on"
  fi
}

while :
do
  get_ac_status

  if [ "$ACJACK" == "off" ]; then
    shutdown -P 1
  fi

  sleep 1;
done
```

Nota del Diseño

El proyecto UPS fue creado porque teníamos un exceso de baterías en nuestro inventario. Sin embargo, el circuito de protección interna de nuestras baterías era demasiado sensible para la carga de corriente tan intensa, así que decidimos usar 2 celdas en paralelo. Otra cuestión era el desequilibrio de energía en las dos baterías separadas, que puede conllevar un riesgo químico. Con el fin de abordar esta cuestión, hemos implementado dos veces los circuitos sobre el diseño de la placa UPS, lo que significa que los esquemas no están tan bien optimizados.

Si está pensando en hacer tu propio sistema de baterías, asegúrate primero de comprobar la corriente de salida máxima de la batería. Cuando la carga eléctrica es muy intensa, el circuito de protección de las baterías desconecta la carga automáticamente. Para restablecer el circuito de protección debes desconectar y reconectar los conectores de la batería. Aunque se puede comprobar el estado de entrada máxima de AC (AC_OK) utilizando un script, puede no ser suficiente. Cuando se utiliza el ODROID-UPS con un sistema de energia, tendrás que incluir circuitos adicionales para transmitir el nivel de la batería a la gestión inteligente de la energía.

Para especificaciones más detalladas y comprar tu propio ODROID-UPS, visita en siguiente enlace: `http://bit.ly/1fDb3ds`.

SO DESTACADO: FULLY LOADED

UBUNTU 12.11 CON EL ENTORNO DE ESCRITORIO UNITY 2D

por Rob Roy, Editor Jefe

Fully Loaded con Kernel 3.0 para U2/U3/X/X2 puede descargarse desde `http://bit.ly/1rhHymu`. Vigila los foros en `http://forum.odroid.com/` Fully Loaded II con Kernel 3.8, estará dispinible muy pronto!"

Los foros ODROID ofrece muchas y excelentes imágenes de sistemas operativos pre-compilados, cada una cuenta con personalizaciones únicas para usos muy diversos como reproductor multimedia, desarrollo de software, música y robótica. Puesto que la familia de ordenadores ODROID pretende ser una plataforma de desarrollo, muchos usuarios prefieren compilar sus propios sistemas operativos para tener un completo control sobre todo los aspectos del hardware. En el número anterior de ODROID Magazine, el desarrollador de Hardkernel Mauro Ribeiro presentó una guía útil para desarrollar tu propia versión de Ubuntu desde el código fuente con el fin de ayudar a aquellos que quieren aprender a "hacerlo por sí mismos". Pero ¿Y si sólo quieres utilizar tu ODROID al instante sin tener que invertir conocimientos técnicos y tiempo en la compilación de tu propio sistema operativo?

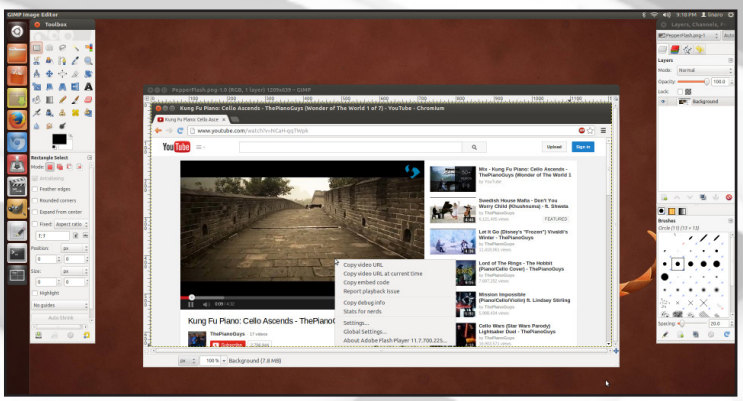
Fully Loaded, se presentó por primera vez en 2013 y ha sido actualizada con regularidad. Fue una de las primeras imágenes de la comunidad en ofrecer una experiencia de escritorio preconfigurada sin tener que invertir tiempo configurando software, instalando aplicaciones y depuración entornos de escritorio. Contiene casi todos los entornos de Ubuntu disponible para 12,11 incluyendo Gnome, Lubuntu (LXDE), Kubuntu (KDE), Blackbox, Openbox, Fluxbox, Unity Xubuntu (XFCE). Se puede cambiar el entorno de escritorio con el icono circular junto al nombre de usuario en la pantalla de inicio de sesión. Yo personalmente recomiendo KDE Plasma Workspace por sus efectos visuales acelerados por hardware y una interfaz al estilo Windows 7, pero cada entorno tiene sus particularidades.

Para empezar, descarga y copia la imagen Fully Loaded para tu hardware (X, X2, U2/U3) a un módulo eMMC o tarjeta SD. Para obtener más información sobre cómo hacer esto, consulta artículo sobre "Cómo copiar un archivo de imagen", también incluido en este número de la revista. Una vez que inicies la imagen, verás la típica pantalla de inicio de sesión de Ubuntu con una imagen de su mascota, el Pangolín. Si tienes el sonido activado, también oirás el clásico sonido de tambor único de Ubuntu.

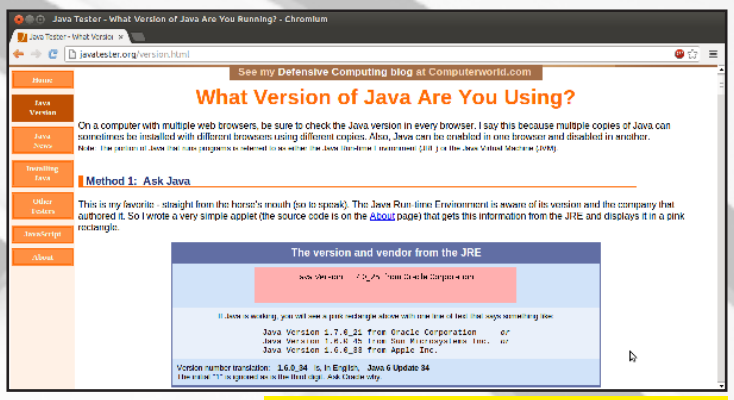
El nombre de usuario por defecto para Fully Loaded es "linaro" con la contraseña "linaro". Después de iniciar sesión, puedes configurar tu zona horaria local, seleccionar tu idioma y crear usuarios, haciendo clic en la opción de menú "Ajustes del sistema" en la esquina superior derecha del escritorio. La contraseña de root es también "linaro", pero este inicio de sesión sólo se debe usar para mantenimiento y no para uso diario con el fin de evitar daños accidentales al sistema de archivos. Una de las primeras cosas que verás cuando uses Fully Loaded es que el escritorio Unity está fijado por defecto, ya que ha resultado ser muy popular entre principiantes y expertos por sus iconos amigables, una cómoda barra de tareas y una estabilidad incomparable . Puesto que la plataforma ODROID soporta la librería de gráficos GLES en lugar de OpenGL, la versión 2D de Unity suele preferirse por su mejor rendimiento. Se ejecuta mucho más rápido que la versión 3D estándar, eliminando gran parte de las extrañas "animaciones visuales" y el correspondiente retraso gráfico que fue introducido con Ubuntu 13.04.

El entorno de escritorio es una cuestión personal y la elección del éste no afecta a la librería de software disponible. Ubuntu está diseñado para mantener la coherencia en sus aplicaciones al tiempo que permite una completa personalización de la interfaz gráfica de usuario (GUI). Este artículo incluye un resumen de las principales aplicaciones disponibles en Fully Loaded, las cuales son de código abierto y están disponibles gratuitamente en el Centro de Software de Ubuntu.

Ubuntu 12.11 es uno de los más estables sistemas operativos disponibles para ODROID



GIMP, programa de edición de imágenes de GNU, tiene un largo historial de desarrollo.



Los Applets Java en el navegador web ofrece compatibilidad multiplataforma.

GIMP (Programa de edición de imágenes de GNU)

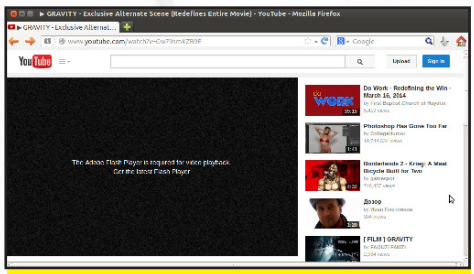
¿Quién necesita comprar el Photoshop cuando se puede tener uno de más potentes editores gráficos y completamente gratis? GIMP ha estado en continuo desarrollo desde 1996 e incluye una amplia librería de mejoras y aportaciones de los usuarios. Se necesita algo de experiencia para su uso, sin embargo los resultados pueden llegar a ser sorprendentes.

GIMP tiene casi todo lo que ofrece Photoshop, con la posibilidad de crear, modificar y guardar las imagenes en varios formatos, JPG, GIF, PNG, PSD, y AutoCAD. La caja de herramientas en el lado izquierdo contiene botones que se usan para la selección de área, pinceles, edición de texto, cambio de color, enmascaramiento, clonación y formas. Muchos de los efectos visuales también están disponibles desde el menú Filtros de la ventana central, tales como desenfoque, nitidez, ruido, detección de bordes, sombras y otros procesos gráficos.

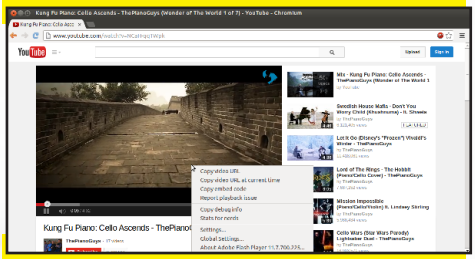
GIMP también incluye un potente plugin llamado "Script-Fu", basado en el lenguaje Scheme. Puedes diseñar tus propios procesos y efectos visuales utilizando transformaciones matemáticas complejas y luego, compartir tu trabajo con otros. Más información sobre el uso de Script-Fu para mejorar GIMP en `http://bit.ly/1fBPgTA`.

Navegadores Web

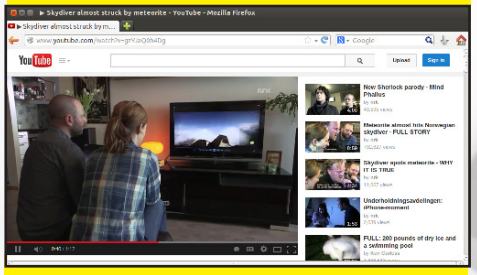
Un entorno informático moderno no estaría completo sin un navegador web. Fully Loaded incluye Firefox y Chromium, ambos ofrecen una experiencia de navegación completa con soporte tanto para Java como Flash. El plugin de código abierto IcedTea está activado en Firefox y Chromium, que permite a los applets Java ejecutarse dentro de un navegador. Adblock Plus también está instalado en ambas aplicaciones, para evitar que los anuncios y ventanas emergentes interfieran en tu sesión de Internet.



¡Diantres! No hay Flash. ¿En serio?



Genial! Hay Flash, pero no es Adobe.



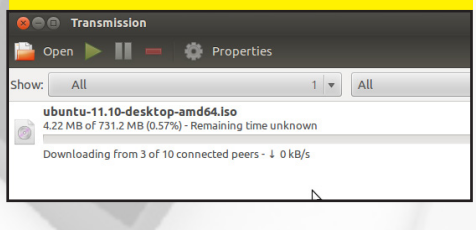
Mira mamá! Podemos ver cosas sin flash!

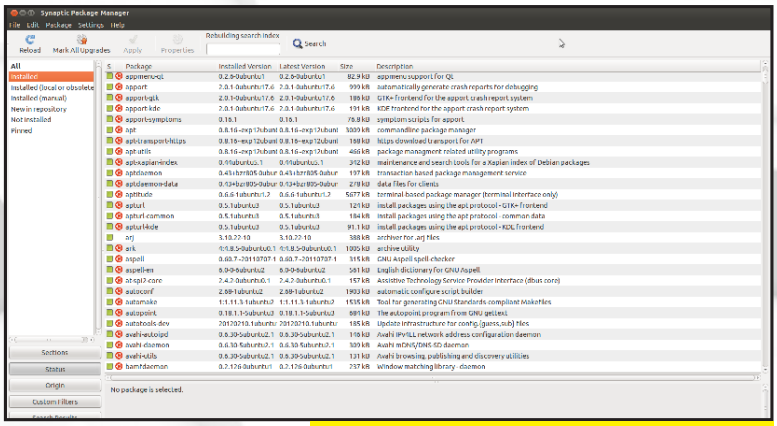
Cuando navegamos por Youtube o sitios similares, Chromium y Firefox soportan el reproductor HTML5 recientemente añadido, pero no todos los videos en Internet están disponibles en este formato. Aunque Firefox no incluye un reproductor Flash de código abierto, el plugin PepperFlash opcional para Chromium reemplaza el reproductor Flash estándar y permite a dispositivos ARM como ODROID reproducir vídeos Flash, aunque Adobe ya no lo soporta.

Transmission

Transmission es el cliente de Linux estándar para el protocolo BitTorrent, que permite descargar archivos desde una red P2P, con velocidades de descarga muy altas al usar una red de ordenadores en lugar de acceder a un solo equipo. Para utilizar Transmission, inicie Firefox y vaya a cualquier sitio web que ofrezca torrents, luego haga clic en el enlace Magnet para descargar el archivo torrent. Transmission se iniciará automáticamente y comenzar la descarga, guárdalo en el directorio de descargas, una vez completado. Chromium también puede ser configurado para usar torrents, pero Firefox ya está asociado a Transmission por defecto.

Torrent, la forma de conseguirlo todo!





¿La interfaz gráfica para apt-get? Synaptic por supuesto!

Xine es tan increíble, que vamos a enumerar todo lo que puede soportar:

- **Medios físicos: CDs, DVDs, Video CDs[6]**
- **Formatos contenedores: 3gp, AVI, ASF, FLV, Matroska, MOV (QuickTime), MP4, NUT, Ogg, OGM, RealMedia**
- **Formatos de audio: AAC, AC3, ALAC, AMR, FLAC, MP3, RealAudio, Shorten, Speex, Vorbis, WMA**
- **Formatos de video: Cinepak, DV, H.263, H.264/MPEG-4 AVC, HuffYUV, Indeo, MJPEG, MPEG-I, MPEG-2, MPEG-4 ASP, RealVideo, Sorenson, Theora, WMV (parcial incluyendo WMV1, WMV2 and WMV3; via FFmpeg)**
- **Dispositivos de Video: V4L, DVB, PVR**
- **Protocolos de red: HTTP, TCP, UDP, RTP, SMB, MMS, PNM, RTSP**

(y los mejores y mayores subtítulos de pantalla jamas vistos!)

Xine y ffmpeg

Si quieres ver un video descargado en tu ODROID, Xine es la mejor aplicación disponible para 12,11 y es compatible con muchos formatos conocidos. Aunque se trata de descodificación por software, la mayoría de los vídeos 720p se reproducen muy bien en el U3 y los videos a 1080p son aceptables a pesar de que algunos fotogramas se ralentizan. Puesto que la decodificación de video por hardware no está disponible con el Kernel 3.0, cuando se desean ejecutar videos a 1080p en Linux es aconsejable usar la imagen Ubuntu 13 Dream Machine con XBMC.

Para usar Xine, haz doble clic en cualquier vídeo desde el explorador de archivos y pulsa "g" para mostrar el HUD, que contiene los controles de búsqueda, controles de volumen y otras funciones útiles. Es básicamente un envoltorio para la potente utilidad de reproductor de vídeo ffmpeg. Fully Loaded incluye una versión especial de ffmpeg compilado específicamente para la arquitectura NEON de la GPU Mali. FFmpeg también se puede activar sin Xine escribiendo ffmpeg desde la línea de comandos.

Gestor de paquetes Synaptic

Synaptic es la principal aplicación para descargar nuevos paquetes de software y actualizar los existentes. Ofrece miles de librerías de desarrollo, paquetes completos, entornos de escritorio y mucho más. Si está usando ODROID para el desarrollo de software y deseas instalar las dependencias que te faltan, este es el mejor lugar para encontrarlas. Descubrirás muchas joyas ocultas en Synaptic, si dedicas algo de tiempo a mirar la enorme lista de paquetes de código abierto disponibles. La contraseña para iniciar Synaptic es "Linaro".

Centro de Software Ubuntu

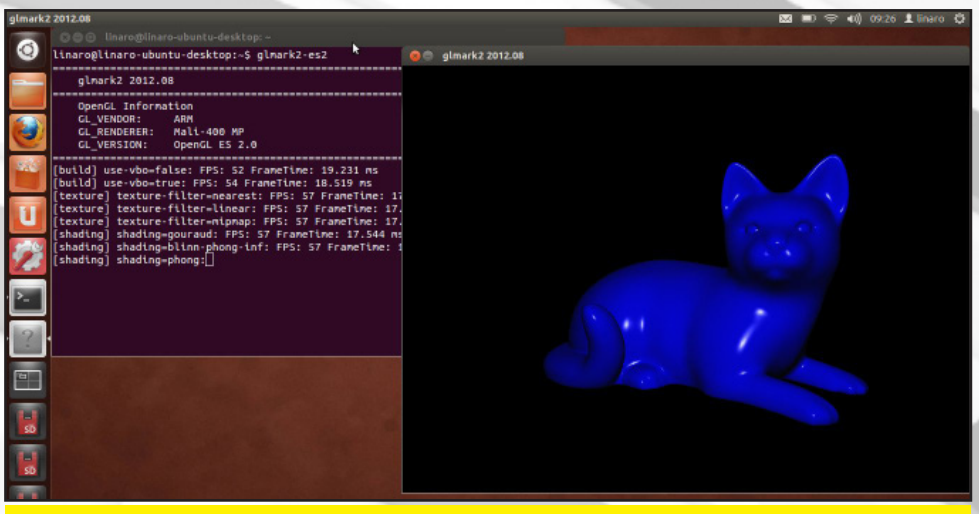
El Centro de Software Ubuntu es una interfaz amigable de los repositorios de software Canoncial que ofrece paquetes similares al Gestor de paquetes Synaptic, pero en un formato más amigable. Su ventaja sobre Synaptic es que el software esta clasificado e incluye una breve descripción de cada aplicación, pero no incluye las librerías de desarrollo en sus listas. Es el equivalente a iTunes para OSX o Google Play Store para Android.

Terminal

Muchos de los artículos que aparecen en los foros y en esta revista necesitan incluir cadenas de comandos en la ventana de terminal, que es la interfaz de línea de comandos (CLI) estándar que acompaña a casi todas las distribuciones de Linux. Comandos útiles como *sudo,* que permite a un comando ser ejecutado con acceso de superusuario, *ls* que muestra el contenido de un directorio y *cd* que navega a una carpeta especificada. Escribe *cd ~* para ir a la carpeta de inicio y presione Tab para utilizar la función de autocompletado. También puede pulsar la flecha hacia arriba para volver a utilizar los comandos recientemente escritos. Un atajo útil para iniciar Terminal en la mayoría de los entornos de escritorio de Ubuntu consiste en presionar la teclas Ctrl-Alt-T.

Drivers Mali 3D

La GPU Mali de los modelos X, X2, U2 y U3 tiene un gran potencial 3D. Puedes realizar una prueba gráfica, escribiendo es2gears o glmark2-es2 en la ventana de Terminal. La animación de medusas en glmark2-es2 es especialmente bonita. Los desarrolladores gráficos y de juegos deben familiarizarse con los comandos de OpenGL ES 2.0 para programar en ODROID, que se trata de un subgrupo optimizado del lenguaje original OpenGL.



¿Eso es una captura de pantalla que muestra el uso simultánea de Terminal y driver Malí 3D, o una referencia secreta a nuestro artículo del gato en siesta del número anterior?

Kit de desarrollo Java Oracle (JDK8)

Fully Loaded viene con Oracle JDK8 instalado, lo que permite a los programas Java como Minecraft Server ejecutarse desde la línea de comandos. La máquina virtual de Java se puede activar escribiendo *java* en la ventana de Terminal. Muchos de los programas que se han escritos para otro hardware, por lo general también funcionan en ODROID, ya que Java es un lenguaje independiente de la plataforma en la que se ejecuta.

Mednafen

Hay una enorme colección de juegos disponibles para ODROID y Mednafen soporta muchos y diferentes sistemas emulados, incluyendo Gameboy, NES, SNES y Sega. Un script que viene con Fully Loaded denominado play_rom ajusta automáticamente los valores óptimos para la escala y la resolución en Mednafen. Los Archivos ROM se pueden ejecutar haciendo doble click desde el explorador de archivos, o ejecutándolos usando el comando <rom file> de play_rom en la ventana de Terminal. Para más información sobre Mednafen y formatos soportados consulte la guía práctica en **http://bit.ly/1pYi1hu**.

Otros consejos y trucos

Para arrancar directamente en el entorno de escritorio utilizado recientemente, escriba:

```
sudo /usr/lib/lightdm/
lightdm-set-defaults
--autologin linaro
```

en la ventana de Terminal. Este comando pasa por alto la pantalla de inicio de sesión ahorrando tiempo, si reinicias con frecuencia. Para cambiar el entorno por defecto, sólo tiene que cerrar sesión en el escritorio actual y selecciona otro en la pantalla de inicio de sesión.

Para aumentar la velocidad y el rendimiento de tu ODROID, Fully Loaded permite "overclocking" a 1.92 GHz tecleando *sudo gedit/etc/rc.local* en la ventana de Terminal y eliminando la "#" al principio de la línea que comienza por "echo 1920000". Es muy recomendable utilizar un ventilador cuando se realiza "overclocking" con el fin de evitar bloqueos por sobrecalentamiento

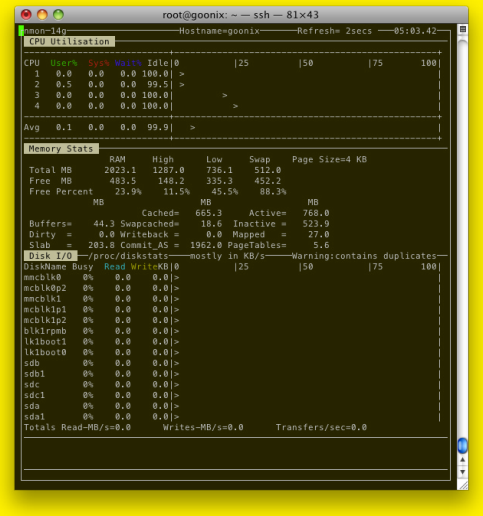
Si dispones de más de un ODROID, Fully Loaded incluye un práctico script de intercambio de kernel para permitir el arranque desde una única MicroSD compartida entre U2 y un X2. Para ello, escriba */media/boot/tools/swap_odroid.sh [u2 | x2]* en la ventana de Terminal antes de apagarlo. Para más detalles, visite **http://bit.ly/1lIDHtQ**.

El kernel de Fully Loaded también soporta memoria virtual para extender la memoria de tu ODROID por encima de los 2 GB. Para más información sobre el archivo de intercambio y cómo activarlo, visite **http://bit.ly/1pYfWSY**.

MONITORIZA
TU LINUX CON NMON

por Bruno Doiche

Sigues usando la parte superior para monitorizar tus estadísticas globales del sistema... Intenta usar NMON. Es una gran herramienta para monitorizar casi todo, desde tus procesos a las conexiones de red. Además de supervisar tu sistema, puedes exportar los datos a un archivo .csv para crear informes detallados del rendimiento de tu ODROID.




```
sudo apt-get install nmon
```

Para recopilar los datos, ejecute nmon como el siguiente ejemplo (-f significa que el nmon registrará un archivo de datos, -s es el tiempo entre actualizaciones y -c el recuento de actualizaciones que nmon hará para poner fin a la recogida de datos), que tendrá una duración de 1 hora:

```
nmon -f -s 30 -c 120
```

Nmon creará un archivo en el directorio actual:

```
<hostname>_date_time.nmon
```

Cuando se ejecuta con estos indicadores, nmon no muestra la misma interfaz gráfica puesto que se ejecuta en segundo plano. Puedes cerrar la sesión mientras se recopilan tus datos para una futura revisión.

CONSTRUIR UN VEHICULO TODOTERRENO AUTOMATICO CON ODROID

PARTE I: ESQUEMA GENERAL, MONTAJE DE PLATAFORMA, Y DISTRIBUCION DE ENERGIA

por Christopher D. McMurrough

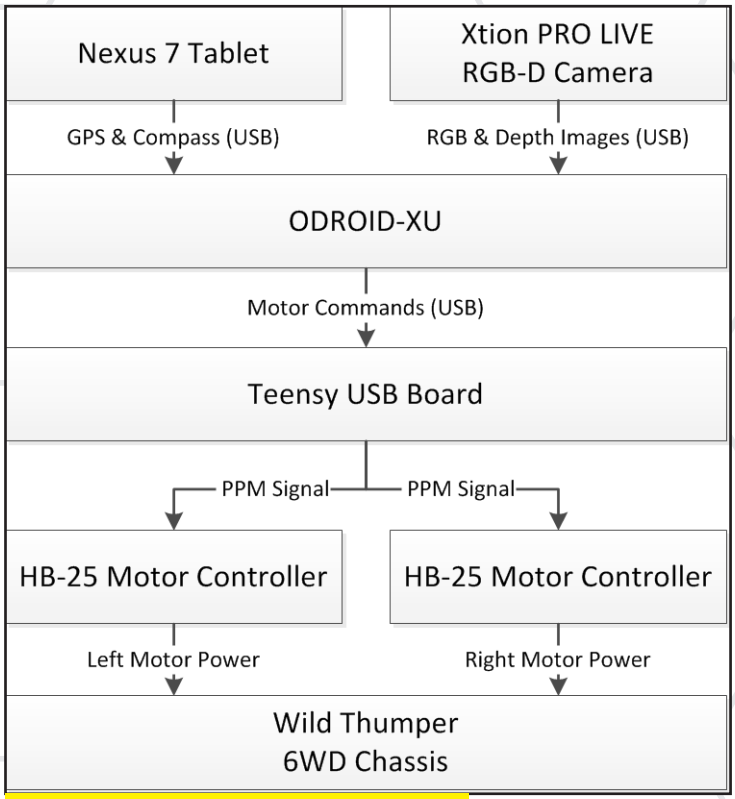
En esta serie de artículos, vamos a construir nuestro propio vehículo todoterreno automático (VTA) usando la placa de desarrollo ODROID-XU. Nuestro objetivo será la creación de un robot que sea capaz de atravesar terrenos al aire libre mientras se desplaza entre puntos GPS y también, presentar al lector una plataforma sólida para futuros desarrollos. Usaremos los datos de navegación facilitados por un dispositivo Android y obtendremos la información del entrono en 3D mediante una cámara RGB-D. La serie se divide en 3 artículos: Esquema general de la plataforma y distribución de energía, la interconexión de motores y sensores con ODROID, y programar el robot para seguir de forma autónoma Puntos GPS.

Si esto fuera una clásica revista que se vende en un quiosco, la imagen del VTA de Chris nos obligaría a usar pañuelos para las babas!

Introducción

Voy a suponer que tú, el lector, al que le gusta mucho ODROID (después de todo, estás leyendo ODROID Magazine) es muy probable que también le gusten los robots. La buena noticia es que los ODROIDs son perfectos como robots! No obstante, el desarrollo de un robot simpre es un reto. Tanto si acabas de empezar como si has trabajado con ellos con anterioridad, cada robot es diferente y requiere muchas y diferentes consi-deraciones para que el proyecto llegue a ser un éxito. Esta serie tiene la intención de proporcionar una visión general sobre la creación de un completo sistema VTA como tantos disponibles a nivel comercial. Vamos a cubrir los aspectos del diseño mecánico, eléctrico y de software de nuestro sistema, y ofreceremos material complementario, como esquemas y código fuente. En este primer artículo, nos centraremos en el montaje mecánico general y la distribución de energía. Aunque no pretende ser una completa y exhaustiva guía práctica que muestre cada paso del proceso de desarrollo, animo a realizar preguntas y comentarios sobre esta serie de artículos en el debate abierto sobre " Robótica ODROID" en el Foro ODROID (Boardindex/Hardkernel/General Chat/ODROID Robotics). Responderé a las preguntas y facilitaré más detalles a quien lo solicite. Mi intención es empezar con un activo debate sobre robótica dentro de la comunidad ODROID, así que por favor !Participa!

Componente	Cta	Precio	Fuente
ODROID-XU	1	169.00	Hardkernel
Wild Thumper 6WD Chassis	1	249.95	Pololu
Nexus 7 Tegra 3	1	149.99	Newegg
Asus XTION PRO LIVE	1	169.99	Newegg
HB-25 Motor Controller	2	49.99	Parallax
M2596 Buck DC-DC Adjustable PSU	1	3.45	Amazon



Arquitectura del Sistema VTA

Esquema General del Sistema

Nuestro robot, cuando esté completado, podrá desplazarse entre puntos GPS al mismo tiempo que evitará obstáculos. Usaremos el GPS y la brújula desde una tableta Nexus 7 Android como sensor de posición y la información 3D desde una cámara en Xtion Pro Live RGB-D, para evitar los obstáculos. Estos dispositivos proporcionan un montón de información que no necesariamente necesitamos para nuestra demostración de seguimiento de coordenadas, pero pueden ser útiles en futuros proyectos. ODROID-XU con Ubuntu Linux procesará la información de estos dispositivos utilizando Robotic Operating System (ROS), que trataremos más a fondo en el próximo artículo de la serie.

El chasis que usaremos es el 6WD Wild Thumper. Esta plataforma es ideal para entornos todoterreno, puesto que cada uno de los 6 sistemas de propulsión cuenta con suspensión independiente. Los motores están diseñados para funcionar con pack de pilas RC estándar de 7,2 voltios, y el chasis tiene espacio para insertar 4 pack debajo de la placa de montaje superior. Vamos a alimentar nuestro chasis con 3 pack de pilas NiMH (capacidad de 3000 mAh) conectadas en paralelo que nos dará un total de 9000 mAh de potencia de motor. El cuarto pack de pilas estará destinado a alimentar el ODROID, sensores y otros dispositivos electrónicos. Separar estas pilas del resto proporciona a la electrónica una capa de aislamiento eléctrico, y evitará que nuestro sistema de reinicie debido a las fluctuaciones de potencia relacionadas con la corriente del motor. La pilas de los dispositivos electrónicos se regularán a unos 5 voltios usando la fuente de alimentación LM2596 DC-DC.

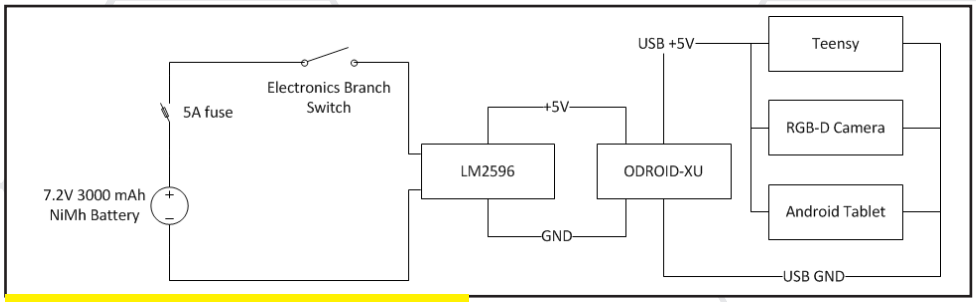
Usaremos un doble controlador de motor HB-25 para activar los 6 motores en el chasis Wild Thumper. Los 3 motores de la izquierda se conectar a uno HB-25 en paralelo, mientras que los 3 motores de la derecha se conectarán al otro. La corriente de cada uno de los 6 motores es de 6,6 amperios, lo que requiere que cada HB-25 proporcione un máximo de 19,8 amperios. El HB-25 puede proporcionar un máximo de 25 amperios, pero vamos a sustituir el fusible incluido por otro de 20 amperios. Esto no es obligatorio, pero te asegura que el HB-25 no proporciona más de 20 amperios en caso de cortocircuito. Si nos damos cuenta que los fusibles saltan con frecuencia, sabremos que los motores están usando más corriente de la máxima especificada y se puede actuar en consecuencia. El componente eléctrico final que usaremos es una placa USB Teensy. Este microcontrolador nos permite generar las señales de control para el HB-25 (servo pulsos PPM), así como la interfaz con sensores y componentes adicionales. Por ahora sólo vamos a realizar el montaje mecánico del dispositivo, pero en la parte 2 analizaremos el dispositivo con mucho más detalle.

Montaje de la Plataforma

El chasis 6WD, una vez montado de acuerdo con las instrucciones del fabricante está listo para alojar los sensores y dispositivos. Montaremos la cámara RGB-D y controladores de motor en la placa del chasis superior. Los controladores de motor se colocan con un par de separadores de aluminio y se fijan con tornillos 6-32. La cámara RGB-D también se puede montar con un par de tornillos 6-32 haciendo dos agujeros pequeños en la base del plástico.

El ODROID-XU, Teensy, módulo LM2596 y los interruptores se montan en el interior de una resistente carcasa. Se trata de una caja de plástico montada en el chasis del vehículo con separadores de aluminio 4 1-1/2" y tornillos 6-32. La altura de los separadores evita que las ruedas traseras rocen la carcasa al conducir sobre terreno escarpado, puesto que la caja es un poco más ancha que la placa de chasis superior (pero más estrecha que la distancia entre los ejes).

Con ODROID-XU integrado, este VTA es capaz de explorar, a la vez que ofrece una capacidad de procesamiento asombrosa.



Ramificación de la potecia electrónica

Asimismo nos aseguraremos que nuestra electrónica está fijada a la caja usando tornillos y separadores, y realizando agujeros en ésta en caso de ser necesario para crear los puntos de anclaje.

La caja de equipamiento nos permite proteger la electrónica de la intemperie, pero tendremos que tener cuidado y asegurarnos que la carcasa se enfría correctamente. Por ahora, nos limitaremos a mantener la tapa abierta hasta que terminemos el montaje y la configuración. Antes de empezar a realizar pruebas reales añadiremos ventiladores de refrigeración para regular la temperatura interna de la caja.

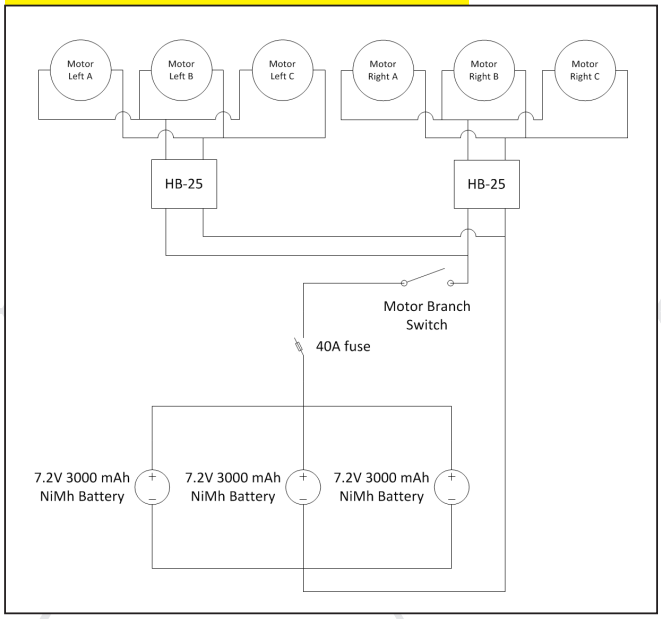
Distribución Energia

Como se ha mencionado anteriormente, estaremos suministrando energia a la electrónica de la caja (regulado por el LM2596) y a los controladores de motor que utilizan filas por separado, de modo que vamos a montar dos interruptores en la carcasa. Esto es importante por dos razones. En primer lugar, queremos asegurarnos que el Teensy genera señales de control antes de que los controladores de motor reciban energía. Si los controladores están alimentados antes de recibir una señal adecuada, no se garantiza un correcto funcionamiento, lo que dará lugar a que los motores cojan energía de forma descontrolada. En segundo lugar, habrá momentos durante el desarrollo en lo que no queramos proporcionar energía al motor (por ejemplo cuando el robot esté sobre tu mesa de trabajo mientras que el ODROID está conectado a un monitor y teclado). El uso de dos interruptores nos permitirá deshabilitar cualquier rama de energía de forma independiente. También vamos a colocar porta fusibles en cada rama de alimentación para evitar cortocircuitos. El fusible de la rama del moto será de 40 amperios, mientras que la rama de la electrónica será de 5 Amp. Los controladores HB-25 tienen cada uno su propio fusible de 20Amp, pero vamos a incluir un fusible principal en cada rama para mayor protección.

Una vez que el regulador LM2596 esté conectado a las pilas de la rama electrónica, fijaremos una tensión de 5V constante antes de conectarlo al ODROID-XU. Además, debemos soldar un conector cilíndrico a los puntos de soldadura de salida del dispositivo para que podamos alimentar el ODROID, y de forma independiente los dispositivos adicionales serán alimentados desde el bus USB. Los Conectores se puede encontrar en algunas tiendas de electrónica, pero el receptor para el ODROID XU es bastante común y puede ser obtener fácilmente de una antigua fuente de alimentación DC. Después de soldar el cable al LM2596, utilice un voltímetro para verificar la polaridad del conector. A continuación, con el voltímetro conectado al conector, ajuste el potenciómetro en el LM2596 hasta que marque una tensión constante de 5,0 voltios. Es muy importante que el nivel de la polaridad y el voltaje se compruebe antes de conectar el ODROID. Una vez que haya verificado la polaridad y el nivel de voltaje, puedes conectar la clavija PSU al ODROID.

En un prototipo de desarrollo anterior, VTA utilizó un Kinect XBOX 360.

Raminifación de la potencia del motor



Conclusión

Este artículo es la primera entrega de nuestra serie de 3 partes sobre robótica impulsada por ODROID. En la Parte 2, nos centrarnos en Linux y Robot Operating System (ROS) e interactuaremos con nuestros dispositivos para controlar los motores y leer los sensores. Sigue el foro de debate de nuestro proyecto en `http://forum.odroid.com`.

CONOCIENDO A AN ODROIDIAN

SIMONE (@SERT00), UN APASIONADO DE ODROID DESDE HACE TIEMPO Y UN EXPERTO INFORMATICO

by Robert Hall

La pesona en si misma, @sert00 se puede encontrar en todas partes dentro del foro.

Por favor, Háblanos un poco sobre ti.

Soy un electricista industrial de 27 años, trabajo como operario de electromecánica en una gran empresa en Italia con más de 2.700 personas. Procesamos y vendemos alimentos, en particular carne de pollo. Mi trabajo consiste en localizar y reparar los problemas de las máquinas que intervienen en la cadena de producción, como sistemas de comunicación, automatización, motores y sistemas neumáticos.

¿Cómo fueron tus inicios con los ordenadores?

Empecé con los ordenadores con 9 años. Mi padre me traía los PCs, impresoras, monitores y periféricos de su trabajo que iban a ser tirados a la basura. Comencé con DOS, luego use Windows Workgroup 3.11.

¿Qué tipos de proyectos has realizado con tu ODROID?

Al principio me compré mi primer ODROID U2 sin pensar en ningún uso. Me encanta el mundo ARM, sigo el desarrollo de Android desde hace algunos años y luego decidí probar una placa basada en Android para ver cómo funcionaba. Mientras tanto, también me pasé a Linux después de usar Windows durante años y pensé que sería un buen comienzo on el U2, ya que mi viejo portátil se rompió.

Luego gané otro U2 como premio mensual, que utilizo sobre todo para recompilar el kernel de Android y mis teléfonos y tablets. También profundice con Linux, usandolo como servidor web y multimedia

El año pasado, recibí una versión beta XU-E (rev. 0.2) como una muestra de ingeniería, luego compré otro XU-E (rev. 0.3) que maneja con más fiabilidad los picos de voltaje de entrada. Con éste he probado de todo, desde la creación de un centro multimedia a reajustar la electrónica. Lo que me gusta de ODROID es el hecho de que puedes probar algo, usarlo y luego cambiar la configuración y dedicarlo a algo diferente.

La última placa que compré fue el U3, que es impresionante. Lamentablemente, no he tenido tiempo para usarla. Sin embargo, mi objetivo será hacer lo que hice con mi viejo Arduino, controlar algunas cosas de mi habitación como las luces y el televisor mediante un APK Java desde una conexión inalámbrica. Deje el proyecto a un lado porque necesitaba Arduino para otras cosas.

Tengo todos los complementos y aparatos que ofrece Hardkernel y en espacial, la pantalla táctil que uso con mi XU-E ejecutando Android desde una eMMC de 64GB. Es una especie de tablet de fabricación casera. De hecho, será mi próximo proyecto cuando tenga algo de tiempo libre, tomando prestadas algunas ideas del artículo de Mauro (de la edición de abril) sobre cómo desarrollar una resistente tablet.

Qué otras aficiones e intereses tienes?

Me gusta caminar por la naturaleza y montar en bicicleta de montaña en la temporada de primavera. Donde yo vivo, hay muchos lugares para esto.

¿Qué es lo que más te gusta de la comunidad ODROID?

Me gusta mucho la amabilidad del personal Hardkernel como Justin, Lisa y Mauro. Todos los miembros del foro están allí para ayudar. Hay algunas personas con mucho talento en la comunidad ODROID.

¿Estás involucrado en cualquier otro proyecto de software o hardware?

Recientemente, he estado centrado en algo diferente que siempre quise aprender. Compré un S7-200, módulo analógico y Ethernet de Siemens, y estoy estudiando la programación de PLC. PLC será útil en mi trabajo y también tengo la intención de usarla para automatizar mi casa junto con mi ODROID-U3 y el Protector E/S.

Una pequeña muestra de la mesa de trabajo de @ sert00. Él también tiene un monitor en la pared y un montón de routers.