

ODROID

Año Dos
Núm. #22
Oct 2015

Magazine

Apache TOMCAT



Tu servidor web y contenedor servlet ejecutándose en la plataforma informática más eficiente del mundo

Plex
Media
Server



Juegos Linux: Emula la última consola de Sega: La Dreamcast



Qué defendemos.

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

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

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

Simple, moderno y único.

De modo que tienes a tu alcance lo mejor



HARDKERNEL



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

Dirección: Max-Pollin-Straße 1
85104 Pförring Alemania

Teléfono & Fax

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

email : service@pollin.de

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





Este mes, presentamos dos servidores que pueden serte muy útiles y que se ejecutan muy bien en la plataforma **ODROID**: **Apache Tomcat** y **Plex Media Server**. **Apache Tomcat** es un servidor web de código abierto y un contenedor de servlets que proporciona un entorno de servidor web **HTTP** de “puro **Java**” para ejecutar código **Java**. Te permite escribir aplicaciones web complejas en **Java** sin necesidad de aprender un lenguaje de servidor específico, como **.NET** o **PHP**. **Plex Media Server** organiza tus colecciones de videos, música y fotos y los transmite a todos a tus pantallas. Nuestros tutoriales te guiarán paso a paso para que puedas instalar este servidor y disfrutar ejecutando un servidor avanzado en casa de un forma eficiente y barata. El reciente lanzamiento de **Lakka** para **ODROID**, una distribución basada en **OpenELEC**, hace que sea más fácil jugar a tus juegos favoritos. **Tobias** analiza el emulador de **Dreamcast**, que es uno de los emuladores de consolas más avanzados para **ODROID**, **Nanik** continúa mostrándonos cómo compilar **Android** para el **ODROID-C1**, **Bruno** describe cómo migrar datos utilizando **LVM**, y aprendemos cómo controlar **ODROID-SHOW** usando **Python**. Como de costumbre, presentamos nuestros queridos juegos de **Linux** que te proporcionarán largas horas de diversión.

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. Para información sobre cómo enviar artículos, contacta con odroidmagazine@gmail.com, o visita <http://bit.ly/lyplmXs>. 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

HARDKERNEL'S EXCLUSIVE NORTH AMERICAN DISTRIBUTOR



All Hardkernel products in stock
at AmeriDroid.com



USB GPS MODULE
\$26.95



ODROID-C1
\$36.95



ODROID-VU
\$119.95



C1 3.2 INCH TOUCHSCREEN DISPLAY
SHIELD
\$26.95

ODROID

Magazine



**Rob Roy,
Editor Jefe**

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



**Robert Cleere,
Editor**

Soy un diseñador de hardware y software que actualmente vive en Huntsville, Alabama. Aunque semi-retirado del diseño de los sistemas integrados, incluyendo más de una década trabajando en el programa del transbordador espacial, continúo diseñando productos de software y hardware, y me interesa la producción de audio/video y las obras de arte. Mis lenguajes de programación son Java, C y C ++, y tengo experiencia con bastantes sistemas operativos integrados. Actualmente, mis proyectos principales son los sistemas navales de seguimiento y control, monitoreo ambiental y la energía solar. Actualmente estoy trabajando con varios procesadores ARM Cortex, pero mi ODROID-C1 es en gran medida el más poderoso de todos



**Bruno Doiche,
Editor
Artístico
Senior**

El otoño en el hemisferio norte significa que el verano se acerca a Brasil, y este es el momento en que nuestros fans patean para mantener a nuestros procesadores fresquitos. No es que mis ODROIDS sufran demasiado. Sin embargo, por desgracia, soy el único que sale con algunas cervezas frías por la piscina

O quizás algún día invente un ODROID submarino robotizado que beba cerveza...



**Nicole Scott,
Editor
Artístico**

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



**James
LeFevour,
Editor
Artístico**

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

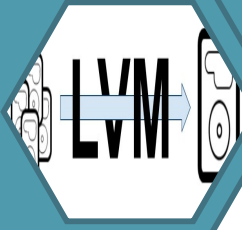


**Manuel
Adamuz,
Editor
Español**

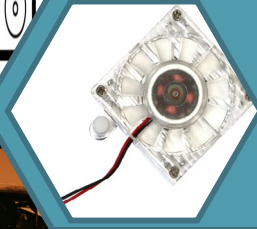
Tengo 31 años y vivo en Sevilla, España, aunque nací en Granada. Estoy casado con una mujer maravillosa y tengo un hijo. Hace unos años trabajé como técnico informático y programador, pero mi trabajo actual está relacionado con la gestión de calidad y las tecnologías de la información: ISO 9001, ISO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



SO DESTACADO: LAKKA - 6



LVM - 8



CONTROL DE VENTILADOR DEL XU4- 9



APACHE TOMCAT- 10



WIKI DE LA COMUNIDAD - 17



SPEEDY NINJA - 16



INSTALADOR PARA PLEX MEDIA - 18



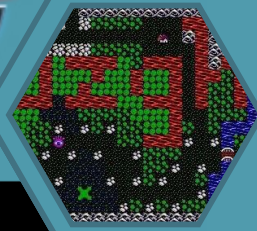
USB-UART EN OSX- 26



DESARROLLO ANDROID - 28



FREEORION- 30



HAXIMA NAZGHUL - 31



SHOWTIME - 32



PRINCE OF PERSIA - 33



JUEGOS LINUX: DREAMCAST - 34



CONOCIENDO A UN ODROIDIAN - 41

SO DESTACADO: LAKKA EN EL ODROID-C1

CONSOLA DE EMULACION RETRO

editado por Rob Roy

Existen grandes imágenes de juegos creadas por la comunidad para las plataformas ODROID como son la ODROID GameStation Turbo basada en Debian y la Pocket Rocket basada en Android. La última imagen de juegos que ha sido liberada incluye un sistema operativo multiplataforma de código abierto basado en OpenElec llamado Lakka, que utiliza el popular software RetroArch para emular diferentes tipos de juegos. Recientemente ha sido exportada al ODROID-C1 con la intención de facilitar su instalación, configuración y uso. Soporta los siguientes sistemas:

Atari 2600
Atari Jaguar
Atari Lynx
Cave Story
Dinothawr
Doom
FB Alpha
FFmpeg
Game Boy
Game Boy Advance
Game Boy Color
Master System

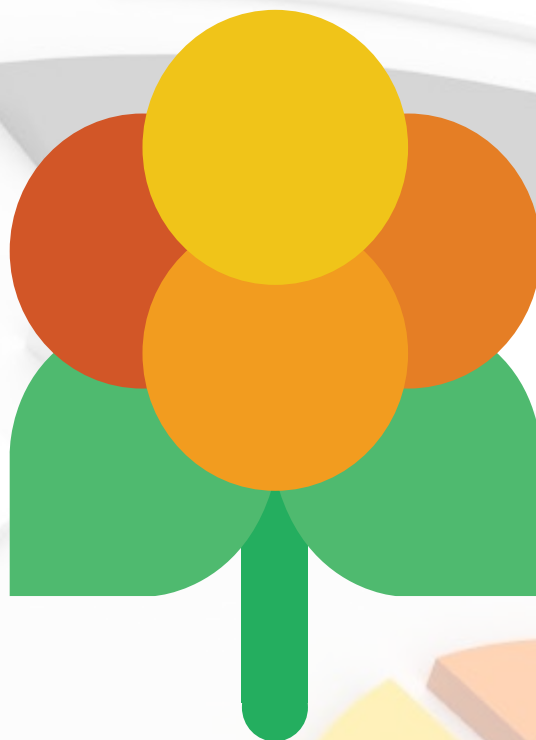
Mega Drive
Nintendo Entertainment System (NES)
Neo Geo Pocket
PCEngine
PlayStation
PlayStation Portable (PSP)
Sega 32X
Super Nintendo Entertainment System (SNES)
Vectrex

Una de la ventajas de Lakka es que detecta automáticamente muchos tipos de mandos pre-configurados, incluyendo los de la Xbox 360, PS3 / PS4, Saitek, Logitech y Zeemote.

Empecemos

Para instalar Lakka, descarga la imagen pre-compilada para ODROID-C1 desde <http://bit.ly/1YIOrvw> en un sistema Linux. Descomprime el archivo, luego determina el nombre de dispositivo para la tarjeta SD accediendo al listado de unidades y particiones actuales:

```
$ ls -l /dev/sd*
```



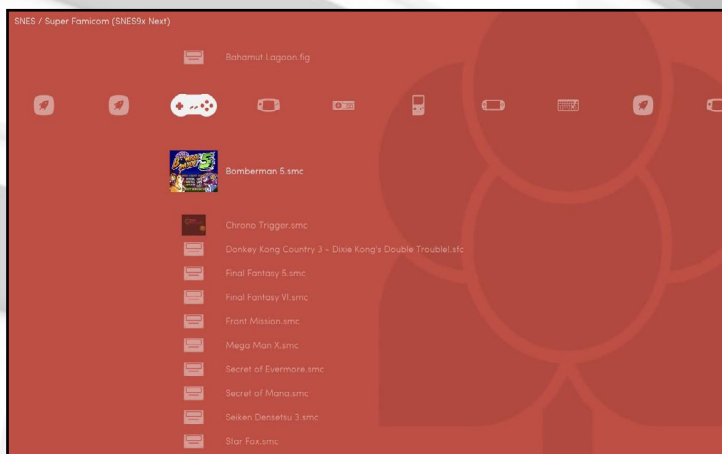
```
brw-rw---- 1 root disk 8, 0 22 mars 23:01 /dev/sda
brw-rw---- 1 root disk 8, 1 22 mars 23:01 /dev/sda1
brw-rw---- 1 root disk 8, 2 22 mars 23:01 /dev/sda2
brw-rw---- 1 root disk 8, 3 22 mars 23:01 /dev/sda3
brw-rw---- 1 root disk 8, 4 22 mars 23:01 /dev/sda4
brw-rw---- 1 root disk 8, 5 22 mars 23:01 /dev/sda5
brw-rw-r-- 1 root users 8, 16 22 mars 23:01 /dev/sdb
```

Las entradas que terminan en números son las particiones y el resto son las unidades. En este ejemplo, sda es el disco duro principal y desde la sda1 a la sda5 son sus particiones. Inserta una tarjeta microSD vacía en el equipo y escribe el comando nuevamente:

```
$ ls -l /dev/sd*
```

```
brw-rw---- 1 root disk 8, 0 22 mars 23:01 /dev/sda
brw-rw---- 1 root disk 8, 1 22 mars 23:01 /dev/sda1
brw-rw---- 1 root disk 8, 2 22 mars 23:01 /dev/sda2
brw-rw---- 1 root disk 8, 3 22 mars 23:01 /dev/sda3
brw-rw---- 1 root disk 8, 4 22 mars 23:01 /dev/sda4
brw-rw---- 1 root disk 8, 5 22 mars 23:01 /dev/sda5
brw-rw-r-- 1 root users 8, 16 22 mars 23:49 /dev/sdb
brw-rw---- 1 root disk 8, 17 22 mars 23:49 /dev/sdb1
brw-rw---- 1 root disk 8, 18 22 mars 23:49 /dev/sdb2
```

Observa que sdb incluye una o más particiones, que en este ejemplo aparecen como sdb1 y sdb2. Esto significa que sdb representa el lector de tarjetas SD, aunque en tu sistema podría tener una letra diferente. Asegúrate de utilizar la letra correspondiente a tu unidad en el resto del tutorial.



Menu de Lakka

Grabar la imagen

Ahora que conoces la unidad de tu tarjeta SD, ve al directorio donde extrajiste Lakka y graba la imagen en la tarjeta, colocando la letra correcta de tu unidad en sdx:

```
$ sudo dd if=Lakka-*.img of=/dev/sdX
```

Pasaran unos minutos hasta que vuelva aparecer el prompt. Una vez completado el proceso, retira tu tarjeta SD y pasa al siguiente paso.

Primer arranque

Para ejecutar Lakka, sigue estos pasos:

- Inserta la tarjeta microSD en el ODRROID-C1**
- Conecta un cable HDMI entre tu ODRROID y tu televisor**
- Enciende el televisor**
- Conecta el cable Ethernet al ODRROID-C1 (opcional)**
- Conecta uno de los mandos compatibles en uno de los 4 puertos USB del ODRROID**
- Conecta la fuente de alimentación al ODRROID-C1**

Deberías ver la pantalla de Lakka. El paquete ampliará automáticamente el sistema de archivos y reiniciará el C1 a los 30 segundos. Esto sólo ocurre con el primer arranque, los arranques posteriores deberían ser bastante más rápidos. Si todo ha ido bien, ahora deberías poder navegar por el menú de Lakka, nuestra interfaz gráfica, como muestra la Figura de arriba. ¡Felicidades, has instalado correctamente Lakka!

Ejecutar los juegos

Inserta una unidad USB con las ROMs que te gustaría ejecutar. Tu unidad USB debe estar formateada en FAT o NTFS. La partición se montará automáticamente en una nueva carpeta bajo /storage/roms/ y tus ROMs aparecerán en el menú de Lakka.

Manufacturer	Model	BIOS	Additional info
3DO	3DO (4DO)	panafz10.bin	
Atari	Atari 2600 (Stella)	'not needed'	
Atari	Atari 7800 (ProSystem)	ProSystem.dat	Atari 7800 Database
Atari	Atari 7800 (ProSystem)	7800 BIOS (U).rom	Atari 7800 BIOS
Atari	Lynx (Handy)	lynxboot.img	(Lynx Boot Image)
Id Software	Doom (PrBoom)	prboom.wad	(PrBoom WAD) Need to be near your Doom wads
Magnavox	Odyssey2	o2rom.bin	(Odyssey 2 BIOS)
NEC	PC-FX	pcfx.bios	
NEC	PC Engine/PCE-CD	syscard3.pce	(PCE-CD BIOS)
Nintendo	NES (Nestopia)	disksys.rom	(Famicom Disk System BIOS)
Nintendo	N64 (Mupen64Plus)	'not needed'	
Nintendo	Game Boy Advance	gba_bios.bin	(GBA BIOS)
Sony	PS (Beetle PSX)	scph5500.bin	(PS1 JP BIOS)
Sony	PS (Beetle PSX)	scph5501.bin	(PS1 US BIOS)
Sony	PS (Beetle PSX)	scph5502.bin	(PS1 EU BIOS)
Sega	MegaCD	bios_CD_E.bin	(MegaCD EU BIOS)
Sega	SegaCD	bios_CD_U.bin	(SegaCD US BIOS)
Sega	MegaCD	bios_CD_J.bin	(MegaCD JP BIOS)
Sega	Saturn	saturn_bios.bin	(Saturn BIOS)
SNK	NeoGeo	neogeo.zip	(NeoGeo BIOS) MUST BE PLACED IN ROMS FOLDER

Tabla BIOS de Lakka

Algunos núcleos libretro requieren de una BIOS para ejecutarse. Necesitas encontrar esas BIOS por ti mismo, ya que es ilegal facilitarlas. Esas BIOS deben colocarse en la carpeta "system". La imagen anterior muestra los diferentes archivos BIOS que son necesarios para emular cada tipo de sistema. Recuerda que Linux es un sistema sensible a las mayúsculas y minúsculas, por lo que necesitarás cambiar el nombre de los archivos BIOS para que coincidan con los de esta tabla, para que así Lakka sea capaz de encontrarlos. En la Figura de abajo se detallan las extensiones de los archivos utilizados por los distintos emuladores.

Para más información o enviar comentarios, preguntas o sugerencias sobre Lakka, puedes visitar la página web de Lakka en <http://www.lakka.tv> o los foros Libretro en <http://bit.ly/1P09vcs>.

System	Model	Filename Extension	Additional info
3DO	3DO (4DO)	.iso .cue	
Atari	Atari ST/STE/TT/Falcon	.st .msa .zip	
Atari	Atari 2600	.a26 .bin	
Atari	Atari 7800 (ProSystem)	.a78 .bin	
Atari	Lynx (Handy)	.lnx	
Id Software	Quake	.pak	
MSX	MSX (fMSX)	.rom .mx1 .mx2	
Magnavox	Odyssey2	.bin	
NEC	PC-FX	.cue .ccd .toc	
NEC	PC Engine/PCE-CD	.pce .cue .ccd	
Nintendo	SNES / Super Famicom	.sfc .smc .bml	
Nintendo	NES / Famicom (FCEUmm)	.fds .nes .unif	
Nintendo	N64	.n64 .v64 .z64 .bin .u1	
Nintendo	Game Boy Advance	.gba .agb .bin	
PSX	PS (Beetle PSX)	.bin .cue .toc .m3u .ccd .exe .mdf .cbn .pbp	
Sega	MS/GG/MD/CD +GX	.mdx .md .smd .gen .bin .cue .iso .sms .gg .sg	
Sega	Sega Saturn	.bin .cue .iso	
Sega	Sega Saturn	.bin .cue .iso	

Tabla de extensiones de las ROM

GESTION DE VOLUMENES LOGICOS

HAZ QUE TU MIGRACION DE DATOS SEA MAS SENCILLA CON LVM

por David Gabriel



Supongamos que acabas de comprar un nuevo ODROID-XU4 a estrenar y deseas migrar a éste todos los servicios que tienes ejecutándose en un viejo ODROID. Lo tienes todo configurado, el tener que instalar y configurar todo de nuevo, crear los volúmenes lógicos, los sistemas de archivos y ajustar los permisos te llevaría demasiado tiempo. Además, necesitas transferir todos los archivos al nuevo ODROID que, dependiendo del volumen, te llevará más o menos tiempo.

Tener todos tus archivos configurados bajo una unidad externa utilizando LVM puede hacer que todo este proceso de migración sea mucho más sencillo y necesitas tan sólo unos minutos. ¿Cómo? Exportando tu grupo de volúmenes del sistema antiguo e importándolo al nuevo sistema.

Ten en cuenta que todos los comandos se deben ejecutar con privilegios root. En primer lugar, tendrás que detener todos los servicios ejecutados en ese grupo de volúmenes:

```
# service <daemon name> stop
```

Luego, desmonta los sistemas de archivos adjuntos al grupo de volúmenes. En mi caso, sólo tengo /home

```
# umount /home
```

A continuación, tienes que desactivar

los volúmenes lógicos (LVs) que tiene el grupo de volúmenes que vas a exportar. Primero debe comprobar el estado actual:

```
# lvscan
```

El comando anterior te dará el estado de los VLs, mostrándolos bajo /dev/<nombre_gv>. Es similar a la estructura /dev/mapper/<nombre_gv>-<nombre_vl > que vimos en el artículo anterior. Ambos son vínculos al archivo de bloque lvm en /dev/dm-x. Para desactivar el volumen lógico, escribe:

```
# lvchange -a n \
/dev/rootvg/homelv
```

Si tienes más de uno, simplemente pégalos uno tras otro, separados por espacios. Si ejecutas lvscan de nuevo, verás que los VLs cambian de activos a inactivos. Una vez que todos los VLs del GV estén inactivos, puedes exportar el GV:

```
# vgexport rootvg
```

Después, realiza una comprobación final ejecutando vgscan, aparecerá que el grupo de volúmenes está ahora exportado. Llegados a este punto, puedes quitar la unidad de tu viejo sistema y conectarla al nuevo. A continuación, ejecuta pvscan, deberías ver todas tus particiones en el nuevo sistema.

Para importar tus datos, escribe:

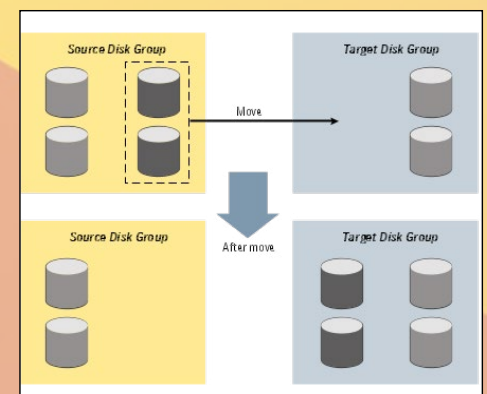
```
# vgimport rootvg
```

Esto debería permitirte ver tus volúmenes lógicos. Ahora, simplemente monta el sistema de archivo de nuevo:

```
# mount /dev/rootvg/homelv
```

Esto debería restaurarlo todo en el nuevo ODROID sin perder tiempo copiando o volviendo a crear toda la estructura que ya tenías. Por supuesto, aún tienes que reinstalar el software que no se almacena en el LVM.

Ahora ya sabes cómo migrar tus datos entre diferentes sistemas aprovechándote de las ventajas de LVM. Espero que te sea de mucha utilidad y ahorres tiempo cuando quieras mover tus archivos de un equipo a otro.

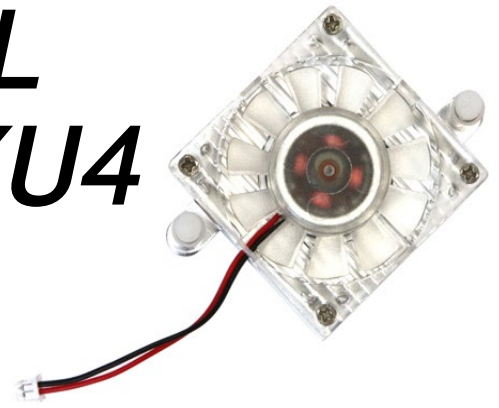


Mover volúmenes entre grupos de discos es muy sencillo

CONTROL SOBRE EL VENTILADOR DEL XU4

UNA FORMA DE GESTIONAR EL CONSUMO DE ENERGIA Y LA TEMPERATURA DEL XU4

por @Grotus



El driver del ventilador del ODROID utiliza la modulación por impulsos (PWM) para controlar la velocidad del ventilador, ajustando el ciclo de trabajo de la PWM en base a la temperatura de la CPU. El driver tiene cuatro velocidades, que se activan en función de tres valores de temperatura. De modo que, si la temperatura actual de la CPU es inferior al valor de temperatura definido como el más bajo se activa la primera velocidad del ventilador, cuando la temperatura se encuentra entre el primer y segundo valor se activa la segunda velocidad del ventilador, cuando la temperatura se encuentra entre el segundo y tercer valor se activa la tercera velocidad, y cuando la temperatura supera el tercer valor se activa la cuarta velocidad del ventilador.

Se pueden realizar una serie de ajustes en sysfs en relación al driver del ventilador del ODROID, en el XU4 esto está en `/sys/devices/ODROID_fan.13`, mientras que en el XU3 está en `/sys/devices/ODROID_fan.14`.

Los ajustes son:

fan_mode: Auto o Manual (fijar **1** para auto y **0** para manual, por defecto Auto)

fan_speeds: Cuatro valores en porcentaje que definen la velocidad del ventilador delimitados por espacios, en orden ascendente. (por defecto "1 51 71 91")

pwm_duty: Configuración actual del ciclo de trabajo de la PWM (0-255, se ajusta dinámicamente)

pwm_enable: On o Off (por defecto On)

temp_levels: Tres valores de temperatura de la CPU en Celsius delimitados por espacios, en orden ascendente (por Defecto "57 63 68")

Existen dos forma de obtener la temperatura de la CPU en sysfs: `/sys/devices/10060000.tmu/temp` y `/sys/devices/virtual/thermal/thermal_zone0/temp`. La primera sólo la puede leer root o un usuario del grupo root y muestra las temperaturas de cinco sensores. El segundo es legible por todos los usuarios y proporciona una única temperatura. En ambos casos, las temperaturas están en miligrados Celsius, por lo que tendrás que dividir las por 1.000 para obtener Celsius. El valor más alto en `/sys/devices/10060000.tmu/temp` se utiliza para controlar la ve-

locidad del ventilador.

Los ajustes de la velocidad del ventilador se especifican en porcentajes, y deben estar en el rango de 0-100. El ciclo de trabajo PWM se especifica en el rango de 0 a 255, en el modo automático se calcula multiplicando el valor de velocidad del ventilador por 255 y dividiendo por 100. Por ejemplo, en el caso por defecto, cuando la temperatura alcanza los 57 grados, el ventilador gira al 51%, lo que equivale al ciclo de trabajo PWM de $51 \cdot 255 / 100 = 130$

El script que controla el ventilador funciona ajustando el `fan_mode` a manual y cambiando el `pwm_duty` al valor deseado basándose en la temperatura. El script tiene definidos 9 niveles para el ventilador en contraposición a los 4 que tiene el driver del ventilador del ODROID. Para configurar el modo automático en el ventilador, puedes aplicar la nueva configuración a `fan_speeds` y `temp_levels` y tendrán efecto inmediato.

Ejemplo

Este es un ejemplo para que el ventilador se conecte al 20% a los 50°C, gire al 50% a los 70 °C, y alcance 95% a los 80°C en el ODROID-XU4:

```
$ sudo echo "1 20 50 95" > /sys/devices/ODROID_fan.13/fan_speeds
$ sudo echo "50 70 80" > /sys/devices/ODROID_fan.13/temp_levels
```

Configurar los valores de esta forma no se mantendrán si se reinicia. Para que los ajustes se apliquen en el arranque, se puede fijar una regla para udev, creando un archivo en `/etc/udev/rules.d` con los ajustes deseados. Usé `60-ODROID_fan.rules` como nombre en mi sistema. El siguiente debería funcionar tanto en el XU3 como en el XU4 ya que se basa en el nombre del driver del ventilador del ODROID, en lugar del nombre del kernel que es diferente en las dos versiones:

```
DRIVER=="ODROID-fan", ACTION=="add", ATTR{fan_speeds}="1 20 50 95", ATTR{temp_levels}="50 70 80"
```

Para enviar comentarios, preguntas o sugerencias, puedes visitar el hilo original en <http://bit.ly/1jit0Rx>.

APACHE TOMCAT

UN PODEROSO SERVIDOR PARA APPLLET Y PAGINAS WEB BASADO EN JAVA

por Andrew Ruggeri

A Apache Tomcat, o simplemente Tomcat es un servidor web o contenedor web HTTP de código abierto creado en 1998. Tomcat es un programa multiplataforma escrito en Java, que esta mantenido activamente por la Apache Software Foundation. Tomcat se utiliza para ejecutar programas Java especiales tales como Servlets o JavaServer Pages (JSP), que son conocidos como aplicaciones web (o apps web).

Una descripción muy simple de Tomcat es que se trata de un servidor web: lo que significa que cuando recibe una solicitud de un ordenador, devuelve una página web. Esta página web se crea a partir de un programa conocido como webapp, escrito en Java y ejecutado por Tomcat.

Esta guía está pensada para que sea fácil de utilizar, y está dirigida a alguien que está pensando en empezar con aplicaciones web Java. A continuación se describen los pasos básicos necesarios para instalar Tomcat en un ODROID-C1, configurar Tomcat para ejecutar un simple servlet y por último, crear un sencillo servlet/webapp que muestre la temperatura de la CPU del C1. Aunque esta guía está escrita para el ODROID-C1, se pueden aplicar pasos similares en otros dispositivos.

Instalar Tomcat

Hay varias formas de instalar Tomcat en el C1, los tres más populares son instalar y compilar la fuente, “apt-get” y tener que ejecutarlo de forma independiente. Una rápida búsqueda apt-cache para “Tomcat” muestra que Tomcat 7 se encuentra disponible en los repositorios por defecto del C1. Para simplificar las cosas, haremos una simple instalación desde apt-get usando los siguientes comandos:

```
sudo apt-get install tomcat7
```

[Opcional] La documentación se puede descargar con:

```
sudo apt-get install tomcat7-docs
```

[Opcional] Se puede descargar varios ejemplos Tomcat con

```
sudo apt-get install tomcat7-examples
```

Aunque la versión más reciente de Tomcat es la 8, ésta está disponible a través de otros repositorios o compilando la fuente. Para mantener esta guía a un nivel para principiantes, me centraré sólo en Tomcat 7 instalándolo desde los repositorios por defecto. Si desea instalar Tomcat 8 desde la fuente, puedes aún así, seguir esta guía ya que los pasos para la instalación no varían.

Ejecutar y probar

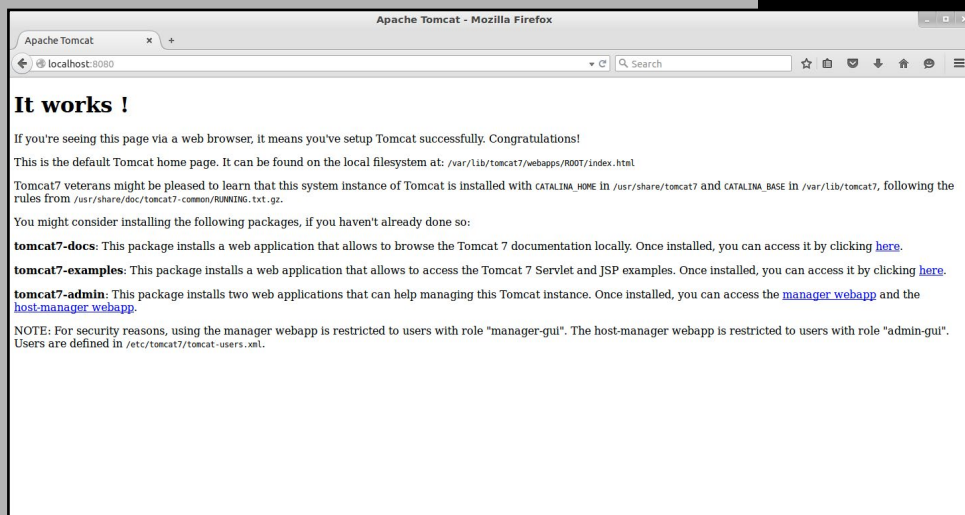
Una vez instalado, Tomcat se ejecutará como un servicio y se puede iniciar, reiniciar o detener con los siguientes comandos.

```
sudo service tomcat7 start
sudo service tomcat7 restart
sudo service tomcat7 stop
```

La forma más sencilla de comprobar si Tomcat7 está funcionando correctamente es ver si se carga la página web de prueba de Tomcat. Abre un navegador web (Firefox, Chromium, ect) en el C1. En la barra de direcciones escribe:

```
localhost:8080
```

Veamos lo que significa la dirección: en primer lugar, con localhost le estamos diciendo al navegador que mire en el equipo local en el cual se está ejecutando (al igual que ocurre cuando escribes google.com, le estás diciendo al navegador que mire en el equipo en el cual se está ejecutando google.com). '8080' es el puerto Tomcat en el cual se reciben las peticiones (8080 es el predeterminado). Este valor se puede cambiar y se trata en la sección de configuración opcional de este artículo. Si todo funciona correctamente, se debería cargar la siguiente página web.



Apache Tomcat - Mozilla Firefox

localhost:8080

It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat7/webapps/ROOT/index.html`

Tomcat7 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat7` and `CATALINA_BASE` in `/var/lib/tomcat7`, following the rules from `/usr/share/doc/tomcat7-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

- tomcat7-docs:** This package installs a web application that allows to browse the Tomcat 7 documentation locally. Once installed, you can access it by clicking [here](#).
- tomcat7-examples:** This package installs a web application that allows to access the Tomcat 7 Servlet and JSP examples. Once installed, you can access it by clicking [here](#).
- tomcat7-admin:** This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the [manager webapp](#) and the [host-manager webapp](#).

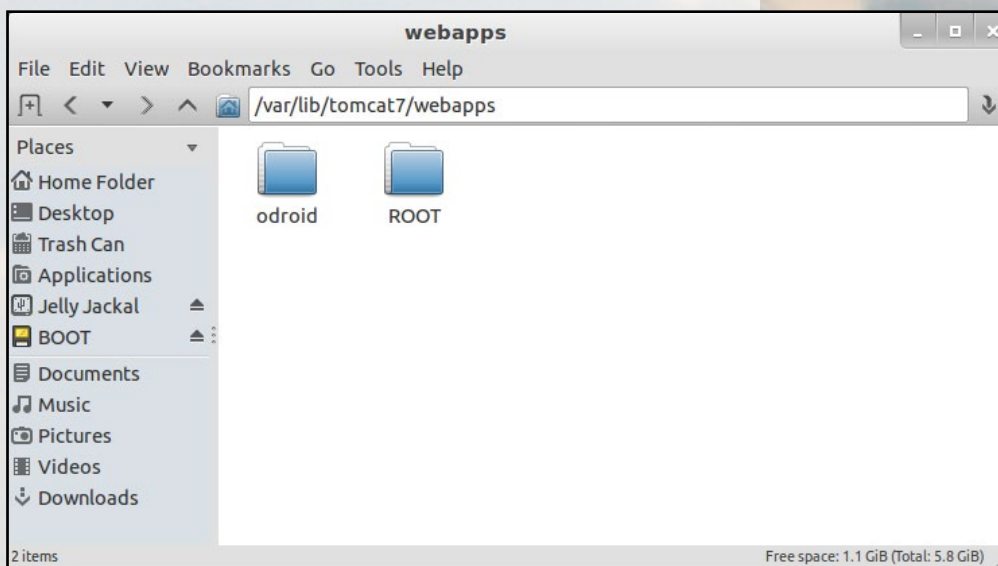
NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat7/tomcat-users.xml`.

Siempre que instales Tomcat, esta página confirma que se está ejecutando correctamente.

Configuración [No-Opcional]

Después del apt-get install, tenemos que configurar algunas cosas para conseguir que Tomcat se ejecute con el servlet que vamos a crear. Lo que vamos a hacer es decirle a Tomcat lo que tiene que hacer cuando reciba una petición web, o una solicitud HTTP como la que usaremos.

Ve a /var/lib/tomcat7/webapps/ ya sea con un explorador de archivos o con Terminal. Debería haber una carpeta: 'ROOT', esta es la página web por defecto que vimos durante la prueba. Ahora vamos a preparar el lugar para nuestra nueva página. Empieza por crear una carpeta llamada "odroid" (puede que necesites privilegios de root) en el directorio webapps, de modo que ahora debería tener dos carpetas una junto a la otra.



Aquí en webapps es donde se van a colocar tus páginas web

Ahora ve a la carpeta odroid (/var/lib/tomcat7/webapps/odroid/) y crea una carpeta llamada "WEB-INF". Una vez más, muévete a la carpeta "WEB-INF" y crea una carpeta llamada "classes". Deberías haber creado un total de 3 carpetas (marcadas en negrita) con las siguientes rutas:

```
/var/lib/tomcat7/webapps/odroid/  
/var/lib/tomcat7/webapps/odroid/WEB-INF/  
/var/lib/tomcat7/webapps/odroid/WEB-INF/classes/
```

odroid: Esta carpeta y cualquier otra carpeta dentro de la carpeta "webapps" (como ROOT) es conocida como "directorio base de documentos". Esta es la carpeta donde se deben colocar los archivos auxiliares a la webapp, como son las imágenes, archivos javascript, CSSs o archivos HTML adicionales.

WEB-INF: Todo "directorio base de documentos" contiene esta carpeta. Dentro de cada una de estas carpetas, verás un directorio 'classes' así como un archivo llamado "web.xml". Mas

adelante analizaremos con más detalle los archivos “web.xml”.

classes: Esta es la carpeta que contendrá los archivos servlets de Java compilados.

Ahora que tenemos todos los directorios en su lugar, ve a la carpeta WEB-INF/Varr/lib/tomcat7/odroid/WEB-INF/. Abre un editor de archivos (gedit, kate, nano, vim, etc.) y crear un nuevo archivo XML llamado “web.xml”. Este es el archivo que dirá a Tomcat que servlet debe ejecutar cuando reciba una petición web. El contenido del archivo web.xml es el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app version="3.0" xmlns="http://java.sun.com/
xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://java.
sun.com/xml/ns/javaee http://java.sun.com/xml/ns/ja-
vae/web-app_3_0.xsd">

    <servlet>
        <servlet-name>odroidTemperature</servlet-
name>
        <servlet-class>temperatureServlet</servlet-
class>
    </servlet>

    <servlet-mapping>
        <servlet-name>odroidTemperature</servlet-
name>
        <url-pattern>/temperature</url-pattern>
    </servlet-mapping>

</web-app>
```

El encabezado de este documento xml es estándar, verás esas mismas líneas en cada archivo web.xml, son necesarias para configurar los espacios de nombres XML. Las dos partes del medio son en las que nos vamos a centrar ya que son las específicas de cada página web servlet que configuremos. El elemento servlet siempre tiene que ir delante del elemento servlet-mapping. Los elementos servlet describen el servlet que generará el html y servlet-mapping describe la ruta url que activará este servlet. Podemos considerarlo como un sistema de dos partes. Si miras dentro de “servlet” y “servlet-mapping” te darás cuenta de que ambos contienen “servlet-name” y tienen el mismo valor para “odroidTemperature”. Esto se debe a que el servlet-name se conecta al elemento “servlet” y al elemento “servletmapping” a la vez. En el elemento servlet existe algo llamado “ervlet-class”, este es simplemente el nombre del servlet que vamos a crear (crearemos más adelante un archivo llamado “temperatureServlet.java”). Por último existe “url-patern” que se encuentra en el elemento servlet-mapping. Este dirá a Tom-

cat que espere cualquier url con /temperature al final y si la encuentra, la enviará al servlet llamado temperatureServlet.

Configuración [Optional]

Tomcat es muy configurable, aunque funciona directamente con una simple instalación. Si deseas personalizar Tomcat, puedes hacerlo editando algunos archivos xml en /etc/tomcat7 que son context.xml, server.xml, web.xml. Para editarlos utiliza cualquier editor de texto (vim, gedit, nano, etc.). Contienen muchos parámetros que se pueden cambiar. A continuación se recoge una breve descripción de cada uno de estos archivos xml. Echa un vistazo a los archivos, o consulta la documentación de Apache para conseguir información más detallada.

server.xml : Cambia Tomcat en sí mismo. Echa un vistazo a todos posibles cambios que puedes realizar. Este es el archivo en el que puede cambiar el puerto 8080 por defecto de Tomcat. Ten en cuenta que para la depuración es recomendable utilizar puertos superiores al 1024.

context.xml : Cambia el comportamiento de Tomcat. Puedes tener a Tomcat actualizando automáticamente una página web para ver los cambios del código. Esto es muy útil durante la depuración, pero debe desactivarse para un uso normal ya que provoca una sobrecarga innecesaria.

web.xml : Las propiedades por defecto utilizadas para todas las aplicaciones Web.

Crear un servlet

¿Qué es un servlet de Java? Podemos simplificarlo diciendo que es un programa Java que acepta la información que le envía un navegador web y responde a ésta con HTML. Como ejemplo vamos a crear un servlet que active odroidTemp (más fácil que odroid_Temperature). Cuando reciba una petición desde un navegador, creará una web con la temperatura del ODROID.

Para empezar con nuestro servlet crearemos un archivo Java en la carpeta 'classes'. Abre un editor de texto y crear un nuevo archivo con el nombre "temperatureServlet.java". Al guardar el archivo asegúrate de colocarlo en la ubicación: /var/lib/tomcat7/webapps/odroid/WEB-INF/classes/.

```
import java.io.*;
// From /usr/share/tomcat7/lib/servlet-api.jar
import javax.servlet.*;
import javax.servlet.http.*;

public class temperatureServlet extends HttpServlet {

    @Override
    public void doGet(HttpServletRequest request,
```





```

HttpServletResponse response) throws IOException,
ServletException {

    // MIME type
    response.setContentType("text/html");

    PrintWriter htmlResponse = response.getWrit-
er();
    try {
        // Get the Temperature
        String TemperatureValue;
        BufferedReader br = new
BufferedReader(new FileReader("/sys/devices/virtual/
thermal/thermal_zone0/temp"));
        // 1 line file with current temp
        TemperatureValue = br.readLine();
        // Clean it up a bit
        TemperatureValue = TemperatureValue ==
null ? "NA" : TemperatureValue.substring(0,2);

        // HTML TIME
        // Open
        htmlResponse.println("<html>");
        htmlResponse.println("<head><title>Odroid
Temperature</title></head>");
        htmlResponse.println("<body>");

        // Show Temperature
        htmlResponse.println("<h1>Odroid Tempura-
ture</h1>");
        htmlResponse.println("<p>Temperature C: "
+ TemperatureValue + "</p>");

        // Close
        htmlResponse.println("</body>");
        htmlResponse.println("</html>");

    } finally {
        // Close the writer and we're finished
        htmlResponse.close();
    }
}

```

Deberías tener conocimientos básicos sobre Java para poder captar la esencia de lo que sucede en este código. Sino, no se preocupe, hay muchas y excelentes guías de java para principiantes. Sin profundizar demasiado en los conceptos básicos de java, me gustaría explicar lo que está sucediendo en el servlet. Este servlet anula la función doPost, que es la que intercepta una solicitud HTTP POST (del mismo modo que hay una

getPost que intercepta una solicitud HTTP GET). La respuesta que este servlet devuelve es una cadena de HTML que se forma en el bloque try, y está determinada por el type MIME.

El siguiente paso es compilar el código para ser utilizado por Tomcat, y lo vamos a hacer directamente desde el terminal. Abre el terminal, ve al directorio donde tienes el archivo temperatureServlet.java y ejecuta los siguientes comandos.

```
$ javac -target 1.7 -source 1.7 -cp ./usr/share/\
tomcat7/lib/servlet-api.jar temperatureServlet.java
```

El comando javac invoca el compilador java que cogerá nuestro código fuente de Java y lo compilará en un programa que pueda ejecutar Tomcat. Añadimos los argumentos 'target' 1.7 y "source" 1.7 para decirle al compilador que compile para Java 1.7. Esto se hace porque Tomcat7 se ejecutará con JVM 1.7, pero al invocar javac solo compilará para Java 1.8 provocando problemas de compatibilidad. La tercera parte del comando que observas es "-cp ./usr/share/tomcat7/lib/servlet-api.jar" que hace que el compilador de java utilice el servlet-api.jar para ayudar a desarrollar el programa temperatureServlet.java. Tenemos que añadir esto porque estamos, como ya sabe, creando un servlet java que necesita ayuda de este archivo servlet-api.jar externo.

Ejecutar el servlet

Antes de que comprobamos el servlet, primero tenemos que hacer una rápido chequeo de la estructura de archivos de la webapp odroid (una comprobación rápida puede salvarte de un gran quebradero de cabeza). La estructura de carpetas debe coincidir con la siguiente:

```
Webapps\
  odroid\
    WEB-INF\
      web.xml
      classes\
        temperatureServlet.java
        temperatureServlet.class
```

¡Muy bien! Ahora es el momento de probar y ver si todo funciona. Si aún no lo has hecho, reinicia Tomcat con el comando "sudo service tomcat7 restart". Ahora abre un navegador y carga la página "localhost:8080\odroid\temperature". Si todo ha ido bien, debería haber cargado una página.

¡Enhorabuena! eso ha sido todo por esta guía. Como te puedes imaginar, hay mucha más información sobre este tema, te recomiendo encarecidamente que utilices esta guía como punto de partida y uses ejemplos y guías de Apache: ver tomcat.apache.org/tomcat-7.0-doc/index.html para más información.



WIKI DE LA COMUNIDAD

CONTRIBUYE A AMPLIAR LA BASE DE CONOCIMIENTO DE ODROID

por Rob Roy

Hardkernel ha puesto en marcha recientemente un gran recurso para los ODROIDians que deseen aportar sus conocimientos a una wiki de la comunidad, disponible en <http://wiki.odroid.in>. Está

hecha con la intención de complementar la wiki oficial de Hardkernel de <http://bit.ly/1R6DogZ>, útil para que publiques tus consejos, enlaces a imágenes de la comunidad, proyectos y cualquier otra cosa que pueda ser beneficiosa para la comunidad de Hardkernel.

Si desea participar, haz clic en el botón "Request Account" en la parte superior derecha, e incluye tu nombre de usuario del foro ODROID en la sección "Personal Biography". Para comentarios, preguntas y sugerencias relacionadas con la nueva wiki, por favor visite el hilo del foro original en <http://bit.ly/1QDMNoT>.



HARDKERNEL



Page Discussion

Main Page

Hardkernel ODROID Unofficial Wiki (Community Supported)

[Official Hardkernel Wiki](#)

This wiki requires account activation. During the register process please

This page was last modified on 15 September 2015, at 17:40.

Welcome to the ODROID Support Page

This place is for the ODROID boards.

About

History

The ODROID means Open + Droid. It is a development platform for the hardware as well as the software.

Here is a brief history of ODROID.

- ODROID : The world first Android mobile game console development platform with S5PC100 (2009' Fall)
- ODROID-T : The world first Android 10.1" tablet development platform with Exynos3110 (2010' Spring)
- ODROID-S : An affordable Mobile development platform with Exynos3110 (2010' Summer)
- ODROID-7 : E-Book/CNS development platform with Exynos3110 (2010' Fall)
- ODROID-A : The world first Dual-core & 3G modem integrated tablet development platform with Exynos4210 (2011' Spring)
- ODROID-PC : Internet TV and Smart Set-top box development platform with Exynos4210 (2011' Winter)
- ODROID-A4 : Palm sized Handheld Mobile & Media player development platform with Exynos4210 (2012' Spring)
- ODROID-Q : The world first ARM Quad-Core integrated tablet development platform with Exynos4412 (2012' Summer)
- ODROID-X : The world lowest cost ARM Quad-Core development board with Exynos4412 (2012' Summer)
- ODROID-X2 : The upgrade version of ODROID-X with 1.7GHz Exynos4412 Prime and 2GB RAM (2012' Fall)
- ODROID-U2 : The upgrade version of ODROID-U with 1.7GHz Exynos4412 Prime and 2GB RAM (2012' Winter)
- ODROID-XU3 : The world lowest cost ARM Octa-Core big.LITTLE board computer with Exynos5410 (2013' Summer)
- ODROID-U3 : The upgrade version of ODROID-U2 with 1.7GHz Exynos4412 Prime and 2GB RAM (2013' Winter)
- ODROID-XU3 : The world's first HMP enabled ARM Octa-Core big.LITTLE board computer with Exynos5422 (2014' Summer)

Table of Contents

- Welcome to the ODROID Support Page
- About
- History
- Getting Started
- References

DIVERSION CARGADA DE ADRENALINA

SPEEDY NINJA, EL NUEVO E INCANSABLE CORREDOR QUE ESTABAS BUSCANDO

por Bruno Doiche



En la oficina de diseño de la revista, junto a mi fiel clúster ODROID, dispongo de una buena combinación de cerveza y café y un montón de artículos, me gusta probar todos los corredores que encuentro. Entre las docenas a los que he jugado, Speedy Ninja sin duda merece tu atención. Exige tanto reflejos como capacidad de pensar, hay que estar siempre al tanto de lo que pasa por ambos lados para atrapar tantas monedas como puedas. ¿La recompensa? ¡Un dragón Super divertido paseando al estilo ninja!

https://play.google.com/store/apps/details?id=com.netease_na.nmd2

¡Para cada movimiento y hazaña lograda, un increíble viaje con el dragón!



PLEX MEDIA SERVER

TUS ARCHIVOS MULTIMEDIA EN TODOS TUS DISPOSITIVOS

por Bruno Doiche y Rick Doiche

Hace unos dos meses, mi hermano menor se encontraba buscando una nueva placa montar un servidor multimedia, en una milésima de segundo le dije: “¡Hazte con un ODROID!”.

Luego le di un ODROID-XU4 con el que ha trabajado para migrar todo su contenido a su nueva máquina. Cuando me pregunto por un tutorial sobre la instalación de Plex Media Server, le indique que consultara la página web de Plex y le expliqué lo que tenía que hacer para instalarlo, me dijo que podía hacerlo por sí mismo.

Unos días más tarde, contactó conmigo por un chat y me dijo:

“Sabes una cosa, he escrito un script que facilita la instalación de Plex, ¿quieres incluirlo en la revista?”

Así que, sin más preámbulos, aquí está el script de mi hermano para que instales Plex Media Server en tu ODROID:

```
#!/bin/bash

#####
# Install Plex Media Server
#
#
# Odroid Magazine 2015
# http://magazine.odroid.com/
#
# This script will install
# Plex Media Server
# Beta 0.2
# 26 Aug 2015
#####

# Cheching system packages dependencies
DEPENDENCIES() {
PACK_LIBC="libc6-armel";
PACK_MULTILIB="gcc-multilib";
CHECK=$(dpkg-query -l $PACK_LIBC $PACK_MULTILIB > /dev/null 2>&1 ; echo
$?);
if [ "$CHECK" -eq "1" ]; then
    echo "Installing Packages $PACK_LIBC and $PACK_MULTILIB "
    apt-get install -y libc6-armel gcc-multilib ;
    locale-gen en_US.UTF-8 ;
```

```

dpkg-reconfigure locales ;
else
echo "INFO: Packages $PACK_LIBC and $PACK_MULTILIB are installed
already"
fi )

# Creating a build in environment
BUILD(){
URL="https://downloads.plex.tv/plex-media-server/0.9.12.11.1406-8403350/";
PLEX_SPK="PlexMediaServer-0.9.12.11.1406-8403350-arm.spk";

mkdir /tmp/plex ; cd /tmp/plex ;
wget -P /tmp/plex $URL$PLEX_SPK ;
mv $PLEX_SPK PlexMediaServer.tgz ;
tar -xvf PlexMediaServer.tgz ;
mkdir /tmp/plex/package ;
tar -xvf /tmp/plex/package.tgz -C /tmp/plex/package ;
mkdir -p /apps/plexmediaserver/Binaries ;
mv /tmp/plex/package/** /apps/plexmediaserver/Binaries ;
mkdir /apps/plexmediaserver/temp ;
mkdir /apps/plexmediaserver/MediaLibrary ; };
touch /var/log/plex/plexms.log ; chown plex /var/log/plex/plexms.log ;

ADD_PLEX(){

K=$(useradd plex -s /bin/bash -d /home/plex ; echo $?);
if [ "$K" -eq "0" ]; then
mkdir /home/plex
chown -R plex.plex /home/plex
echo "INFO: Plex user has been created sucessfully";
else
echo "INFO: Plex user already exists";
fi };

REMOVE_TEMP_BUILD(){
rm -rf /tmp/plex ; };

UNINSTALL(){
/etc/init.d/plex stop
userdel plex ;
rm -rf /home/plex /apps/plexmediaserver /etc/default/plexmediaserver-en-
vironment /etc/init/plexmediaserver.conf /etc/plex /etc/init.d/plex /
var/log/plexms.log ;
update-rc.d plex remove ;
};

```

```

PLEX_CONF() {
mkdir /etc/plex ;
ln -s /home/plex/Library/Application\ Support/Plex\ Media\ Server/Prefer-
ences.xml /etc/plex/Preferences.xml ;
};

CALL_INFO() {
echo -e "\033[01;31m# ODDROID MAGAZINE - Plex installation script
\033[00;37m"
echo -e "\033[01;31m# INFO:\033[00;37m";
echo -e "\033[01;31m# -----
-----\033[00;37m"
echo -e "\033[01;31m# Plex script will install Plex media Server on your
system.\033[00;37m";
echo -e "\033[01;31m# As requirement this script must be run as
root.\033[00;37m"
echo -e "\033[01;31m# Plex script will also add \"plex\" user to your
system in order to avoid security issues\033[00;37m";
echo -e "\033[01;31m# Directories as /home/plex and /apps/plexmediaserver
will be created.\033[00;37m";
echo -e "\033[01;31m# Plex script requires internet access once it has to
access and download Plex media server from http://Plex.tv\033[00;37m";
echo -e "\033[01;31m# Please note that some System package libs are also
required and script will try to install it\033[00;37m";
echo -e "\033[01;31m# Libs: libc6-armel and gcc-multilib \033[00;37m"
echo -e "";
echo -e "";
echo -e "TERM:"
echo -e "\033[01;31m# Running this script you acknowledge and accept that
ODDROID MAGAZINE will not be responsible for any damage caused in your
system. \033[00;37m"
echo "";

echo -e "\033[01;31m# -----
-----\033[00;37m" ;
echo " ";
echo " ";
echo -e "\033[01;32mOps.. please try $0 {install|uninstall|info}\033[00;
37m";
echo -e "\033[01;32mExample: $0 install\033[00;37m";
echo "";
echo -e "May the force be with you";

};

DEBIAN_SYSTEM_SCRIPT() {

sudo bash -c `cat <<EOT > /etc/init.d/plex
#!/bin/bash

```

```

##      ##   ##   ##   ##   ##   ##   ##
##      ##   ## #   ##   ##   ##   ##
##      ##   ## #   ##   ##   ##   ##
#####  ##   ##   ###  #####   ##   ##

#####
# Henrique Doiche                               #
# Plex Media Center                             #
# http://www.plexapp.com/                       #
# Last edition 07-02-2015                       #
# Plex script                                   #
#####
### BEGIN INIT INFO
# Provides:          scriptname
# Required-Start:    \\$remote_fs \\$syslog
# Required-Stop:     \\$remote_fs \\$syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start daemon at boot time
# Description:       Enable service provided by daemon.
### END INIT INFO

#####Comment#####
# NOTE:
# You can also add plex script into Debian /etc/init.d/
# and add it to run in startup as priority 50
# update-rc.d plex defaults 50
# That way you are \"Debian\" compliance
#
#
# Old school
# If you add this script named as plex in
# /usr/local/bin directory, Linux will
# be able to read it as System default \\$PATH
# so you'll be able to add it into rc.local:
# Add: \"plex start\" to your /etc/rc.local
# file. without \"\" =)
# Ex: Script Location: /usr/local/bin/plex
# rc.local exemple: cat /etc/rc.local
# plex start

#####

START(){
    sudo su - plex -c \"/apps/plexmediaserver/Binaries/start.sh > /
var/log/plex/plexms.log 2>&1 &\" ;
    echo -e \"Starting Plex [\033[01;32m Done \033[00;37m]\"; }

STOP(){

```

```

PIDS=\\$(ps aux | grep plex | grep -v grep | grep -v root | awk
{'print \\$2'});
PIDS_DLNA=\\$(ps aux | grep DLNA | grep -v grep | awk {'print
\\$2'});
if [ -z "\\${PIDS}" ] || [ -z "\\${PIDS_DLNA}" ]; then
    echo \"Plex isn't running. Nothing to do.\";
else
    echo \"Starting graceful shutdown...\";
    kill -s TERM $PIDS $PIDS_DLNA 2> /dev/null ;
    sleep 5;
    if [ -z "\\${PIDS}" ] && [ -z "\\${PIDS_DLNA}" ] ; then
        echo -e \"Graceful shutdown was [\\033[01;32m Suc-
cessful \\033[00;37m] \";
    else
        echo -e \"Plex process are still running. Killing
Process [\\033[01;32m Done \\033[00;37m]\";
        kill -9 \\${PIDS} \\${PIDS_DLNA} 2> /dev/null ;
    fi
fi }

RESTART(){
    STOP;START;
    echo -e \"Restarting Plex [\\033[01;32m Done \\033[00;37m]\"; }

STATUS(){
    STATUS=\\$(ps aux | grep plex | grep -v grep | grep -v root |
awk {'print \\$2'});
    if [ -z "\\${STATUS}" ]; then
        echo \"Plex isn't running\";
    else
        echo -e \"Plex is running on PIDs \\n\\033[01;31m\\${STATUS}
\\033[00;37m\";
    fi
}

case \\$1 in
'start') START ;;
'stop') STOP ;;
'restart') STOP; START ;;
'status') STATUS ;;
*)
echo \"Ops.. please try \\$0 {start|stop|restart|status}\";
exit 0
;;
esac

EOT"

```

```

chmod 755 /etc/init.d/plex

}

PLEX_MEDIA_CONF() {
bash -c `cat <<EOT > /etc/init/plexmediaserver.conf
# plexpms - service job file

description `Plex Media Server`
author `http://www.plexapp.com/`

# When to start the service
start on runlevel [2345]

# When to stop the service
stop on runlevel [016]

# Automatically restart process if crashed
respawn

# Sets nice and ionice level for job
nice -5

# What to execute
script
/etc/init.d/plex
end script

EOT`
};

PLEX_MEDIA_ENV() {
# Creating plexmediaserver_environment

bash -c `cat <<EOT > /etc/default/plexmediaserver_environment
# default script for Plex Media Server

# the number of plugins that can run at the same time
PLEX_MEDIA_SERVER_MAX_PLUGIN_PROCS=6

# ulimit -s \\$PLEX_MEDIA_SERVER_MAX_STACK_SIZE
PLEX_MEDIA_SERVER_MAX_STACK_SIZE=3000

# uncomment to set it to something else
PLEX_MEDIA_SERVER_APPLICATION_SUPPORT_DIR="/apps/plexmediaserver/MediaLibrary"

# let's set the tmp dir to something useful.
TMPDIR="/apps/plexmediaserver/temp`

```

```

# We need to catch our libraries
LD_LIBRARY_PATH="/apps/plexmediaserver/Binaries:\\\\$LD_LIBRARY_PATH\\"

EOT"
};

PLEX_STARTUP() {
# Creating start.sh
rm -rf /apps/plexmediaserver/Binaries/start.sh ;
bash -c "cat <<EOT > /apps/plexmediaserver/Binaries/start.sh
#!/bin/bash

#SCRIPTPATH=\\\\$(dirname \\\$(python -c 'import sys,os;print os.path.
realpath(sys.argv[1])' \\\$0))
SCRIPT=\\\\$(readlink -f \\\$0)
SCRIPTPATH=\\\\`dirname \\\${SCRIPT}`\\\\`
export LD_LIBRARY_PATH=\\\\`\\\\\${SCRIPTPATH}`\\\\`
export PLEX_MEDIA_SERVER_HOME=\\\\`\\\\\${SCRIPTPATH}`\\\\`
export PLEX_MEDIA_SERVER_MAX_PLUGIN_PROCS=6
export LC_ALL="en_US.UTF-8\\"
export LANG="en_US.UTF-8\\"
ulimit -s 3000
cd \\\${SCRIPTPATH}
./Plex\\ Media\\ Server
EOT"
chmod 755 /apps/plexmediaserver/Binaries/start.sh ;
};

ROOT=$(whoami);

case $1 in
`install`)

if [ "$ROOT" == "root" ]; then
clear ;
DEPENDENCIES ;
ADD_PLEX ;
BUILD ;
PLEX_CONF ;
PLEX_MEDIA_ENV ;
PLEX_MEDIA_CONF ;
PLEX_STARTUP ;
DEBIAN_SYSTEM_SCRIPT ;
REMOVE_TEMP_BUILD ;
# update-rc.d plex defaults;
clear ;
echo "-----";
echo "INFO:";

```



```

    echo "Please use service plex start | service plex stop | service
plex restart";
    echo "Plex installation completed";
    echo "You can reach server typing http://127.0.0.1:32400/web/index.
html into browser";
    echo "Install completed";
    else
    echo -e "\033[01;31mINFO:\033[00;37m";
    echo -e "\033[01;31mPlex installation script must be run as root
user\033[00;37m";
fi

;;
'uninstall')

UNINSTALL ;;
'info')

CALL_INFO ; ;;

*)
CALL_INFO ;
exit 0
;;
esac

```

Este script ha sido probado en los modelos U2, U3, X2, XU3 y XU4. También está disponible para que lo descargues desde <http://bit.ly/1LgYazS>.

En el caso de que no estés utilizando la distribución de Linux por defecto que proporciona Hardkernel, tendrás que crear el directorio `/var/log/plex` y darle permisos `777` usando `chmod`.

Acabé preguntando a Rick si estaba interesado en escribir más artículos sobre diversos temas puesto que es un experto en Linux, simplemente dijo, "h, ahora No hermanito, algún día... ¡quién sabe!"



Bruno espera que su hermano escriba con él otro artículo para la revista

USAR EL USB-UART CON OSX DE MAC

AYUDANDO A NUESTROS USUARIOS DE MAC A TENER ACCESO POR CONSOLA A SUS ODROIDS

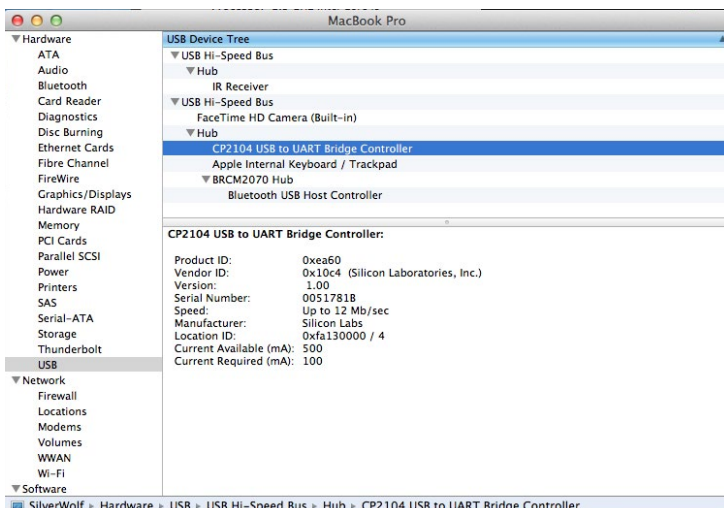
por @midel

Si utilizas un Macintosh como equipo principal, puedes usar el kit USB-UART de Hardkernel para leer la consola de un ODROID sin necesidad de instalar una máquina virtual Linux, ya que OSX esta basado en BSD. En este artículo se describe cómo instalar el software y los drivers necesarios para utilizar un Macintosh como consola de depuración para el desarrollo con ODROID.

Instalar los drivers

El primer paso es conseguir los drivers del UART, que puedes descargar desde <http://bit.ly/1Fk1rBu>. Descomprime e instala el paquete, después reinicie el sistema.

Asegúrate de que el driver se ha instalado correctamente conectando el UART en el puerto USB de tu Mac, y comprobando la información del sistema bajo **Apple Logo > About This Mac > More Info... > System Report... > Hardware > USB**, busca la entrada CP2104 USB to UART Bridge Connector



Revisando el sistema para verificar la instalación del driver

Configurar el software

Minicom permite mostrar los resultados de la consola del ODROID en la pantalla del Macintosh. Para usar Minicom, es necesario instalar primero homebrew desde <http://bit.ly/1R4sYYX>, así como instalar las herramientas de línea de co-



Puedes utilizar OSX para conectarte a la consola de tu ODROID

mandos (CLT) para Xcode desde <http://apple.co/1JsNXyi>. El paquete CLT es necesario para compilar el software con brew, ports o fink. Ya que eres un desarrollador de plataformas Mac, no es mala idea guardarlo si quieres utilizar las herramientas Linux GNU en tu máquina.

Abre la aplicación Terminal, que se encuentra en /Applications/Utilities/Terminal. Instala Minicom con este comando:

```
$ brew install minicom
```

Llevará unos minutos compilar el programa. El siguiente paso es conseguir el nombre real del dispositivo para el KIT UART.

Conectar el UART

Ahora que tenemos los driver y Minicom instalados, es el momento de averiguar dónde está. Escribe el siguiente comando en una ventana de terminal. El resultado será similar al que aparece en la siguiente imagen.

```
$ ioreg -c IOSerialBSDClient | grep USBtoUART
```

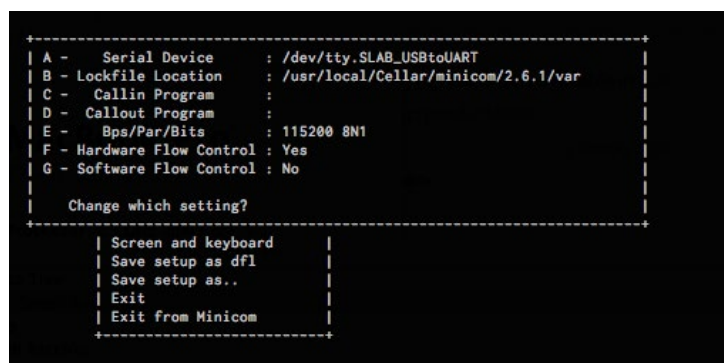


Resultado del ioreg

A continuación, abre Minicom en modo SETUP con este comando:

```
$ sudo minicom -s
```

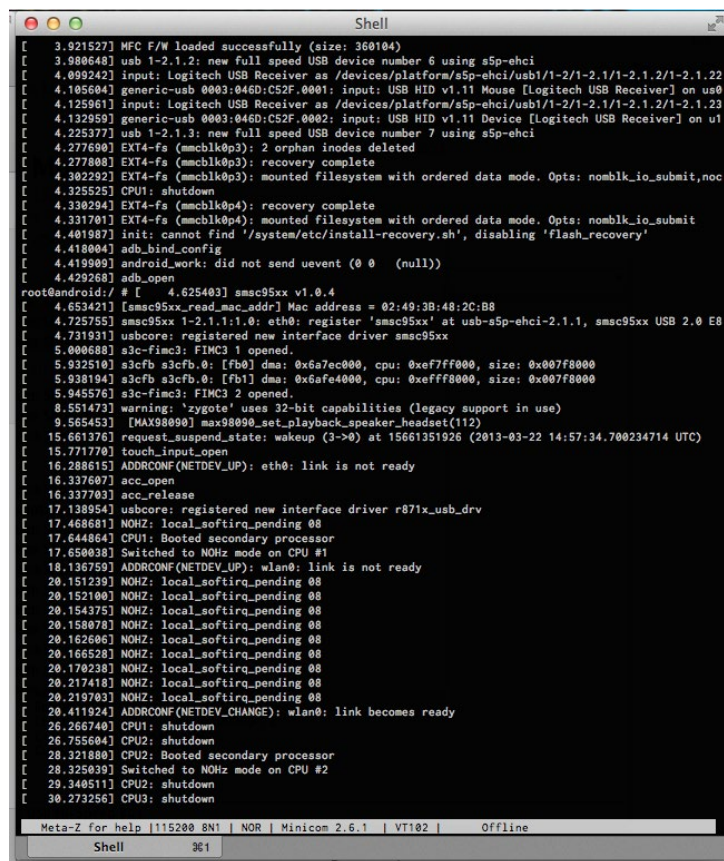
Introduce tu contraseña de usuario y pulse enter, Luego, dirígete usando las teclas de flechas a la configuración del puerto serie/opción A para cambiar el “Serial Device” por el resultado del comando anterior. El opción “hardware flow con-



Cambiando el dispositivo serie usando Minicom

rol” debe estar en OFF y el “software flow control” debe estar en ON.

Pulsa Intro hasta que regreses al menú principal, y selecciona “Save setup as dfl “ para guardar la configuración y así no tener que repetirla de nuevo. Selecciona Exit para entrar en la vista principal. Luego, salte de Minicom usando Esc + X.



Datos de ejemplo del dispositivo USB-UART

Ingresar al sistema

Escriba lo siguiente comando para iniciar Minicom, después pulsa ESC + L.

```
$ sudo minicom
```

Facilita un nombre de archivo y presiona intro. Ejecuta tu ODDROID y toma nota de la entrada de datos. Después, detén o cierra la conexión con ESC + L. Encontrarás el archivo log en el directorio desde el cual fue iniciado Minicom

Consejos

Si no deseas utilizar MiniCom, OSX viene con una pantalla GNU que forma parte de la instalación por defecto. Esto significa que una vez que hayas instalado los drivers de USB-UART, puedes localizar el puerto correcto para establecer la conexión con el siguiente comando:

```
$ ls -l /dev/tty.*
```

En mi sistema, el dispositivo UART aparece como “dev/tty.SLAB_USBtoUART”. Puedes conectarte al ODDROID utilizando la aplicación “screen”:

```
$ screen /dev/tty.SLAB_USBtoUART 115200
```

Para enviar comentarios, preguntas o sugerencias sobre el uso del kit USB-UART en OSX, visita el hilo original en <http://bit.ly/1Wm6BRs>.

Este artículo nos da una razón para utilizar realmente un Macintosh



DESARROLLO ANDROID

COMPILAR ANDROID PARA EL ODROID-C1 - PARTE 2

por Nanik Tolaram



En mi anterior artículo explique cómo compilar Android para el ODROID-C1. Es de esperar que ahora estés familiarizado con el desarrollo de imágenes Android partiendo de cero, y que hayas hecho algunos experimentos con tu placa. En este artículo voy a tratar el proceso de inicio de Android para el C1, ya que en este caso es ligeramente diferente al que normalmente encontramos en otras placas ODROID.

selfinstall-odroidc.bin

El sistema de compilación del ODROID-C1 produce un único archivo llamado selfinstall-odroidc.bin que se tiene que copiar a la tarjeta microSD. Lo fascinante de este archivo es que contiene todas las imágenes importantes de Android listas para usarse con la placa, como se muestra en la Figura 1.

Este archivo actúa como un contenedor, agrupando a varios

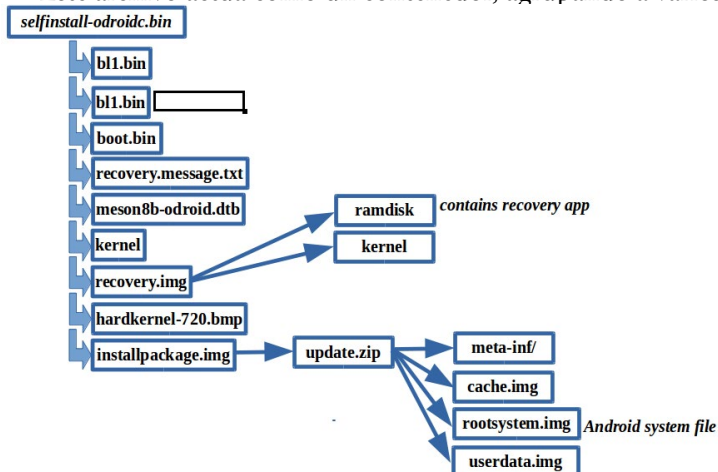


Figura 1: Contenido de selfinstall-odroidc.bin

archivos formando un paquete completo para el proceso de instalación. La extracción e instalación de este sistema de ficheros tan diversos durante el proceso de arranque se realiza con la aplicación "recovery" que se carga dentro del archivo recovery.img. La figura 2 muestra el fragmento del script de compilación que agrupa los archivos formando un único archivo .bin.

```
$(PRODUCT_OUT)/selfinstall-$(TARGET_DEVICE).bin: \
$(INSTALLED_BOOTIMAGE_TARGET) \
$(INSTALLED_RECOVERYIMAGE_TARGET) \
$(PRODUCT_OUT)/bl1.bin.hardkernel \
$(PRODUCT_OUT)/u-boot.bin \
$(PRODUCT_OUT)/installpackage.img
@echo "Creating installable single image file..."
dd if=$(PRODUCT_OUT)/bl1.bin.hardkernel of=$@ bs=1 count=442
dd if=$(PRODUCT_OUT)/bl1.bin.hardkernel of=$@ bs=512 skip=1 seek=1
dd if=$(PRODUCT_OUT)/u-boot.bin of=$@ bs=512 seek=64
dd if=$(RECOVERY_MESSAGE_FILE) of=$@ bs=512 seek=1016
dd if=$(PRODUCT_OUT)/meson8b_odroidc.dtb of=$@ bs=512 seek=1088
dd if=$(PRODUCT_OUT)/kernel of=$@ bs=512 seek=1216
dd if=$(INSTALLED_RECOVERYIMAGE_TARGET) of=$@ bs=512 seek=17600
dd if=$(PRODUCT_OUT)/hardkernel-720.bmp of=$@ bs=512 seek=33984
dd if=$(PRODUCT_OUT)/installpackage.img of=$@ bs=512 seek=49152
sync
@echo "Done."
```

```
.PHONY: selfinstall
selfinstall: $(PRODUCT_OUT)/selfinstall-$(TARGET_DEVICE).bin
```

Figura 2: Script de compilación de selfinstall-odroidc1.bin

Es importante tener en cuenta que cambiar la secuencia o eliminar cualquier cosa del script de compilación puede hacer que tu imagen no arranque. Los cambios en el script requieren de otros cambios en el gestor de arranque.

Flujo del arranque

El proceso de arranque de esta placa en particular es un poco más complicado de lo normal. Hay 2 fases, como muestra la Figura 3. La 1ª fase del arranque comprueba si la tarjeta microSD se ha formateado y la formatea si fuera necesario. Una vez completada la primera fase, continua el arranque pasando a la segunda fase en la cual se lanza el proceso init de Android.

Cuando la placa se enciende, el chip Amlogic ejecuta la primera parte del proceso de arranque ejecutando el gestor de arranque principal bl1.bin proporcionado por el fabricante del chip. Una vez que termine el gestor de arranque principal, el U-Boot iniciará la ejecución, que es la parte del gestor de arranque que decide cuál será el siguiente paso, en función de si la tarjeta microSD ha sido formateada. Si U-Boot detecta que la tarjeta microSD no ha sido formateada, formateará la tarjeta con el tipo de partición correspondiente y copiará archivos

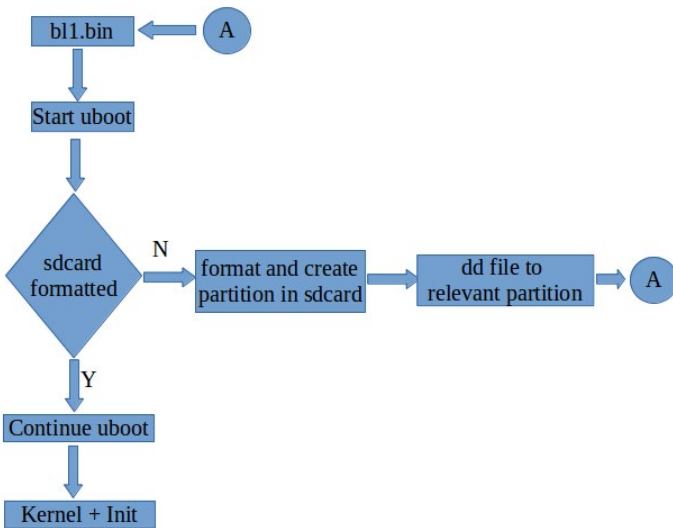


Figura 3: Flujo de arranque del ODROID-C1

en la misma. Una vez completado este proceso, se reiniciará la placa. En la segunda fase de arranque, si el U-Boot detecta que la partición es la correcta, entrega el control al kernel.

U-Boot

La Tabla 1 muestra un registro log parcial del U-Boot durante la primera fase del proceso de arranque, donde se puede observar que creó las desaparecidas particiones de Android requeridas. Una vez completado el formato, ejecuta la aplicación recovery y copia las imágenes a la partición recién creada.

Dado que en la primera fase de arranque no hay particiones en la tarjeta microSD, ¿cómo sabe U-Boot qué leer y desde

Tabla 1: Registro log de la primera fase del arranque

```

-----
* Welcome to Hardkernel's ODROID-C... (Built at 19:33:00 Dec  8 2014) *
-----
CPU : AMLogic s805
MEM : 1024MB (DDR3@792MHz)
DRAM: 1 GiB
relocation Offset is: 2ff1c000
....
U-boot-00000-gc878e20-dirty(odroidc@ec878e205) (Jun 28 2015 - 20:20:12)
Partition #1: 1 GiB
mmc save env ok
vpu clk_level = 3
....
-----
Vendor: Man 035344 Snr ala3a066 Rev: 8.0 Prod: SU08G
Type: Removable Hard Disk
Capacity: 7580.0 MB = 7.4 GB (15523840 x 512)
-----
Partition  Start Sector  Num Sectors  Type
bad MBR sector signature 0x0000
-----
Creating default partition...
partition #  size(MB)  block start #  block count  partition_id
1           2924      9486336        5988352       0x0C
2           1024       49152         2097152       0x83
3           3072      2146304        6291456       0x83
4            512       8437760        1048576       0x83
MMC read: dev # 0, block # 17600, count 16384 ... 16384 blocks read: OK
## ANDROID Format IMAGE
## Booting kernel from Legacy Image at 12000000 ...
Image Name:   Linux-3.10.33
Image Type:   ARM Linux Kernel Image (lzo compressed)
Data Size:   5008332 Bytes = 4.8 MiB
Load Address: 00208000
Entry Point: 00208000

```

donde? Para averiguarlo, vamos a ver cómo es internamente la tarjeta microSD antes de que U-Boot cree cualquier partición. La figura 4 muestra que la tarjeta contiene diferentes tipos de información, la aplicación recovery, el gestor de arranque, el logo, el kernel y mucho más. Obviamente, si analizamos este tipo de información, U-Boot debe tener una estructura similar almacenada en algún lugar del código fuente que puede utilizar para leer la información, la figura 5 muestra esta estructura.

Utilizando la orden `sys_partitions []`, U-Boot es capaz de

```
442 bytes | bl1.bin | U-boot.bin | recovery.message.txt | mesonfb.odroid.dtb | kernel | recovery.img | hardkernel-720.bmp | installpackage.img
```

Figura 4: Contenido de la SD en el primera fase del arranque

determinar la ubicación de la información que necesita. Por ejemplo, sabe donde se encuentra la aplicación recovery, de modo que puede leerla desde la tarjeta microSD e incluirla en la memoria para ser ejecutada. Una vez que todos los archivos necesarios están en su lugar, el proceso de arranque puede continuar y finalmente mostrar el escritorio para que el usuario empiece a interactuar.

```

struct fbt_partition fbt_partitions[] = {
    {
        .name = "system",           /* 2nd primary partition */
        .type = "ext4",
        .size_kb = BOARD_SYSTEMIMAGE_PARTITION_SIZE * 1024
    }, {
        .name = "userdata",        /* 2rd primary partition */
        .type = "ext4",
        .size_kb = BOARD_USERDATAIMAGE_PARTITION_SIZE * 1024
    }, {
        .name = "cache",           /* 3rd primary partition */
        .type = "ext4",
        .size_kb = BOARD_CACHEIMAGE_PARTITION_SIZE * 1024
    }, {
        .name = "fat",              /* 1st primary partition */
        .type = "vfat",
        /* FIXME: MUST fit in remaind blocks after followed by this */
        .size_kb = 1024 * 1024,
    }, {
        .name = 0,
        .type = 0,
        .size_kb = 0
    },
};

```

Figura 5: Ubicación en la SD de diferentes contenidos

Partición

La figura 6 muestra las particiones creadas por el U-Boot durante la primera fase del arranque. Como puede verse en la imagen, las particiones contienen `/system`, `/data`, `/cache` y `vfat`. U-Boot también almacena toda la información de las particiones dentro del código fuente, como puede verse en la Figura 7.

La estructura `fbt_partitions []` contiene las diferentes particiones que U-Boot creará durante la primera fase del arranque. Puedes comprobar las referencias y ver que la información del tamaño de la partición se indica en el campo `size_kb` detallado en la estructura que coincide con el registro log de la Tabla 1.

Un truco de lujo

Para entender mejor el contenido de `selfinstall-odroidc.bin`, te voy a decir cómo extraer el archivo `.bmp` que se utiliza como logo de U-Boot. Recuerda que el siguiente paso sólo lo puedes hacer después de haber grabado `selfinstall-odroidc.bin` en tu



FREE ORION

CONQUISTA LA GALAXIA

por Tobias Schaaf

FreeOrion es un juego de estrategia por turnos ambientado en el espacio donde conquistas galaxias e imperios, gratuito y de código abierto. Está inspirado en la saga Master of Orion. Puedes ver un video del juego en <http://bit.ly/1LD0x3R>.

Prerequisitos

Primero, actualizar el kernel con el script ODROID-Utility. Después, vincula los drivers Mali (en el XU3 y XU4, usa libmali.so en lugar de libMali.so):

```
$ sudo ln -sf /usr/lib/arm-linux-gnueabi/hf/mali-egl/libMali.so \
/usr/lib/arm-linux-gnueabi/hf/\
libGLESv1_CM.so
$ sudo ln -sf /usr/lib/arm-linux-gnueabi/hf/mali-egl/libMali.so \
/usr/lib/arm-linux-gnueabi/hf/\
libGLESv2.so
$ sudo ln -sf /usr/lib/arm-linux-gnueabi/hf/mali-egl/libMali.so \
/usr/lib/arm-linux-gnueabi/hf/\
libEGL.so
$ cd ~ && mkdir freeorion
$ wget http://oph.mdrjr.net/\
meveric/other/freeorion/\
libgl-odroid_20150922-1_armhf.deb
$ wget http://oph.mdrjr.net/\
meveric/other/freeorion/\
libglues-odroid_\
20140903-1_armhf.deb
$ wget http://oph.mdrjr.net/\
meveric/other/freeorion/\
libglew-odroid_1.11.0-2_armhf.deb
```

Ubuntu 14.04

```
$ wget http://oph.mdrjr.net/\
meveric/other/freeorion/\
```

Disk /dev/sdg: 7948 MB, 7948206080 bytes
255 heads, 63 sectors/track, 966 cylinders, total 15523840 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: 0x0000de21

Device	Boot	Start	End	Blocks	Id	System
/dev/sdg1		9486336	15474687	2994176	c	W95 FAT32 (LBA)
/dev/sdg2		49152	2146303	1048576	83	Linux
/dev/sdg3		2146304	8437759	3145728	83	Linux
/dev/sdg4		8437760	9486335	524288	83	Linux

VFAT

Name	Size	Type	Date Modified
Alarms	0 Items	Folder	Tue 01 Jan 1
Android	2 Items	Folder	Tue 01 Jan 1
DCIM	0 Items	Folder	Tue 01 Jan 1
Download	0 Items	Folder	Tue 01 Jan 1
LOST.DIR	0 Items	Folder	Tue 01 Jan 1
Movies	0 Items	Folder	Tue 01 Jan 1
Music	0 Items	Folder	Tue 01 Jan 1

1.1 GB Volume

Name	Size	Type	Date Modified
app	44 Items	Folder	Wed 01 Jul 2015 13:06:44 EST
bin	195 Items	Folder	Mon 21 Sep 2015 20:20:26 EST
boot	0 Items	Folder	Wed 01 Jul 2015 11:35:14 EST
data	0 Items	Folder	Wed 01 Jul 2015 12:04:46 EST
dev	0 Items	Folder	Wed 01 Jul 2015 12:04:46 EST
etc	0 Items	Folder	Mon 21 Sep 2015 20:19:48 EST
fonts	0 Items	Folder	Wed 01 Jul 2015 12:29:22 EST
framework	0 Items	Folder	Wed 01 Jul 2015 13:07:17 EST
lib	0 Items	Folder	Wed 01 Jul 2015 12:04:46 EST
lost+found	7 Items	Folder	unknown

1.1 GB Volume

Name	Size	Type	Date Modified
app-asset	2 Items	Folder	Fri 02 Jan 1970 10:01:23 EST
app-lib	2 Items	Folder	Thu 01 Jan 1970 10:00:04 EST
app-private	0 Items	Folder	Thu 01 Jan 1970 10:00:04 EST
backup	2 Items	Folder	Fri 02 Jan 1970 10:00:16 EST
dalvik-cache	67 Items	Folder	Fri 02 Jan 1970 10:01:26 EST
data	69 Items	Folder	Fri 02 Jan 1970 10:01:23 EST
font-pacific	7 Items	Folder	Thu 01 Jan 1970 10:00:04 EST
font	7 Items	Folder	Thu 01 Jan 1970 10:00:04 EST
local	7 Items	Folder	Thu 01 Jan 1970 10:00:04 EST
lost+found	7 Items	Folder	unknown
media	7 Items	Folder	Thu 01 Jan 1970 10:00:04 EST
mediadm	7 Items	Folder	Thu 01 Jan 1970 10:00:04 EST

537 MB Volume

Name	Size	Type
backup	0 Items	Folder
lost-found	7 Items	Folder
recovery	2 Items	Folder
checksum_list	180 bytes	plain text document
update.zip	263.3 MB	Zip archive

Figura 6: Las diferentes particiones Android

```
struct fbt_partition fbt_partitions[] = {
    {
        .name = "system", /* 2nd primary partition */
        .type = "ext4",
        .size_kb = BOARD_SYSTEMIMAGE_PARTITION_SIZE * 1024
    }, {
        .name = "userdata", /* 2rd primary partition */
        .type = "ext4",
        .size_kb = BOARD_USERDATAIMAGE_PARTITION_SIZE * 1024
    }, {
        .name = "cache", /* 3rd primary partition */
        .type = "ext4",
        .size_kb = BOARD_CACHEIMAGE_PARTITION_SIZE * 1024
    }, {
        .name = "fat", /* 1st primary partition */
        .type = "vfat",
        /* FIXME: MUST fit in remaind blocks after followed by this */
        .size_kb = 1024 * 1024,
    }, {
        .name = 0,
        .type = 0,
        .size_kb = 0
    },
};
```

Figura 7: Información de las particiones del U-Boot

sdcard. Para extraer el .bmp de la tarjeta SD, escribe la siguiente instrucción en una ventana de terminal:

```
$ sudo dd if=/dev/sdg \
of=logo.bmp bs=512 \
skip=33984 count=5400
```

Verás un archivo llamado logo.bmp en el directorio actual. Una vez modificado el archivo .bmp, puedes colocarlo de nuevo en la tarjeta SD con la siguiente instrucción:

```
$ sudo dd if=./logo.bmp \
of=/dev/sdg bs=512 \
seek=33984
```



```
freeorion-data_0.4.5-\  
1~ppal~trustyl_all.deb  
$ wget http://oph.mdrjr.net/  
meveric/other/freeorion/  
freeorion_0.4.5-1~\  
ppal~trustyl_armhf.deb
```

Ubuntu 15.04

```
$ wget http://oph.mdrjr.net/  
meveric/other/freeorion/  
freeorion-data_0.4.5-\  
1~ppal~vivid1_all.deb  
$ wget http://oph.mdrjr.net/  
meveric/other/freeorion/  
freeorion_0.4.5-1~\  
ppal~vivid1_armhf.deb
```

Debian Jessie

```
$ wget http://oph.mdrjr.net/  
meveric/pool/main/f/freeorion-  
odroid/freeorion-odroid_0.4.5-  
1+deb8_armhf.deb
```

Instalación

```
$ sudo apt-get install gdebi  
$ sudo gdebi libgl-*.deb  
$ sudo gdebi libglues-*.deb  
$ sudo gdebi libglew-*.deb  
$ sudo gdebi freeorion-data*.deb  
$ sudo gdebi freeorion_*.deb
```

Para jugar, haz clic en el icono FreeOrion en la sección de Juegos del menú de aplicaciones. Ten en cuenta que si estás utilizando la imagen GameStation Turbo, el único paso necesario es escribir “apt-get install freeorion-odroid”. Para comentarios, sugerencias y preguntas, puedes visitar el hilo original en <http://bit.ly/1OwEb6i>, o echa un vistazo a la guía para principiantes de FreeOrion en <http://bit.ly/1KULQsv>.

FreeOrion tiene magníficos gráficos



HAXIMA NAZGHUL

NUEVA AVENTURA PARA FANS DE ULTIMA V

por @petevine



Haxima Nazghul es un CRPG (juego de rol) que sigue el modelo de la popular serie Ultima. Es un juego de aventura y fantasía que ofrece una línea argumental independiente del juego original Ultima, aunque el sistema de juego es similar. Se desarrolla siguiendo concretamente el modelo de Ultima V, por lo que si has jugado a este juego, Haxima Nazghul debería serte muy familiar.

Instalación

Descarga el código fuente y los datos del juego de <http://bit.ly/1MOCvEE>, luego descomprimirlo escribiendo lo siguiente en una ventana de terminal:

```
$ cd ~/Downloads  
$ tar xvzf nazghul-0.7.1.tar.gz
```

Después, descarga el parche desde <http://bit.ly/1NZkTGz>, coloca el archivo en el directorio de nivel superior del código fuente, y aplique el parche:

```
$ cd ~/Downloads  
$ mv va_list_patch.txt naz-  
ghul-0.7.1/  
$ cd nazghul-0.7.1/  
$ patch -p0 < va_list_patch.txt
```

Por último, compila el ejecutable desde la fuente y lanza el juego:

```
$ ./configure  
$ make  
$ sudo make install  
$ haxima.sh
```



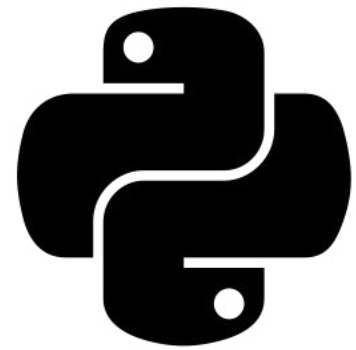
Para conocer mejor Haxima Nazghul, visita la página web <http://bit.ly/1FyW7d8>. Para comentarios, preguntas y sugerencias, consulta el hilo original en <http://bit.ly/1NZkIR0>.



USAR PYTHON CON EL ODRROID-SHOW

CON SHOWTIME TODO ES MAS FACIL

por @Matoking

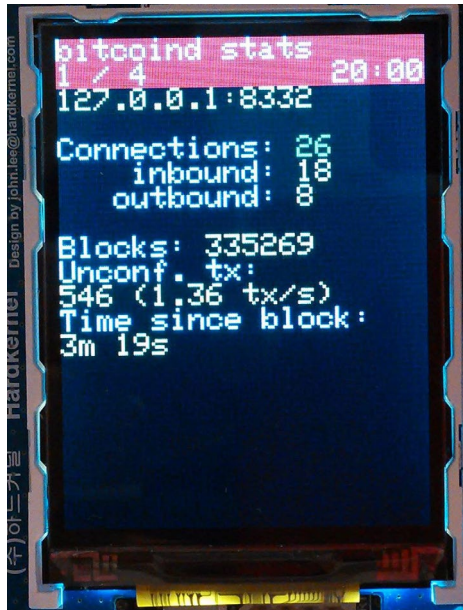


He creado un script Python llamado ShowTime para mostrar distintos tipos de información en un ODRROID-SHOW usando pestañas. Se puede descargar desde mi repositorio Github en <https://github.com/Matoking/SHOWtime>. Básicamente, Showtime utiliza pestañas para mostrar información, como es el uso de RAM, la cotización del Bitcoin o el uso del disco. Las pestañas va cambiando con un intervalo preestablecido, el valor por defecto es 15 segundos.

Otro dato interesante es que en lugar de tener que enviar los códigos de escape ANSI manualmente, he creado una clase para que todo se haga correctamente, usando un método de encadenamiento:

```
ctx.fg_color(Screen.RED).
write("Hello").linebreak().
fg_color(Screen.BLUE).
```

Figuras 1 - 2: Pantallazos del Showtime



```
write("world!")
```

También he incluido una clase llamada Screen-Context en context.py que permite casi todo lo que se puede hacer en el ODRROID-SHOW usando Python y sin tener que preocuparte por el estrangulamiento en la entrada o la entrada manual de códigos escape. La Impresión de texto, cambiar colores de fondo o de primer plano, realizar saltos de línea, así como la mayoría de funcionalidades se pueden hacer usando correctamente el método de encadenamiento.

Empecemos

Suponiendo que hayas creado un archivo .py y hayas realizado los pasos descritos en INSTALL, puedes empezar con la siguiente plantilla:

```
from context import Screen, ScreenContext
import atexit
```

```
ctx = ScreenContext("/dev/tty-
USB0")

# Make sure the cleanup routine
is called to clear the screen
# when we close the script
atexit.register(ctx.cleanup)

# Wait 6 seconds for the screen
to boot up before we start up-
loading anything
ctx.sleep(6).reset_lcd().set_ro-
tation(0)
```

Esta plantilla crea un nuevo contexto en pantalla que podemos utilizar para interactuar con el ODRROID-SHOW. Debemos esperar unos 6 segundos para asegurarnos que ODRROID-SHOW muestra la pantalla de arranque, tras lo cual podemos estar seguros de que todos los códigos se reciben y se gestionan correctamente. Después, podemos empezar con un simple programa "Hello World". Inserta lo siguiente al final del script:

```
# Main loop
while True:
    ctx.fg_color(Screen.RED).
write("Hello").linebreak()
    ctx.fg_color(Screen.BLUE).
write("world!").home()
```

Esto crea un simple bucle que muestra el texto "¡Hello world!" en el ODRROID-SHOW, la palabra "Hello" en rojo en la primera línea y la palabra

“Word” en azul en la segunda línea.

La última secuencia `home()` hace que el cursor se coloque de nuevo en el inicio, de lo contrario las palabras “Hello” y “world” aparecerían hasta salirse de la pantalla. Ahora puede ejecutar el script utilizando el intérprete de Python. Suponiendo que colocaste el nombre `example.py` al fichero, puedes ejecutar lo siguiente en una ventana de terminal:

```
$ python example.py
```

Ten en cuenta que no es necesario activar `sleep()` para acelerar la ejecución del script para mantener el ODRROID-SHOW de forma sincronizada puesto que `ScreenContext` ya se encarga de ello. Sin embargo, si fuera necesario por cualquier razón, puede activar `ctx.sleep(seconds)` para detener la ejecución del script durante los segundos que desees. En caso de que sólo quieras utilizar `ScreenContext` pero no el script `Showtime` en sí, puede simplemente copiar `context.py`, `port_open` y `utils.py` y colocarlos en el mismo directorio que el script.

Todos los métodos de `ScreenContext` han sido comentados, por lo que no deberías tener problemas para comprobarlo tú mismo si lo necesitas. Hay, sin embargo, algunos métodos que pueden necesitar alguna demostración adicional para poder utilizarlos.

Prevenir el efecto fantasma

Vamos a probar el siguiente script.

```
eggs = 555
spam = 1234
while True:
    ctx.write("Eggs
%d" % eggs).line-
break()
    ctx.write("Spam
%d" % spam).home()
    eggs = 99
    spam = 321
```

Observando el código, te podrás imaginar lo que aparecerá en pantalla:

```
Eggs 99
Spam 321
```

Sin embargo y puesto que tenemos que escribir explícitamente sobre el texto ya mostrado para eliminarlo, aparece lo siguiente en su lugar:

```
Eggs 995
Spam 3214
```

Afortunadamente, `ScreenContext` tiene un buen método que imprime el texto enviado y rellena el resto de líneas con espacios en blanco, impidiendo estos problemas de imagen fantasma. Puedes corregir el ejemplo con:

```
eggs = 555
spam = 1234
while True:
    ctx.write_line("Eggs %d" %
eggs)
    ctx.write_line("Spam %d" %
spam).home()
    eggs = 99
    spam = 321
```

Ten en cuenta que esto también elimina la necesidad de utilizar `linebreak()` para cambiar la línea.

Para más información o para enviar preguntas, comentarios o sugerencias, visita los hilos originales en <http://bit.ly/1G7xAa1> y <http://bit.ly/1VfzMmW>.



PRINCE OF PERSIA RESCATA A LA PRINCESA EN ESTE CLASICO DE DOS CON DESPLAZAMIENTOS LATERALES

por Tobias Schaaf

Prince of Persia fue un juego DOS muy querido de la década de los 90. Tienes que evitar trampas mortales, resolver simples salto y puzles ambientales, y combatir con los guardianes. Para instalar Prince of Persia, descarga el archivo .deb de <http://bit.ly/1LIsPLU>, luego, escribe en una ventana de terminal:

```
$ sudo apt-get install \
gdebi xboxdrv
$ cd ~/Downloads
$ sudo gdebi ./sdlpop-odroid*
```

Al juego se puede jugar con un teclado o joystick. Para utilizar un mando Xbox 360, escribe lo siguiente en una nueva ventana de terminal antes de iniciar Prince of Persia:

```
$ sudo xboxdrv --dpad-only
```

Inicia el juego escribiendo lo siguiente en una ventana de terminal:

```
$ cd /usr/local/share/SDLPop
$ prince .
```

Presione `Ctrl-J` para habilitar el joystick y luego ¡Salva a la princesa!

Prince of Persia fue un juego revolucionario que usaba curiosas animaciones



JUEGOS LINUX: DREAMCAST

LA ULTIMA GRAN
CONSOLA DE SEGA
LLEGA A LA
PLATAFORMA ODROID

por Tobias Schaaf



La SEGA Dreamcast es una de mis consolas favoritas de todos los tiempos, y funciona bastante bien en los dispositivos ODROID. Por ello y en honor a SEGA Dreamcast he decidido escribir un artículo ODROID sobre esta impresionante consola, así podrás conocer con más detalle de lo que es capaz si la ejecutas en tu dispositivo ODROID. La SEGA Dreamcast es junto con la PlayStation Portable el sistema gráfico más impresionante que los dispositivos ODROID pueden emular por ahora, con espectaculares gráficos y una jugabilidad vertiginosa.

Dreamcast – un poco de historia

La SEGA Dreamcast fue la última de las grandes consolas creadas por SEGA. En los años 80 y 90, SEGA y Nintendo eran los dos principales rivales en el mercado de las consolas y siempre estaban compitiendo entre sí. Nintendo lanzó su Nintendo Entertainment System (NES) y SEGA respondió con la Master System. Nintendo tuvo su GameBoy y GameBoy Color y SEGA su GameGear. Súper Nintendo Entertainment System frente a la Génesis/MegaDrive. La batalla fue dura y aunque Nintendo llegó a alcanzar algo más de mercado, los sistemas de SEGA a menudo eran superiores en relación a las especificaciones de hardware. Sin embargo, SEGA al final

cometió muchos errores y los desarrolladores empezaron a estar molestos por el rápido y continuo lanzamiento de nuevas consolas (SEGA CD, Sega 32X, etc.). Por ello, cuando SEGA anunció la Dreamcast tuvo problemas para encontrar desarrolladores que dieran soporte al dispositivo, y dio por finalizada la producción de la mayor parte de los juegos para la consola.

La consola era la mejor que podías conseguir por aquel entonces con espectaculares gráficos e incluso soporte para Microsoft Windows CE. Tenía un módem integrado y de hecho era la primera consola que permitía juegos multijugador online, los primeros juegos multijugador a gran escala. Phantasy Star Online fue el primer juego que ofrecía una comunidad online donde podías conocer gente, formar grupos e ir a misiones juntos. Mejor aún, fue el primer juego que salió para diferentes plataformas donde podías jugar con otros jugadores.

Poco después de la aparición de la Dreamcast, la PlayStation 2 fue anunciado con un hardware muy superior y lo peor de todo (para SEGA y Nintendo) soporte para DVD. Sin embargo SEGA y la Dreamcast todavía tenían más de un año para consolidar un mercado con sus juegos y servicios antes de que la PlayStation 2 apareciera en el mercado.

Como he mencionado anteriormente, Dreamcast fue la última consola

de SEGA, luego anunciaría que dejaba de producir cualquier hardware y se limitaría a desarrollar juegos para otras consolas.

Al final, SEGA llegó incluso a producir títulos para su gran rival Nintendo, y de hecho hoy en día es posible jugar al Sonic en la Wii o Wii U.

Más detalles

Hablemos ahora sobre lo que puedes esperar de esta consola:

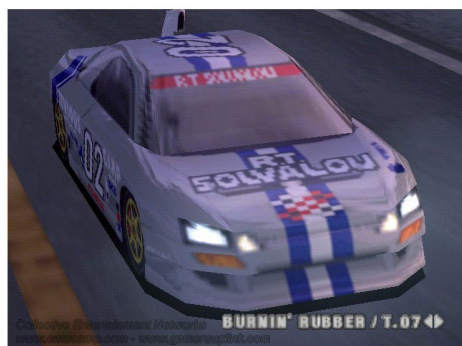
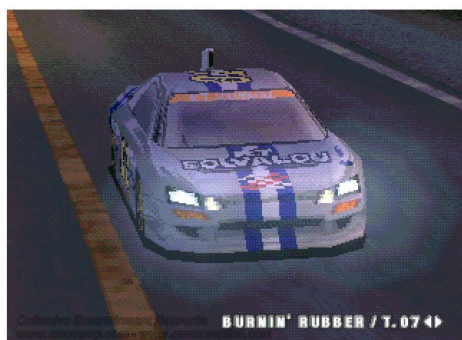
La Dreamcast salió entre la Sony PlayStation 1 y la PlayStation 2, lo cual describe bastante lo que es capaz de hacer. Era mucho mejor que una PlayStation 1, aunque en realidad no tan buena como la PlayStation 2.

De hecho, hubo un proyecto para ejecutar juegos de PlayStation 1 en la Dreamcast (llamado Bleemcast) que hacía que los juegos se viesen realmente mejor que en la propia PlayStation 1 gracias a una mayor resolución y potencia gráfica. No se trataba de REMAKE de los juegos, sino que eran ejecutados a través un emulador, por lo que la Dreamcast en realidad tenía que emular los juegos y aún así era capaz de mejorar los gráficos.

Juegos

Además de esto, posiblemente quieras saber qué tipo de juegos se puede ejecutar en la Dreamcast.

La Dreamcast ofrecía una amplia va-



Ridge Racer Type 4, ejecutándose de forma nativa en la PlayStation (arriba), y bajo el emulador Bleemcast! (abajo)

riedad de juegos de todos los géneros. En total, más de 600 juegos fueron lanzados para Dreamcast para diferentes regiones (EE.UU. / Europa / Japón), y algunos fans de Dreamcast siguen desarrollando sus propios juegos hoy en día.

Juegos arcade

La Dreamcast es probablemente la consola más famosa por la amplia gama de juegos arcade que se hicieron para ella. Antes de la Dreamcast, muchas consolas intentaron llevar los juegos arcade a “las casas”, sin embargo Dreamcast realmente hizo que esto fuera posible, y de hecho resultó ser mejor que las propias máquinas arcade.

Dreamcast llevo muchos juegos que ya existían en las máquinas arcade a los televisores de muchos hogares, con impresionantes gráficos, gran sonido y con hasta cuatro jugadores en una consola.

Juegos de lucha

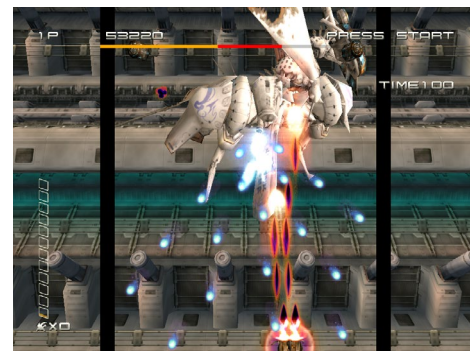
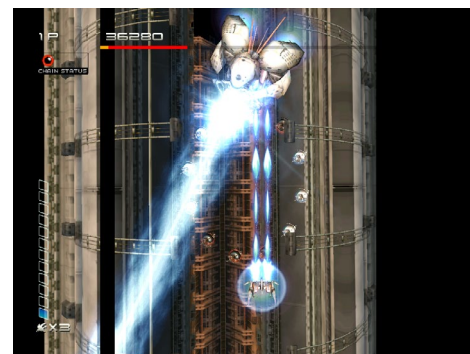
Uno de los juegos más famosos de Dreamcast es Soul Calibur, que resultó ser muy superior a la versión arcade, y en la Dreamcast inició una nueva serie de juegos de lucha. Aunque Soul Calibur ya está disponible para muchas consolas

(incluyendo PS3, Xbox 360, e incluso Android y iOS), la mayoría de estos juegos, probablemente no habría ni siquiera existido si no fuera por el éxito de la versión para Dreamcast. Personalmente prefiero Soul Calibur sobre la mayoría del resto de juegos de lucha, en especial la serie Tekken que en mi opinión es bastante rudo en comparación con los juegos Soul Calibur. Tener espadas, hachas y otras armas, y el hecho de disponer de botones para bloquear el ataque de un enemigo hacían que el juego fuese mucho más atractivo para mí por aquel entonces.

Pero hay muchos más juegos de lucha en Dreamcast, Power Stone 2, por ejemplo, que es un juego de peleas muy bueno para hasta cuatro jugadores, o los muchos y famosos juegos de Capcom como Marvel vs Capcom 2: New Age of Heroes, King of Fighters, Last Blade 2, los juegos de Street Fighter, Dead or Alive 2 (que también tenían estupendos gráficos), Mortal Kombat, y muchos muchos más.

Pero la Dreamcast tiene más juegos Arcade que ofrecer, además de los juegos de lucha y peleas.

Soul Calibur (arriba) y Power Stone 2 (abajo) en el ODDROID-U3 ejecutándose a través del emulador reicast



Ikaruga

Hay muchos y muy buenos side scrollers y shoot arcades como Ikaruga o Giga Wing.

Ikaruga es un shooter arcade japonés para un máximo de dos jugadores en el que puede cambiar el color de tu nave para causar daño extra y evitar ser golpeado por el enemigo. Este shooter tiene impresionantes gráficos en 3D y muchos efectos de iluminación.

Giga Wing es un shooter de “la vieja

GigaWing



escuela” que también es para un máximo de dos jugadores. Hubo incluso un sucesor: Giga Wing 2, ambos se ejecutan muy bien en el ODROID-U3.

Hay muchos más juegos como estos, y por supuesto no son los únicos que se exportaron a Dreamcast. Existe una enorme lista de juegos Arcade que se hicieron para Dreamcast incluyendo juegos como House of Dead 2, Virtua Cop 2, Virtua Fighter, etc.

Juegos originales de Dreamcast

Los juegos arcade no eran los únicos juegos de SEGA (y de otras empresas) llevados a la Dreamcast. Hay muchos títulos para la consola que no se sacaron de las máquinas arcade, o que simplemente se ven mucho mejor en la Dreamcast.

Juegos de Terror

La Dreamcast tuvo juegos en todos los géneros, incluyendo los juegos de supervivencia y terror como la serie Resident Evil. En Dreamcast, este fue el famoso juego “Resident Evil - Code: Veronica”, que primero salió para Dreamcast y más tarde para otros sistemas como la PS2 y

Imágenes del Resident Evil: Code Veronica. Rostro de Claire Redfield (arriba), Primer encuentro con Zombie (abajo)



GameCube. Fue el primero de su tipo, en incorporar en tiempo real fondos 3D en lugar de simplemente imágenes 2D.

Pero este no fue el único juego de su tipo en Dreamcast. También estaba el Alone in the Dark: The New Nightmare, Blue Stinger, Carrier, D2, Nightmare Creatures II y la lista sigue y sigue. Si eres un fan de los juegos de terror, Dreamcast tiene un montón de juegos que ofrecerte.

Juegos de rol (RPG)

Aunque no son los juegos más comunes en Dreamcast, ésta cuenta con algunos muy buenos. Lamentablemente ninguno de los “grandes juegos” tiene su versión para Dreamcast, por lo que no encontrarás juegos como Final Fantasy, la saga Tales o la saga Dragon Quest en Dreamcast. Aunque todavía hay algunos juegos de rol muy buenos para Dreamcast como el famoso Grandia II, Evolution 1 y 2, y el Time Stalkers. Puede que no sean los juegos de rol más conocidos, pero aún así son muy buenos y en su mayoría sólo están para Dreamcast (a excepción de Grandia II).

Grandia 2 es un RPG muy bueno que de hecho venía con dos discos, y para hacer que los efectos especiales fueran más impactantes, éstos eran ejecuta-

Grandia



Evolution

dos sobre la capa del juego creando un ambiente muy intenso e impresionantes efectos especiales en los “mega ataques”. También ofrecía un estilo de lucha muy interesante que te permitía interrumpir un ataque enemigo antes de que te golpeará.

Evolución 2 es un JRPG muy gracioso, con adorables y divertidos personajes. Aunque los juegos RPG son raros en Dreamcast, los pocos que existen son bastante buenos y pueden mantenerte ocupado durante muchas horas. Aún me pregunto cómo deberían verse el Final Fantasy o juegos similares con la impresionante potencia de la CPU y la GPU.

Juegos de carreras

No quiero profundizar demasiado en este tipo de juegos sobre Dreamcast. ¡Sólo lo suficiente para decir que sin duda alguna existen! Hay juegos por ahí como Metropolis Street Racer, Monaco Grand Prix, F335 Challenge, Sega GT, Hydro Thunder, Sega Rally, Test Drive 6, Star Wars Racer, e incluso juegos de coches teledirigidos como Re-Volt (uno de mis favoritos, no sólo en Dreamcast). Los juegos de carreras son divertidos en Dreamcast, aunque el género no es mi favorito y por ello no voy a precisar cuál

podría ser el “mejor” de todos.

Juegos de deportes

Aunque tampoco son mis favoritos, Dreamcast cuenta con un buen número de juegos de este tipo. Empezando por títulos de juegos de NBA, NHL, NFL y Ready 2 Rumble a juegos como Virtua Tennis 2 y juegos de golf. Hay un montón de juegos de deportes, pero al único al que personalmente me gustaba jugar era Virtua Tennis 2. Y aunque por lo general no me gustan demasiado los juegos de deportes, éste era realmente divertido y me pasaba muchas horas jugando en mi Dreamcast.



Virtua Tennis 2: Uno de los pocos juegos de deportes con el que realmente disfruto jugando y está para Dreamcast

Plataformas

Todas las consolas presentan juegos de este estilo, los juegos de plataformas (también llamado Jump 'n Run) son uno de los primeros juegos en salir para toda consola. La Dreamcast tiene algunos juegos de este estilo realmente buenos tales como el famoso Rayman 2 y Jet Grind Radio, un juego en el que juegas con un artista graffiti de patinaje con el

Rayman 2 y Jet Grind Radio eran dos juegos de plataformas muy divertidos



que pueden hacer todo tipo de acrobacias, un juego muy divertido y colorido.

Juegos Especiales

La Dreamcast tiene algunos juegos muy buenos que no quisiera incluir en los diferentes géneros. Como por ejemplo, una versión de Grand Theft Auto 2 (GTA2) o el Half-Life que fue exportado a Dreamcast (junto con sus expansiones Half-Life: Blue Shift y Half-Life: Opposing Force). Salieron varios juegos de Disney para Dreamcast como Donald Duck Goin' Quackers y Toy Story 2. Otros como Quake 3 Arena, Unreal Tournament, Railroad Tycoon 2, Worms Armageddon y Worms World Party también podían encontrarse para Dreamcast. La lista de juegos es muy extensa teniendo en cuenta que la consola sólo estuvo en el mercado desde finales de 1998 hasta 2002, momento en que SEGA anunció que dejaba de crear nuevos dispositivos.

Juegos que definió la Dreamcast

Existen unos cuantos juegos en Dreamcast que realmente definieron la historia de los juegos y destacaron en Dreamcast. Uno de esos juegos que ya he mencionado es Soul Calibur, esta se-

GTA 2 en la Dreamcast: El movimiento es un poco complicado.



rie nació con Dreamcast y realmente fue un escaparate para la consola. Sus gráficos son impresionantes, muy suaves y muestran justamente lo que esta consola es capaz de hacer. Fue exportado de las máquinas Arcade a las consolas de sobremesa, superando incluso la versión original de las máquinas Arcade. Aún es considerado como uno de los mejores juegos de la historia de los videojuegos.

Otro juego que se creó expresamente para Dreamcast es Sonic Adventure y su sucesor Sonic Adventure 2. Hubo algunos intentos para llevar a Sonic al mundo 3D, pero Dreamcast es la única que realmente fue capaz de producir un juego que tuviese todo lo que se puede esperar de un sonic: sonic, anillos, una asombrosa música y lo más importante de todo ¡VELOCIDAD! El juego da realmente la sensación de velocidad, especialmente cuando das vueltas en bucle o por el aire. Encontraras en el juego 3D todo a lo que estás acostumbrado a ver en el 2D: los puestos de control, las cajas extras con anillos, escudos, correr más rápido, etc. El segundo juego incluso te permitía jugar como los malos.

Sonic Adventure 2: Desplazarte por las esquinas capturando los anillos o correr lo suficientemente rápido como para desplazarte por las paredes son tan sólo dos de las cosas que puede hacer.



Otro de los mejores juegos para Dreamcast es Crazy Taxi y Crazy Taxi 2. Ambos salieron primero como juegos arcade, aunque su traslado a Dreamcast fue verdaderamente bueno y además contaban con contenido adicional. La banda sonora es simplemente increíble,

Sonic Adventure 2: Desplazarse por una barandilla y correr en bucle te hacen realmente sentir la velocidad.



Crazy Taxi: Recoge a un cliente y llevarlo al destino de cualquier modo posible



y el juego es muy rápido y divertido. ¿Dónde vas a conducir tu coche por el parque, o saltando por encima de otros coches como un loco?

La acción rápida y las carreras no fueron los únicos grandes juegos de Dreamcast. Los pocos juegos de rol que habían para Dreamcast eran también bastante impresionantes. Skies of Arcadia es un RPG magnífico en el que juegas como Vyse, un "Pirata del cielo" en un universo inspirado en Julio Verne. Viajas entre islas suspendidas en el aire y luchas contra los monstruos y los soldados del Imperio Valuan. Este juego lo tiene todo: buenos personajes con una gran historia de fondo, progresión de los personajes, armas y naves mejorables, combates entre naves, impresionantes hechizos, una historia muy intensa y ¡Oh ¡He mencionado los PIRATAS DEL AIRE?!

Pero aquí no es donde nos detenemos cuando se trata de hablar de la Dreamcast. He mencionado anteriormente que Dreamcast incorporaba un módem que te permitía jugar online con amigos. Esto también abrió el mundo de las

Skies of Arcadia: Muy buenos gráficos, peleas basadas en un sistema de energía, habilidades complejas y los combates entre naves hacen de éste un juego de rol impresionante para la Dreamcast.



Skies of Arcadia: Desde el principio del juego los efectos son muy buenos en todos los ataques especiales.

consolas a los primeros juegos MMO. Fantasy Star Online fue el primer juego multijugador multiplataforma donde podrías permanecer en una ante sala con tu personaje, luego adentrarte en sitios inexplorados y resolver misiones junto con tus amigos. Incluso disponías de un banco para almacenar tu dinero y objetos. Por aquel entonces, este juego era muy bueno y contaba con gráficos bastante aceptables, algunas personas continúan jugando hoy en día en servidores privados. Es un juego de rol muy bueno en el que no tenías que centrarte en cómo configurar tu personaje (como fuerza y agilidad), sino que más bien debías centrarte en equilibrar tus habilidades usándolas en combate. Subir de nivel tus armas y tus habilidades para utilizarlas. La tarea que elijas define cómo progresará tu personaje. Este juego tuvo un gran éxito en Dreamcast, y abrió el mercado de los MMO en consolas y otras plataformas. Ha tenido varios sucesores como Fantasy Star Online v2 (también en Dreamcast), o Fantasy Star Azul Shift (en Windows), y años más tarde la serie Fantasy Star Universe para PC, PS2 y Xbox 360.



Phantasy Star Online: Pantalla de creación del personaje (arriba) la primera visita a la estación espacial Pioneer 1 (abajo). PSO ofrece muchas cosas interesantes, algunas de los cuales hicieron historia.



Shenmue 2: Los detalles faciales son impresionantes, representación gráfica del pelo (arriba). Una alegre plaza (abajo), no es raro ver lugares como este, con aves y muchas personas. Dreamcast podía manejar todo esto.



Phantasy Star Online: Tienda de objetos en la estación espacial (arriba) lucha con monstruos en una misión (abajo). PSO tenía muy buenos gráficos para ser un MMO de 1999.

El último título (o mejor dicho, la serie) que merece la pena mencionar

es la serie Shenmue. Aunque no es un juego 100% de “mundo abierto”, es lo más cerca que se puede llegar con algo como esto a finales de los 90. Shenmue fue un título espectacular. Básicamente fue creado para todos los fanáticos de juegos RPG y de aventura. Este juego es tan intenso que resulta difícil de entender todas las cosas que puedes llegar a hacer en él.

Eres Ryo Hazuki, un estudiante de artes marciales en busca de venganza por el asesinato de tu padre. Este juego es muy intenso. La historia es inmensa y de hecho fue diseñado para ser una trilogía. El juego ofrece ciclos de día y noche, con climas diferentes. Tiene que gestionar tu vida: ir a trabajar para ganar dinero, atender tus contactos sociales, puedes jugar con los mini juegos o simplemente ir de compras. Comer, beber y hablar con la gente para preguntarles por direcciones. No hay casi nada que no puedas hacer en este juego. Incluso tienes que realizar tareas simples como alimentar a un pequeño gatito. A medida que avanzas con el juego, tienes que luchar

contra otras personas, aprender nuevos movimientos de combate de extraños y amigos, y tienen que llevar a cabo eventos rápidos cuando luchas o sigues a determinadas personas. Este juego es uno de los juegos más caros de la historia de la producción de juegos (unos estimados 47 - 70 millones de dólares), y se puede observar que el dinero fue utilizado muy bien durante el desarrollo del juego. Los lugares se sienten muy vivos: siempre tienes muchas personas a tu alrededor que te ayudan con tu trabajo diario, se puede hablar con ellos, preguntar por direcciones, e interactuar con ellos de muchas maneras. Esta serie fue elogiada por la crítica y se puede encontrar en muchas de las listas de los “grandes videojuegos de todos los tiempos”. La serie se detuvo con Shenmue 2, dejándola sin final. Pero se ha anunciado recientemente que finalmente habrá un final con el sucesor Shenmue 3 financiado con Kickstarter por parte del fabricante de Shenmue 1 y 2. Se supone que será lanzado para PS4 y PC en diciembre de 2017.

Shenmue I: Lan Di (el asesino del padre de Ryo) agarrando a Ryo en el aire en la apertura del juego (parte superior). Mantener tu dinero junto no es siempre fácil (abajo).





ODROID Magazine esta ahora en Reddit!



**ODROID Talk
Subreddit**

<http://www.reddit.com/r/odroid>



Dreamcast en ODROID

Mucha gente habla sobre la Dreamcast, pero te estarás preguntando: ¿Qué tal funciona en los dispositivos ODROID?

Todas las imágenes de este artículo (excepto las de PS1 y Bleemcast) están directamente tomadas de un ODROID-U3 ejecutando el emulador reicast. Hay muchos juegos que funcionan muy bien, algunos presenta ciertos problemas y otros no se ejecutan en absoluto. Pero el ratio es bastante alto. Si tuviera que hacer una estimación, diría que entre el 60% y el 75% de todos los juegos de SEGA Dreamcast funcionan con reicast y por lo tanto en los dispositivos ODROID.

Los problemas más comunes son errores gráficos. Parece que la niebla y el LOD no siempre funcionan bien. A lo lejos, los objetos muestran patrones extraños en lugar de que la niebla los haga desaparecer poco a poco. Algunos juegos tienen ciertos problemas de sonido. Y casi todos los juegos tienen problemas con la reproducción de los vídeos, algo que en realidad es bastante extraño ya que los videos son de muy baja calidad.

Holzhaus de la comunidad ODROID desempeña un papel muy importante en el desarrollo de reicast, de hecho fue capaz de integrar soporte ODROID en la versión upstream de reicast.

Junto con otros pequeños errores, la versión actual de reicast funciona muy bien en nuestros ODROIDs (sólo tenía que hacer unos cambios menores para conseguir que funcionase de la forma que yo quería).

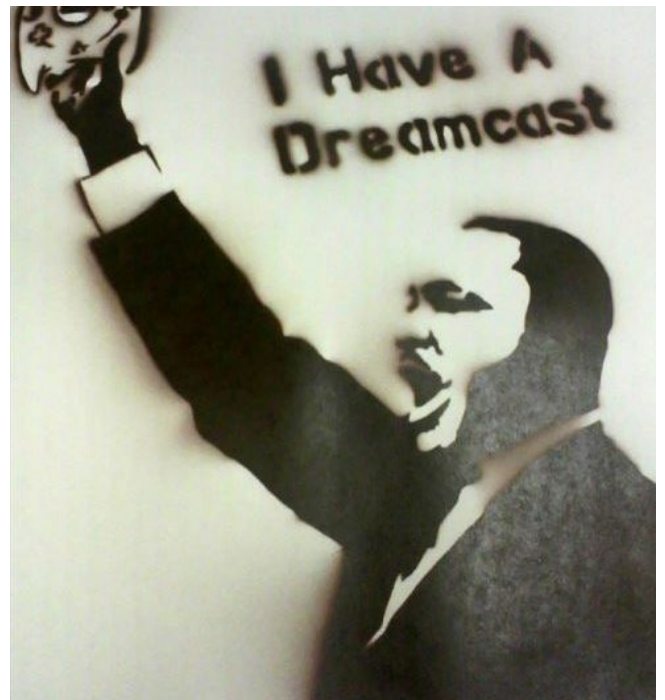
Estoy deseando que lleguen nuevas mejoras del emulador, actualizaré los paquetes de mi repositorio tan a



menudo como me sea posible para tener disponibles todas “las cosas buenas” que llegan para nuestros ODROIDs.

Junto al emulador de PSP, éste se ha convertido en uno de mis emuladores favoritos para los ODROIDs, me permite volver a jugar a mis juegos favoritos de Dreamcast sin tener que coger la vieja consola del sótano y buscar la forma de conectarla al mi actual TV. También me permite jugar con mis amigos, lo que hace que sea aún mejor que la PSP, porque ahora puedo luchar en Soul Calibur o Power Stone contra mis amigos y hacer competiciones.

Los Juegos de Dreamcast son un buen suplemento al ya impresionante número de emuladores y juegos que se ejecutan en dispositivos ODROID, sólo me queda sugerir a todos los que le gustan los juegos que echen un vistazo a los sorprendentes juegos de Dreamcast en la plataforma ODROID.



CONOCER A UN ODROIDIAN

WILLIAM HENNING (@MIKRONAUTS)

PROLIFICO BLOGERO TECNOLÓGICO
Y EXPERTO EN ROBOTICA

editado por Rob Roy

Por favor, hablanos un poco sobre ti.

Mi nombre es Bill y soy un adicto a la robótica y electrónica. Vivo en Langley, un barrio de Vancouver de la hermosa Columbia Británica, Canadá, con mi encantadora esposa Agnes, también conocida como “Wifey”. Soy el dueño, CEO, CTO y director jefe de Mikronauts (<http://www.mikronauts.com>), una consultora especializada en soluciones de software y hardware a medida mayoritariamente para clientes de control industrial. No es sorprendente, ya que tengo más de 30 años de experiencia en el control industrial como programador /analista, analista de sistemas, desarrollador de sistemas y director de proyectos técnicos. También diseño y vendo robótica y productos educativos. Tenemos una completa muchedumbre de “chicos malolientes” (sobrinos, llamados así por mi mujer) y una sobrina, a los que les encanta pasar tiempo con nosotros.

¿Cómo fueron tus inicios con los ordenadores?

Empecé con un mini-ordenador/calculadora HP que incluía un lector de tarjetas perforadas, un lector de cintas y una impresora lineal, ejecutaba un HP basic en grado 10. Más tarde, la escuela compró algunos Apple originales, ordenadores para el laboratorio. Desarrolle una placa Z80, y utilicé un Cosmac-Elf, un Kim-1 y otros ordenadores de placa reducida de la época. ¡Tuvimos un gran profesor de física/electrónica!

Tras conseguir algunos trabajos a tiempo parcial, ahorré algunos centavos y compré un Atari 400, luego un clon de Apple con una tarjeta Z80. Fui a la Universidad Simon Fraser de Ciencias de la Computación, donde también trabajé para el Departamento de Educación, y compré uno de los primeros ordenadores Amiga 1000 en Vancouver. Me quedé asombrado con las demos gráficas, el escritorio gráfico multitarea con 512KB y la unidad de disquete. Poco después, escribí un driver caché para “Wedge”, que era un controlador RLL PC/XT que permitía conectarme a Amiga con otro usuario de Amiga. También exporté un compilador Valgol al MC68000, me divertía un monto con él.

Después, escribí el software para manejar los reproductores de discos láser y dispositivos de video para “Amiga Theatre” en la Expo 86 de Vancouver. Fue curioso ver una pared de televisores ejecutando videos sincronizados controlados por Amiga.

En 1984, empecé a trabajar para Pan-Abode, un fabricante



¡William nos inspira a hacer mucho más con nuestros ODROIDS!

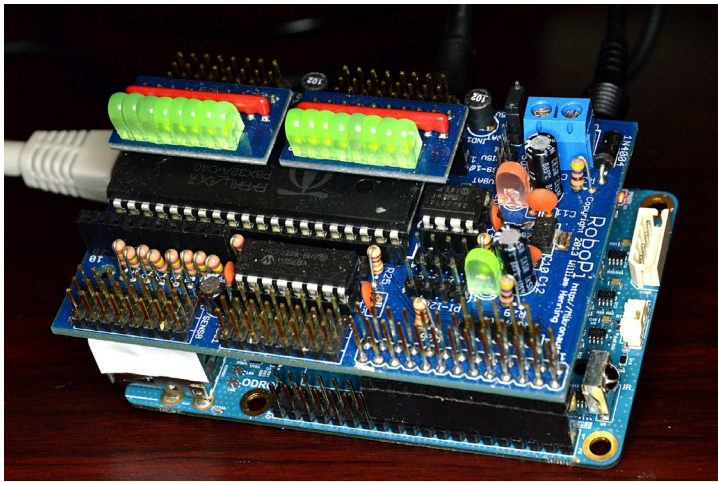
de casa de madera en Richmond, BC, donde empecé mi carrera con el control industrial. Escribí el software para controlar el secado de los troncos en grandes hornos, el software “Smart House” controlado por voz y escribí extensiones Lisp para AutoCAD, incluido los modelos de energía solar para la subida y bajada de temperatura de las casas de madera en función del tamaño de las ventanas, la orientación y los planos de diseño.

Más tarde, trabajé para Universal Dynamics, diseñando soluciones de software y hardware para grandes clientes industriales en América del Norte, como servicios públicos de energía, molinos, terminales de grano, hornos de fundición y muchos otros. A finales de 1990, me convertí en consultor y además cree dos sitios web, que llegaron a ser muy populares:

CPUReview.com, donde analizaba procesadores, placas madre, tarjetas de video de PCs y experimentaba con over-clocking. Lo usaba para escribir críticas sobre los nuevos procesadores de Intel y AMD que recibía bajo NDA con el fin de tener las revisiones listas para el lanzamiento de los productos.

AboutLinux.com, donde analizaba nuevas distribuciones Linux, software Linux y escribía guías y manuales.

Tras el colapso de la publicidad por Internet, dejé de actualizar ambos sitios y lamentablemente perdí los dos nombres de dominio en 2001. Nunca recibí avisos de renovación, y puesto que administraba mis propios servidores web y DNS, no me di cuenta de que los dominios me lo habían arrebatado. Después impartí clases en un laboratorio de redes avanzadas en el Instituto de Tecnología de Columbia Británica, donde más tarde investigue sobre la seguridad en las redes de control industrial,



RoboPi está siendo probado en un ODROID-C1 para analizar el C1

y llegue a poner en marcha un dispositivo de seguridad en red.

En 2006, leí algo sobre un nuevo microcontrolador llamado Parallax Propeller que tenía una arquitectura muy interesante, y estaba muy por delante de su tiempo. Con ocho núcleos RISC de 32 bits, memorias locales y una memoria compartida de 32 KB, era mucho más potente que los mini ordenadores. Empecé con Mikronauts.com como blog sobre Propeller y mis experimentos de electrónica. Pensé en ejecutar una pequeña versión de Unix con lo esencial en Propeller, pero la arquitectura estaba limitada a programas con un máximo de 506 instrucciones. Para superar esta limitación, se me ocurrió usar un “Gran Modelo de Memoria” para Propeller (LMM era un sistema para grandes modelo de memoria de 8086 compiladores) que utiliza un bucle de auto-modificación y ejecución que contiene tan sólo cuatro instrucciones para poder implementar una máquina virtual que permita la ejecución de programas de memoria compartida. Los 32KB permitieron 8192 instrucciones, ¡un incremento por 16 en tamaño sobre lo que se podía ejecutar inicialmente! Empecé a escribir un sistema operativo para el código LMM y un ensamblador LMM, pero por desgracia no tuve tiempo para exportarlo a un compilador C para hacer realidad mi idea de ejecutar un pequeño Unix en Propeller.

Sin embargo, diseñé varios ordenadores de placa reducida en torno a Propeller, incluyendo Morpheus, que era una doble máquina Propeller. Un Propeller se utiliza principalmente para E/S, y el otro para los gráficos de alta resolución. Tenía 512 KB de SRAM externa, ampliable a 16 MB y espacio swap. Llegue a utilizar otro proyecto llamado PropCade para emular VT100 y juegos retro.

Más tarde, utilice Parallax y su puerto GCC para Propeller, que utiliza LMM. Cuando el Raspberry Pi salió, abandone el diseño de equipos basados en Parallax Propeller, ya que no había manera de llegar ni siquiera a la relación precio/rendimiento de la Raspberry Pi. Sabía que Propeller sería una excelente expansión E/S en tiempo real para el Raspberry Pi, así que diseñé la placa controlador robot avanzada RoboPi basada en Propeller

para la Raspberry Pi.

Cuando empecé realmente a implicarme con los ordenadores de placa reducida basados en ARM de bajo costo fueron apareciendo cada vez más en el mercado, decidí volver a Mikronauts donde empezaría a analizar ordenadores de placa reducida y a publicar artículos sobre robótica y proyectos electrónicos en lo que intervienen estos ordenadores de placa reducida.

¿Qué te atrajo de la plataforma ODROID?

He encontrado un hilo sobre el ODROID-W en los foros Raspberry Pi. El ODROID-W parecía ser un gran módulo integrado para la robótica y el control industrial. Debido a su compatibilidad con Raspberry Pi, parecía encajar con RoboPi. Desafortunadamente, poco después de recibir mi pedido de ODROID-W (tres W, dos LCDs y otros módulos de expansión) el hilo fue cancelado, mucho antes de que yo pudiera escribir un análisis del mismo.

Afortunadamente, Hardkernel anunció el ODROID-C1 poco después, y de inmediato compre seis, junto con un montón de accesorios. Esto fue mucho antes de que saliese el Modelo 2 de Raspberry, ¿cómo iba a ir mal un SBC ARM de cuatro núcleos por 35\$? ¡Esta vez, termine el análisis! Descubrí que el ODROID-C1 superaba en gran medida a la Raspberry Pi original.

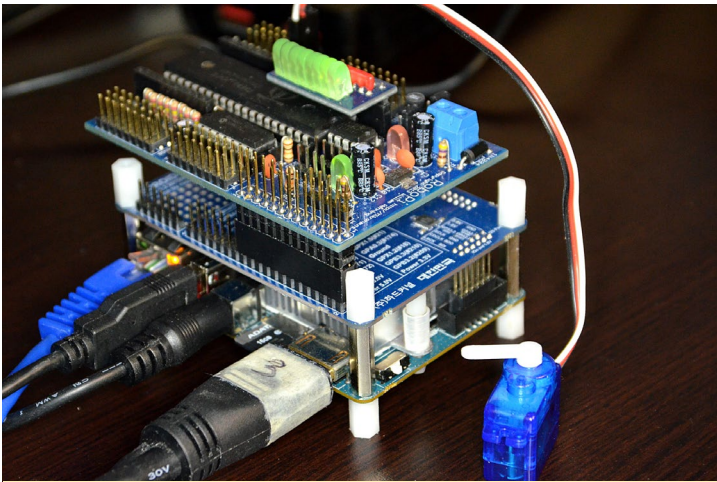
Meses más tarde, después de que la Fundación Raspberry Pi liberara la Raspberry Pi 2 Modelo B, el ODROID-C1 todavía la superaba en rendimiento. Cuando Hardkernel lanzó el XU3, estuve a punto de conseguir uno. Desafortunadamente, el precio era demasiado alto para ser considerada una placa ARM de bajo coste, en comparación con el ODROID-C1, Raspberry Pi 2, Plátano Pi y muchas otras placas ARM. Me encantaban sus características, pero el coste no hacía frente a mis necesidades.

Recientemente, Ameridroid me preguntó si estaría interesado en analizar la nueva ODROID-XU4, y sí que estaba interesado. Basándome en lo que había estado leyendo sobre los chips big. LITTLE ARM con ocho núcleos y con un precio inferior a la mitad del XU3, pensé que el XU4 podría tener la relación precio/rendimiento necesaria para justificar un precio algo mayor. Estaba en lo cierto, el rendimiento es más que excepcional.

¿Cómo utilizas tus ODROIDS?

En este momento, estoy usando mis C1s como pequeños sustitutos de mi PC escritorio y como reproductores multimedia. Mantengo la intención de hacer un robot basado en el C1+RoboPi. Sin embargo, me preocupa el consumo de energía cuando esté apagado el C1. Hace poco recibí una sugerencia para controlar esta cuestión por parte de uno de los administradores de los foros ODROID, pero no la he probado todavía.

He cambiado mi desarrollo de la librería C para RoboPi al XU4, ya que permite compilar mucho más rápido que en una



Probando RoboPi sobre el ODROID-XU4 para un análisis del XU4

Raspberry Pi, las librerías y los ejecutables compilados se ejecutan en mis otras placas basadas en ARM v7. Tengo en mente darles otros usos a mis C1s, que podrias ver en los próximos meses en www.mikronauts.com y quizás aquí en ODROID magazine.

¿Cuál es tu ODROID favorito?

Actualmente la XU4 es mi favorito. Es la placa ARM más rápida que tengo para compilar y probar código ARM v7, no tengo necesidad de preocuparme por la compilación cruzada desde un PC. También es un gran sustituto de mi escritorio y ejecuta kodi muy bien. El C1 está en un cercano segundo lugar, y tal vez sea una mejor opción para aplicaciones que no necesiten tanta velocidad y requieran de un consumo de energía menor.

¿Estás involucrado con otros proyectos informáticos a parte del ODROID?

Sí. Trabajo con muchos micro-controladores, SBC y ordenadores. Puedes esperar muchos más proyectos de electrónica, robótica y IoT en Mikronauts, incluyendo proyectos para pla-

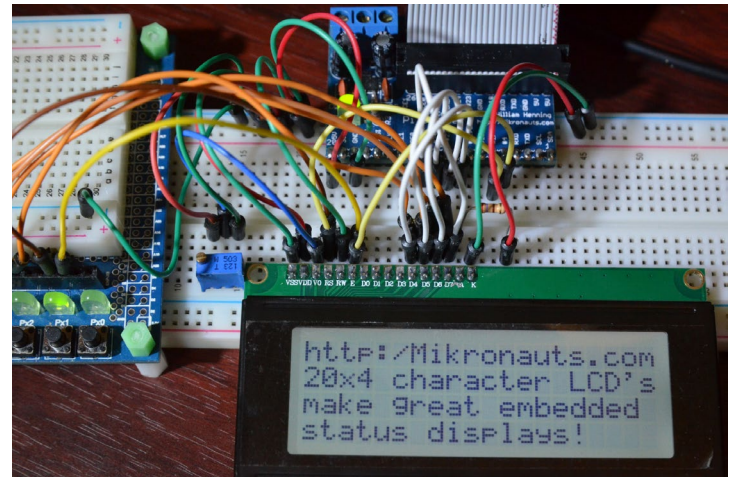
cas ODROID. Estoy desarrollando más productos dirigidos a los mercados de educación y la robótica y los pondré a prueba para ver su compatibilidad con el ODROID-C1 y ODROID-XU4.

Tengo varios proyectos en la sección de Raspberry Pi de mi sitio web que deberían funcionar en el ODROID-C1 y ODROIDXU4 con la placa Shifter simplemente cambiando los números GPIO. Si existe interés en ello, estaría encantado de publicar adaptaciones para C1 y XU4 de los siguientes:

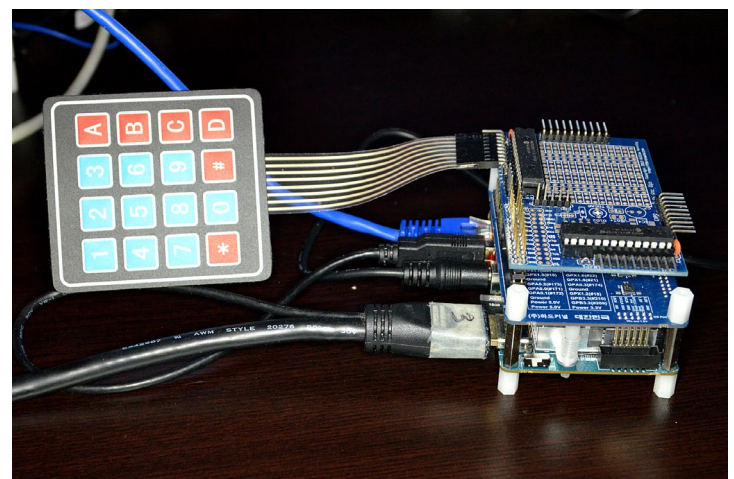
Conexión LCD 20x04 y 16x02 (con librería Python)

Teclado matricial 4x4 conectado con E/S I2C (con librería Python)

Placa de recogida de datos I2 bit 24 canales (con librería Python) ampliable a 64 canales



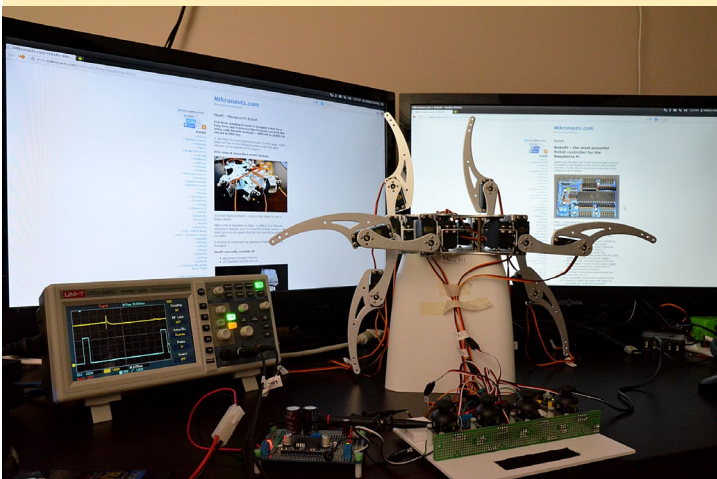
Conexión de LCDs 20x4 y 16x2, completa con librería Python

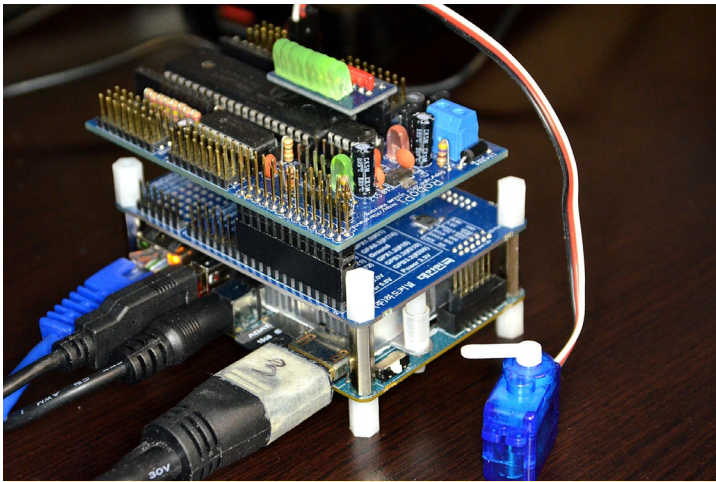


Proyecto de conexión de teclado matricial 4x4 con extensión E/S I2C

Algunos productos de Mikronauts no son totalmente compatibles con el C1 y XU4 debido a la diferencia en algunos pines del conector GPIO de 40 pines, como las entradas analógicas 1.8V. Algunos también dependen de software que

La estación de trabajo de desarrollo de software y CAD de William, muestra que está trabajando en HexPi





Proyecto de recogida de datos de 24 canales usando 3 GPIO, multiplexador y 1 chip SPI

no está disponible para C1 y XU4, como la librería pigpio y el driver del kernel servoblaster. No me malinterpreten, me gusta tener entradas analógicas en las placas ODROID, y tengo la intención de utilizarlas en algunos proyectos en cualquier momento.

¿Qué aficiones e intereses tienes aparte de los ordenadores?

Wifey - La menciono porque no soy idiota

Viajar - mi viaje favorito es volar a Hawaii, quedándome en la playa de Waikiki durante una semana, y navegar por las islas

Fotografía - incluso llegué a ser profesional por un tiempo a principios de este milenio

Lectura - sobre todo la ciencia ficción, leo un motón durante mis viajes

Cine y TV - Soy un gran fan de la ciencia ficción, acción/aventura y comedias

Familia - ponerse al día, jugando con y enseñar a las ratitas de alfombra de la familia todo sobre los robots y los ordenadores

Alimentos - comer las cosas ricas que mi mujer hace para mí

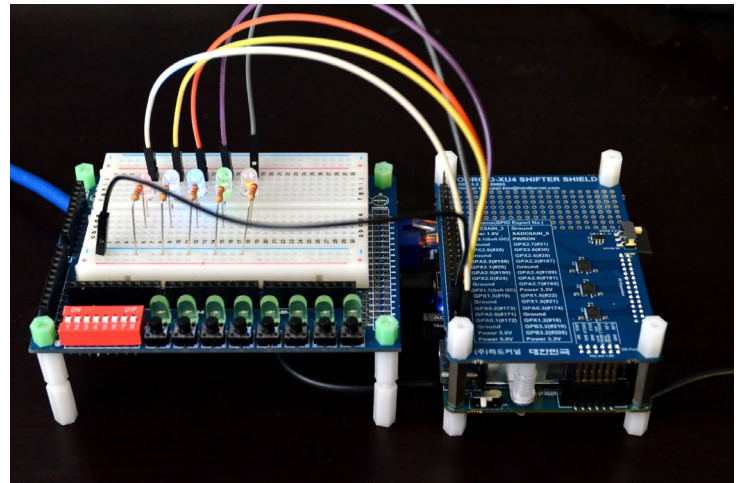
¿Qué tipo de mejoras de hardware te gustaría ver en futuras placas de Hardkernel?

Me gustaría ver USB3.0, SATA, DPI, MPI, un mejor Gig-E, más memoria y GPIO, y soporte para multidispositivos. No pido mucho.

¿Qué consejo le darías a alguien que quiere aprender más sobre la programación y/o hardware?

Buscar un proyecto que esté interesado en hacer. Luego, elegir un idioma y ¡zambullirse y hacerlo! Recuerda que Google es tu amigo, y úsalo para localizar proyectos similares al que quieres hacer, y seguir sus ejemplos de cómo hacer realidad tus proyectos. Hay muchos y excelentes ejemplos por ahí, y algunos no tan buenos.

Pienso que los diagramas de flujo y herramientas gráficas no son muy útiles para aprender, excepto a un nivel elemental. Si te tiene enganchado y quiere aprender más, creo que C es incluso más fácil de entender que C++ y sus alternativas. Por

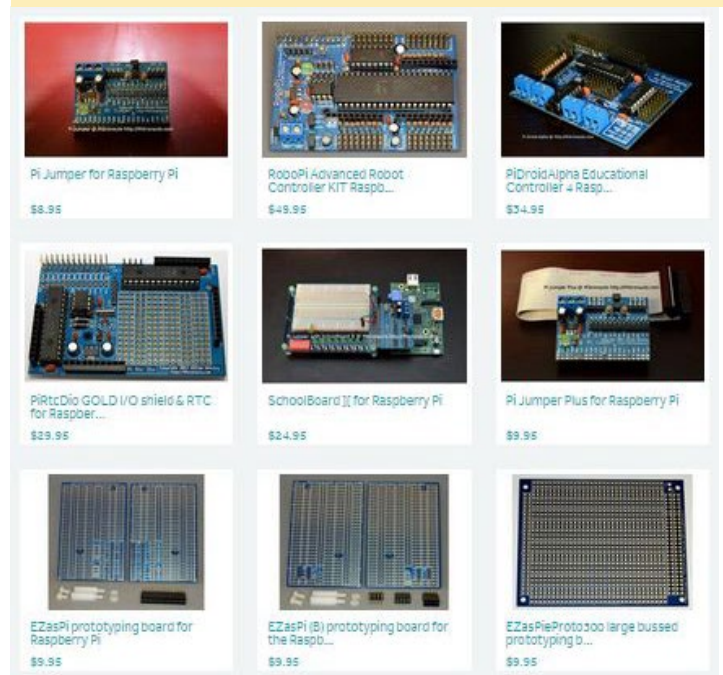


SchoolBoard - placa de desarrollo con Shifter Shield para ODROID-XU4 y cinco del LED superillantes

supuesto, si realmente quieres entender los ordenadores, en algún momento tendrás que conocer el lenguaje ensamblador y la electrónica. Aprende a soldar, luego, crea tus propias placas desde cero.

No creo que nadie te diga que no necesitas entender la programación de bajo nivel. Si no entiende el lenguaje ensamblador y los indicadores, ninguna plantilla y librería C++ te ayudará a comprender por qué tu código no funciona correctamente. Necesitarás osciloscopios, analizadores lógicos y entender muy bien cómo funciona todo para ser capaz realmente de depurar código embebido. ¡Y asegúrate de divertirte!

Echa un vistazo a ebay Mikronauts de Williams o las tiendas Tindy visitando www.mikronauts.com





Conference Overview



REGISTRATION >>

CONNECT WITH ARM TECHCON



Hashtag: #ARMTechCon

Get Ready for ARM TechCon 2015, November 10-12 in Santa Clara, CA!

ARM TechCon Conference Hours

Tue, November 10: 8:30 am – 4:20 pm
 Wed, November 11: 8:30 am – 5:30 pm
 Thu, November 12: 8:30 am – 5:30 pm

ARM TechCon Expo Hours

Tue, November 10: Expo Floor Closed
 Wed, November 11: 11:00 am – 6:30 pm
 *Happy hour: 5:30 pm to 6:30 pm
 Thu, November 12: 11:00 am – 6:30 pm
 *Happy hour: 5:30 pm to 6:30 pm



HARDKERNEL

2015 Event At A Glance

ARM TechCon 2015 Event At-A-Glance

Why Attend?

Who Attends ARM Techcon?

Tracks & Topics

Why Attend?

Join us at ARM TechCon 2015 and:

- Get exclusive access to trending design strategies, methodologies, and tools for building ARM®-based products through technical sessions, hands-on labs, exhibits, demonstrations and keynote and panel discussions.
- Network with fellow design engineers and software developers shaping the future with ARM technology.
- Make a major impact on tomorrow's technology when you collaborate directly with ARM® Ecosystem Partners that are part of an exclusive deep and diverse ARM Connected Community®.
- Meet with ARM Executives, System Architects, and Fellows to assist you with your ARM-based strategies.

2015 SPONSORS

Diamond Sponsors



Platinum Sponsors



Silver Sponsors

