VLs con Franjas • Solarus • Juegos Linux • Desarrollo Android • Gradle

ODROID Magazine

Año Dos
Num. #24
Dec 2015

Robot Portero

El futuro del deporte esta próximo y será controlado por un ODROID

Multi-Arranca tu ODROID-XU4 con facilidad

Un método práctico para ejecutar Contenedores Linux

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

Pollin ELECTRONIC

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

EDITORIAL

Finalmente el día llego cuando los robots reemplazaron totalmente a los seres humanos, ¡Al menos en el campo de fútbol! Conoce a Faro, el portero robotizado impulsado por ODROID, que forma parte de un equipo de robots humanoides avanzados que pertenecen a la Robocup Foundation, cuyo objetivo es derrotar con el tiempo al equipo de fútbol ganador de la Copa del Mundo. Usando un ODROID-C1, el Gadjah Mada Robotic Team ha desarrollado algoritmos que trabajan con una cámara, una brújula y unos sensores giroscópicos para predecir la trayectoria de la pelota en tiempo real e impedir que se marque un gol.

También celebramos el lanzamiento de Fallout 4 con una revisión de Tobias del juego Fallout original. Los fans de Zelda pueden disfrutar de algunos juegos desarrollados por los fans utilizando Solarus y las nuevas estrellas de rock pueden practicar sus riffs utilizando Frets on Fire, un clon de Guitar Hero de código abierto. Jon presenta un tutorial sobre cómo usar los puertos GPIO para hacer lecturas desde un sensor externo, David continúa su serie sobre Gestión de Volumenes Lógicos, Andrew presenta una guía de Gradle y el NDK de Android, Adrian nos da una visión global de los Contenedores Linux y Hardkernel nos muestra sus nuevas oficinas.

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 articulos, contacta con odroidmagazine@gmail.com, o visita http://bit.ly/1ypImXs.
Únete a la comunidad ODROID con miembros en más de 135 países en http://forum.odroid.com/ y explora las nuevas tecnologías que te ofrece Hardkernel en http://www.hardkernel.com/.

HARDKERNEL

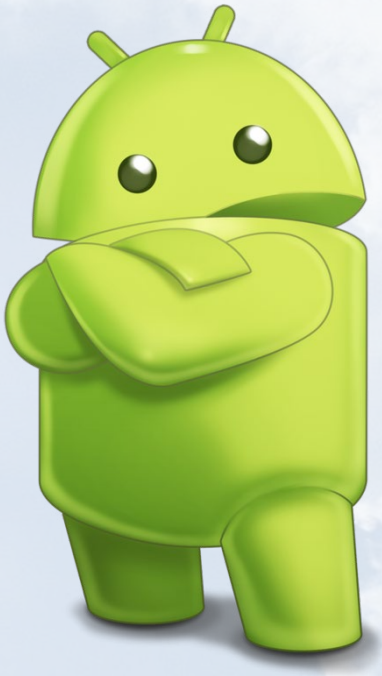
ODROID Magazine

Rob Roy, Editor Jefe

Soy un programador informático que vive y trabaja en San Francisco, CA, en el diseño y desarrollo de aplicaciones web para clients locales sobre mi cluster 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.

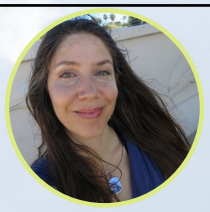
Manuel Adamuz, Editor Español

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.



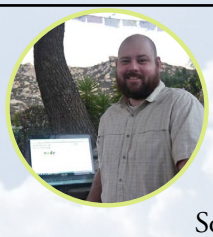
Bruno Doiche, Editor Artístico Senior

¿Qué hacer cuando te tienes que trasladar a tu nueva casa sin dejar trabar para entregar esta revista en plazo? Haces todo cualquier cosa con tu portátil al estilo vagabundo, sentado en el suelo en tu antigua casa que todavía tiene acceso a Internet. Después de todo, este editor artístico chiflado está muy contento con su nueva y humilde morada.



Nicole Scott, Editor Artístico

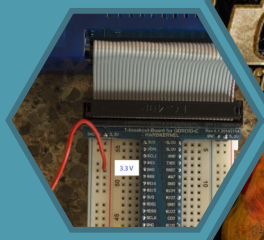
Soy una experta en Producción Transmedia y Estrategia Digital especializa en la optimización online y estrategias de marketing, administración de medios sociales y producción multimedia impresa, web, vídeo y cine. Gestiono múltiples cuentas con agencias y productores de cine, desde Analytics y Adwords 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 enhttp://www.nicolecscott.com.

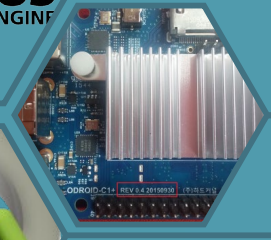


James LeFevour, Editor Artístico

Soy un especialista en m e - dios 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.

INDICE

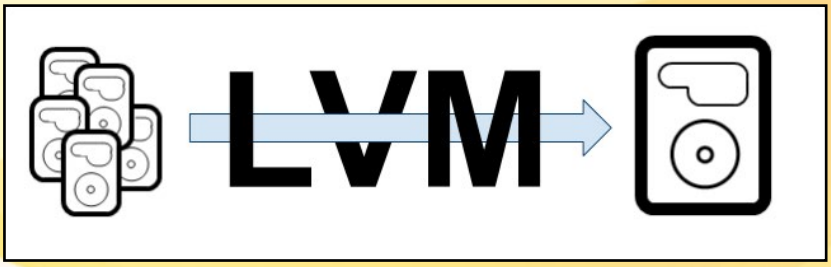




GESTION DE VOLUMENES LOGICOS

MAS ALLA DE LAS BARRERAS DE LA LVM

por David Gabriel

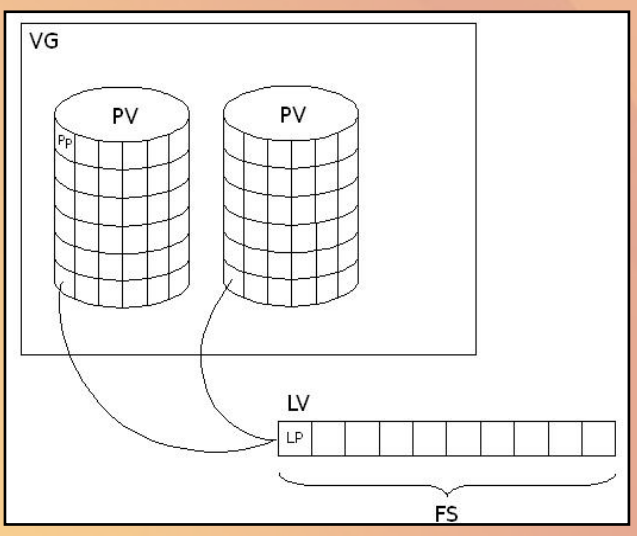


Cuando empiezas a utilizar la Gestión de Volúmenes Lógicos (LVM) fuera de los sistemas personales y domésticos, y la pones en práctica en un entorno empresarial, algunas de sus principales funciones pueden serte de mucha utilidad. En este artículo, voy a describir algunos de los excelentes recursos que nos ofrece la LVM.

Existe una estupenda opción que puede ayudar a mejorar el rendimiento E/S de la LVM: ¡Las franjas! No, no son las líneas en paralelo de tu camisa, sino franjas de datos.

Para crear un volumen lógico (LV) con franjas, necesitarás al menos dos volúmenes físicos. Cuando lo hagas, los trozos de datos llamados extensiones se escribirán a lo largo de todos los volúmenes físicos, cada pieza en un único volumen hasta el último y luego vuelve al primero una y otra vez. En otras palabras, el archivo se dividirá en partes más pequeñas y cada parte se almacenará en un volumen físico diferente, en lugar de tener el fichero guardado en un único volumen, el lineal por defecto. Esto mejora considerablemente la velocidad con la que leemos el archivo, ya que se lee desde diferentes "espacios" físicos de forma simultánea.

El siguiente comando crea un volumen lógico de 10GB distribuido en dos volúmenes físicos, con franjas de 32 kB.




```
$ lvcreate -L 10G -i 2 -I 32 -n applv rootvg
```

Mi consejo es que hagas algunas pruebas guardando archivos al azar en un LV con franjas para ver si esta técnica te interesa. Ten en cuenta que este tipo de configuración es muy sensible a los fallos del dispositivo. Si pierdes alguna unidad donde se almacene una parte del archivo, lo pierdes todo.

Otra opción de volúmenes lógicos es montarlos en espejo. A diferencia de las franjas de datos, el sistema espejo es muy útil cuando aparecen fallos en los dispositivos, puesto que el volumen se mantendrá en funcionamiento si falla uno de los espejos. Cuando guardas datos en el volumen, se crean dos copias de los mismos en dos dispositivos diferentes. Cuando uno de ellos falla, el volumen se convierte en lineal, que es la opción por defecto al crear nuevos volúmenes.

El siguiente comando crea un nuevo LV en espejo:

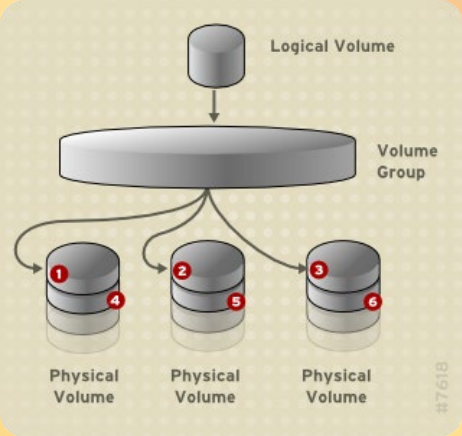
```
$ lvcreate -L 5G -m 1 -n applv rootvg
```

Esto creará dos copias del volumen en dos dispositivos físicos diferentes. También puedes especificar dónde quieres que se creen las copias añadiendo los dispositivos al comando.

```
$ lvcreate -L 5G -m 1 -n applv rootvg /dev/sda1 \
/dev/sdb1 /dev/sdc1
```

Los datos originales irán a /dev/sda1, una copia en espejo irá a /dev/sdb1 y los registros log del sistema espejo irán a /dev/sdc1. Si no especificas el dispositivo para almacenar los registros log, éstos pasarán a la memoria del sistema.

En el caso de que uno de los dispositivos falle, deberías poder continuar utilizando el volumen lógico. Tras sustituir o reparar el dispositivo físico dañado, deberías montar el sistema espejo para volver a tener redundancia de datos. Para ello, es necesario crear de nuevo el volumen físico, añadirlo al grupo de volúmenes y finalmente, convertir el volumen lineal en espejo.




```
$ pvcreate /dev/sda1
$ vgextend rootvg /dev/sda1
$ lvconvert -m 1 \
/dev/rootvg/applv /dev/sda1 \
/dev/sdb1 /dev/sdc1
```

Usa el comando lvs para comprobar que se ha restablecido el sistema espejo.

Ahora, supongamos que tu LVM lleva un tiempo funcionando. Todo está configurado y marcha bien, cuando de repente deja de funcionar. Aparecen mensajes de error o observas comportamientos que nunca ha visto antes. Déjame darte algunos consejos para solventar este tipo de situaciones y solucionar los problemas de tu sistema.

Comprueba la configuración del LVM. Si hiciste cambios en la misma, éste podría ser un buen lugar para empezar.

```
$ lvm dumpconfig
```

El comando lvm tiene diferentes niveles de resultados. Utiliza -v, -vv, -vvv o -vvvv para aumentar estos niveles y así obtener más información.

También es bueno a echar un vistazo a la información del sistema LVM.

```
$ lvmdump
```

Este comando hará un volcado de información con fines de diagnóstico. Echa una ojeada a la guía de lvmdump para obtener una lista completa de los contenidos del archivo.

También puede revisar la información de los siguientes comandos.

```
$ pvs -a
$ lvs -v
$ dmsetup info
```

Estos comandos te dan información adicional del sistema que puede ayudarte con la resolución de problemas.

El servicio LVM realiza copias de seguridad con regularidad de tu configuración y de los metadatos. Guarda información sobre los volúmenes físicos y lógicos, los tamaños de extensión y que volúmenes físicos son utilizado por los volúmenes lógicos. Por defecto, este archivo se almacena en /etc/lvm/backup y algunas versiones más antiguas lo guardan en /etc/lvm/archive. Es bueno tener una copia de este archivo fuera del sistema. Si algo sale mal, no podrás acceder al mismo para recuperar los metadatos del LVM. También puede realizar copias de seguridad manuales de los metadatos con el siguiente comando.

```
$ vgcfgbackup
```

Este comando es muy simple. Puedes especificar dónde quieres guardar el archivo con el argumento -f. Recuerda que esto sólo hace una backup de los metadatos, y no de los archivos y directorios reales que están dentro de los volúmenes.

Los mensajes que puedes recibir cuando faltan o se dañan los metadatos son: “No se pudo localizar el dispositivo con uuid” o “No se pudieron localizar todos los volúmenes físicos para el grupo de volúmenes”. En estos casos, restaurar la configuración sería de gran ayuda.

```
$ vgcfgrestore rootvg
```

Puede que sea necesario activar los volúmenes lógicos tras la restauración.

```
$ lvchange -ay /dev/rootvg/applv
```

Tras completarse este comando, debería tener restaurados tus volúmenes lógicos. Lo puedes comprobar con el comando lvs.

LAS NUEVAS OFICINAS DE HARDKERNEL

UN RAPIDO RECORRIDO POR LA SEDE CENTRAL

editado por Rob Roy

Hardkernel con sede en Corea del Sur, anunciaba hace poco que se mudaba a unas nuevas oficinas. Algunos de los miembros de su equipo trabajan en otros países, pero la mayoría del personal se encuentra en la ciudad de Anyang. Aunque los ODROIDs en sí se fabrican en una fábrica cercana, el diseño del hardware, el desarrollo del código, las pruebas, el envío y la administración se realizan desde la sede central ubicada en el 475-1 de Mananro, Manangu, Anyang, Gyeonggi, Corea del Sur 430-852. @pinkodroid ha compartido algunas fotos de las modernas oficinas a través del blog de Hardkernel http://bit.ly/1ONVPmw.

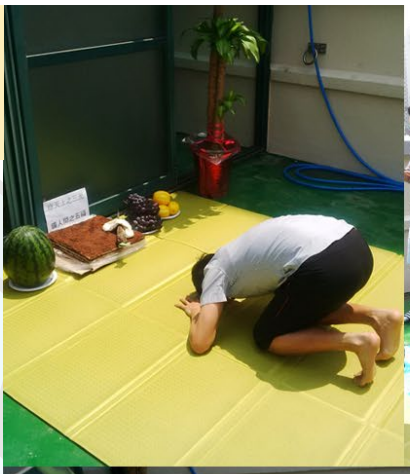
El exterior de las oficinas de Hardkernel parece muy moderno

La primera planta de la nueva oficina es un almacén tipo dúplex

Un jardín en la azotea, ideal para comer cuando el tiempo lo permite

La oficina en la segunda planta, donde trabajan los desarrolladores

El chamanismo es una parte importante de la cultura coreana



El primer día, el equipo se reunió en la azotea para celebrarlo

Siguiendo la tradición coreana, se preparó bastante comida

Se hacen oraciones para el éxito continuo de Hardkernel

Los empleados de Hardkernel disfrutan comiendo juntos tras un largo día

CONTENEDORES LINUX

MONTAR RAPIDAMENTE UN SISTEMA TOTALMENTE AISLADO Y CONFIGURADO PARA REALIZAR PRUEBAS

por Adrian Popa

Has seguido adelante, has comprado un ODROID y te gusta. ¿No quieres más? Quizás necesites un sistema aislado para hacer pruebas o desees ejecutar múltiples versiones de algún software. Tal vez quieras ejecutar diferentes pruebas de red y necesites simular varios clientes independientes. Si la capacidad de procesamiento no es un problema para ti, los contenedores Linux inspirados en Solaris, pueden hacer un gran trabajo en tu ODROID.

Cómo funciona

LXC es un liviano método de virtualización para ejecutar múltiples unidades virtuales de forma simultánea llamadas contenedores, similares a chroot, sobre un único host de control. Los contenedores están aislados por Grupos de Control del Kernel, llamados cgroups y Namespaces. Permite una virtualización a nivel de sistema operativo en la que el kernel controla los contenedores aislados. En otras soluciones más completas de virtualización como son Xen, KVM y libvirt, el procesador simula un completo entorno de hardware y controla sus máquinas virtuales.

Conceptualmente, LXC puede verse como una técnica chroot mejorada. La diferencia es que un entorno chroot sólo separa el sistema de archivos, mientras que LXC va más allá permitiendo la gestión y control de los recursos mediante cgroups.

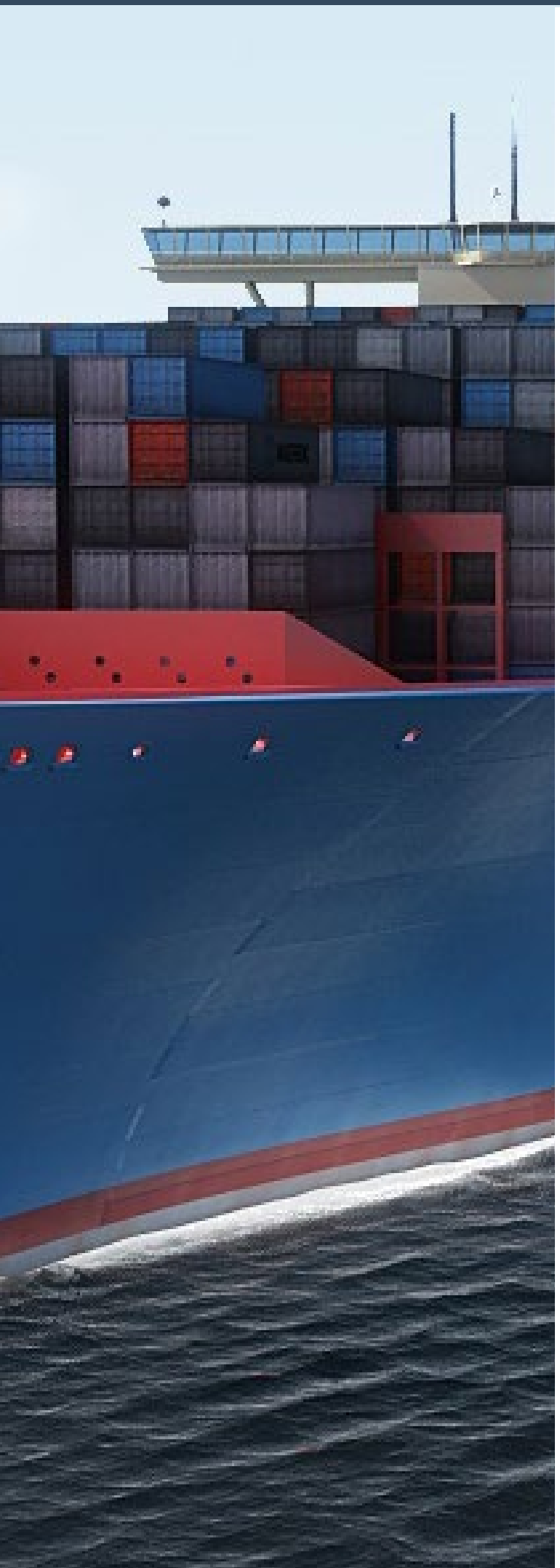
Ventajas a la hora de usar los Contenedores Linux

Aislamiento de aplicaciones y SOs a través de contenedores.
Proporciona un rendimiento casi nativo puesto que LXC gestiona la asignación de los recursos en tiempo real.
Permite controlar las interfaces de red y la aplicación de recursos dentro de contenedores a través de cgroups.

Limitaciones a la hora de usar los Contenedores Linux

Todos los contenedores LXC se ejecutan dentro del kernel del sistema host y no con un kernel diferente.
Sólo permite sistemas operativos Linux "invitados".
LXC no es un completo sistema de virtualización, como lo son Xen, KVM y libvirt.
La seguridad depende del sistema host y LXC no es seguro. Si necesitas un sistema seguro, utilizar KVM.

Máquinas virtuales vs Contenedores



Para instalar y utilizar LXC, simplemente utiliza apt-get para instalar el paquete LXC:

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

Inicialización

Para crear o utilizar contenedores Linux, necesitarás privilegios de root. Los contenedores privados también pueden ser creados por usuarios sin privilegios, pero para que esto sea posible, es necesario al menos la versión 3.13 del kernel. Sin embargo, la imagen de Ubuntu 15.04 de HardKernel viene con la versión 3.10 del kernel. Además de root, también necesitarás hacer algunas configuraciones específicas en el kernel si has compilado tu propio Kernel. El kernel por defecto de HardKernel viene con todo lo necesario para ejecutar LXC sin problemas. Para comprobar que tu sistema está listo para LXC, ejecuta el comando lxc-checkconfig. Si tu kernel no soporta lxc, consulta el artículo Docker del número de Enero del 2015 de ODROID Magazine en http://bit.ly/1OfT2zo para saber lo que necesitas activar en la configuración del kernel.

Para crear un nuevo contenedor, primero tienes que preparar un archivo de configuración inicial y seleccionar la plantilla adecuada. Las plantillas enseñan a LXC cómo descargar los paquetes necesarios para la distribución que elijas. En Ubuntu 15.04 tienes estas plantillas por defecto:

```
# ls /usr/share/lxc/templates/
lxc-alpine     lxc-centos    lxc-fedora        lxc-
oracle  lxc-ubuntu-cloud
lxc-altlinux   lxc-cirros    lxc-gentoo        lxc-
plamo
lxc-archlinux  lxc-debian    lxc-openmandriva  lxc-
sshd
lxc-busybox    lxc-download  lxc-opensuse      lxc-
ubuntu
```

Por desgracia, puesto que no estás ejecutando una plataforma Intel, no todas funcionan en nuestros ODROIDs. Para saber si tu distribución favorita se ejecutará como un contenedor, es necesario conocer si también permite la compilación ARM. Primero vamos a crear un nuevo contenedor para Fedora Linux. Queremos que la configuración de red vaya por lxcbr0 que hará de NAT, de modo que vamos a ajustar la configuración de la siguiente manera:

```
# cat fedora.conf
lxc.utsname = fedoracontainer
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = lxcbr0
lxc.network.name = eth0
```

La configuración sólo especifica que el contenedor se llamará internamente “fedoracontainer” como nombre de host, que el sistema de redes está conectado a lxcbr0 y que el nombre de la interfaz de la red interna es eth0. Por desgracia, la plantilla para Fedora está desactualizada y da un error por defecto, de modo que necesitas corregir algunas rutas:

```
# sed -i 's/mirrorurl="mirrors.kernel.org::fedora"\
/mirrorurl="mirrors.kernel.org::archive\/\
fedora-archive"/' /usr/share/lxc/templates/lxc-fedora
```

Para crear el contenedor, usa este comando:

```
# lxc-create -t /usr/share/lxc/templates/lxc-fedora \
-f fedora.conf -n fedoracontainer -- --release 23
```

El parámetro -t específica la plantilla que será utilizada, -f apunta al fichero de configuración que acabamos de crear y -n determina el nombre del contenedor. El símbolo -- informa a lxc-create para que pase cualquier parámetro adicional a la plantilla (es un script bash), y que estamos solicitando Fedora 23. El script de arranque lo tiene en cuenta y descarga una versión “livecd” de Fedora 20 y la utilizará para instalar una mínima imagen de Fedora 23. Todo el proceso puede llevar un tiempo. Uva vez creado el contenedor, puedes eliminar la rutina de arranque y la memoria caché si no tienes pensado instalar otros contenedores basados en fedora a corto plazo

```
# rm -rf /var/cache/lxc/fedora/armhfp/bootstrap
# rm -rf /var/cache/lxc/fedora/armhfp/LiveOS
```

Justo ahora, el contenedor se cierra y guarda sus archivos, incluyendo la configuración del contenedor, en el directorio /var/lib/lxc/fedoracontainer/ que usa 657MB para la instalación.

Vamos a indagar cómo usar Ubuntu 15.10 como contenedor. He oído que todavía tiene problemas cuando lo ejecutas como sistema operativo principal en ODROID, vamos a verlo.

En primer lugar, prepara el archivo de configuración inicial. Es similar a Fedora, pero esta vez queremos que el contenedor conecte a eth0, por lo que tendrás que crear una interfaz puente que conectará a eth0 y que llamaremos brlan0.

Cambiar tu conexión de red por cable a una interfaz puente puede resultar complicado si lo haces de forma remota. La mejor manera de hacerlo siendo persistente con los reinicios es añadir esta configuración a /etc/network/interfaces, luego reinicia tu ODROID:

```
auto brlan0
iface brlan0 inet dhcp
   bridge_ports eth0
```

Puenteando el adaptador inalámbrico

```
root@procyon:~# lxc-ls --fancy
NAME             STATE    IPV4        IPV6  GROUPS  AUTOSTART
-----------------------------------------------------------------
fedoracontainer  RUNNING  10.0.3.186  -     -       NO
ubuntucontainer  STOPPED  -           -     -       NO
root@procyon:~# lxc-attach -n fedoracontainer
[root@fedoracontainer ~]# uptime
 11:57:47 up 10 min,  0 users,  load average: 1.31, 1.70, 1.63
[root@fedoracontainer ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 11:46 ?        00:00:15 /sbin/init
root        82     1  0 11:47 ?        00:00:05 /usr/lib/systemd/systemd-journa$
dbus       245     1  0 11:48 ?        00:00:01 /usr/bin/dbus-daemon --system -$
root       255     1  0 11:48 ?        00:00:01 /usr/sbin/rsyslogd -n
root       292     1  0 11:48 ?        00:00:00 /usr/lib/systemd/systemd-logind
root       460     1  0 11:48 lxc/console 00:00:00 /sbin/agetty --noclear --kee$
root       519     1  0 11:49 ?        00:00:00 /sbin/dhclient -H fedoracontain$
root       783     1  0 11:50 ?        00:00:00 /usr/sbin/sshd -D
root       914     1  0 11:50 pts/1    00:00:00 /sbin/agetty --noclear --keep-b$
root       935     1  0 11:51 ?        00:00:00 /sbin/agetty --noclear tty2 lin$
root      1005     1  0 11:51 ?        00:00:00 /sbin/agetty --noclear --keep-b$
root      1010     1  0 11:51 lxc/tty3 00:00:00 /sbin/agetty --noclear tty3 lin$
root      1075     1  0 11:51 lxc/tty1 00:00:00 /sbin/agetty --noclear tty1 lin$
root      1080     1  0 11:51 ?        00:00:00 /sbin/agetty --noclear --keep-b$
root      2090     0  0 11:57 pts/2    00:00:00 /bin/bash
root      2128     1  0 11:57 ?        00:00:00 [agetty] <defunct>
root      2144     1  0 11:57 ?        00:00:00 (agetty)
root      2149  2090  0 11:57 pts/2    00:00:00 ps -ef
[root@fedoracontainer ~]#
[root@fedoracontainer ~]# ip addr show eth0
8: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP g$
oup default qlen 1000
    link/ether fe:40:03:17:b5:71 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.186/24 brd 10.0.3.255 scope global dynamic eth0
       valid_lft 3003sec preferred_lft 3003sec
    inet6 fe80::fc40:3ff:fe17:b571/64 scope link
       valid_lft forever preferred_lft forever
[root@fedoracontainer ~]# ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=17.2 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=16.8 ms
```

Ten en cuenta que la interfaz brlan0 se convertirá en la interfaz de capa 3 en tu sistema y cogerá una dirección IP, y eth0 sólo será un puerto conmutador de capa 2. Además, el cambio de la configuración de la red podría interrumpir procesos en los que la interfaz está especificada por el nombre, tales como iptables, arpwatch y munin.

Una vez que la interfaz puente esté instalada y funcionando tras reiniciar el ODROID, utiliza esta configuración para preparar el contenedor:

```
# cat ubuntu.conf
lxc.utsname = ubuntucontainer
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = brlan0
lxc.network.name = eth0
# lxc-create -t /usr/share/lxc/templates/lxc-ubuntu \
-f ubuntu.conf  -n ubuntucontainer -- --release wily
```

La instalación mínima 10,15 utiliza solamente 326MB.

Si estás pensando en puentear tu adaptador inalámbrico, la mala noticia es que no puedes, tal y como se comenta en http://bit.ly/1WZNdOb. Esto es porque el controlador inalámbrico puede crear múltiples interfaces lógicas (como wlan0), y no puedes mover la interfaz lógica a un namespace diferente sin mover toda la tarjeta de red. Sin embargo, LXC ofrece un mecanismo para separar la tarjeta de red de tu sistema host y añadirle un contenedor en funcionamiento:

```
# lxc-device -n container-name add wlan0
```

Una vez que la tarjeta wifi este conectada al contenedor, ya no será visible en el sistema operativo host, de modo que necesitas una alternativa para conectarte a ella.

Iniciar y detener

Ahora que tienes dos contenedores, es el momento de arrancarlos. Esto se puede hacer con el siguiente comando:

```
# lxc-start -n fedoracontainer -d
```

El parámetro -d le dice al comando que inicie el contenedor en segundo plano, de lo contrario debería conectar tu terminal a su consola, y matando el terminal también finalizaría el contenedor. Si tienes problemas para iniciar tu contenedor, quita el parámetro -d para seguir los mensajes del arranque. Para conectar al contenedor y hacer algo de trabajo, usa lxc-attach:

```
# lxc-attach -n fedoracontainer
[root@fedoracontainer ~]#
```

Puedes conectarte antes de que el contenedor se haya iniciado por completo, por lo que algunos servicios como los servicios de red, puede que no estén disponibles. Simplemente espera a que el contenedor se inicie totalmente, lo cual se indica con el proceso getty en tu lista de procesos. Si le echas un vistazo, verás que sólo hay unos cuantos procesos ejecutándose. Después, puede utilizar yum para instalar tus paquetes favoritos, como si el sistema se ejecutara sobre el propio hardware.

Para salir del contenedor sin detenerlo, simplemente escribe exit en el prompt. También puedes acceder al contenedor vía ssh desde el host a través de la red interna. Para apagar un contenedor, puede usar el comando lxc-stop:

```
# lxc-stop -n fedoracontainer
```

Si quieres que tu contenedor se inicie junto con el sistema, puede activar la función de auto inicio modificando la configuración del contenedor ubicada en /var/lib/lxc/fedoracontainer /config y añadir las siguientes líneas:

```
lxc.start.auto = 1
lxc.start.delay = 10
```

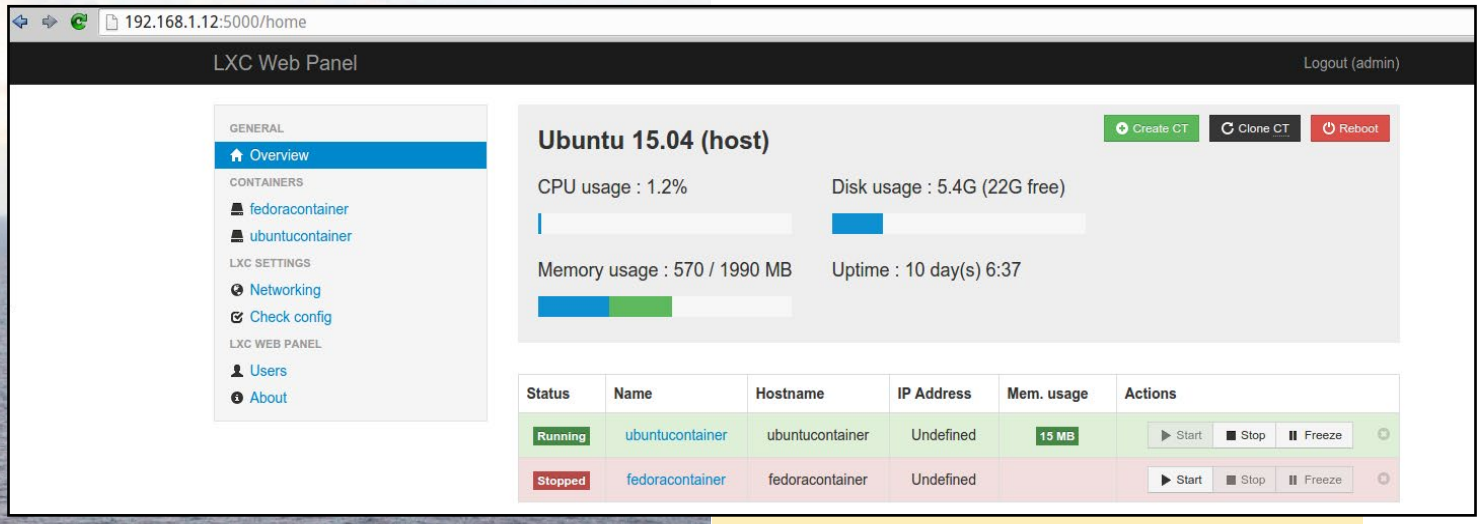
El comando lxc-ls te mostrará si está programado el inicio automático:

```
# lxc-ls --fancy
NAME          STATE   IPV4  IPV6  GROUPS  AUTOSTART
-----------------------------------------------------
---
fedoracontainer  STOPPED  -  -       -       YES
ubuntucontainer  STOPPED  -  -       -       NO
```

Para obtener información adicional sobre tu contenedor en funcionamiento, puedes utilizar el siguiente comando:

```
# lxc-info -n fedoracontainer
Name:           fedoracontainer
State:          RUNNING
PID:            24396
IP:             10.0.3.186
CPU use:        41.87 seconds
Memory use:     15.09 MiB
Link:           vethTSW172
 TX bytes:      3.27 KiB
 RX bytes:      28.89 KiB
 Total bytes:   32.16 KiB
```

Contenedor LXC en funcionamiento



Interfaz web para la gestión de contenedores

Si no te gustar trabajar con líneas de comandos, también puedes administrar tus contenedores desde una interfaz web llamada lxc-webpanel: https://lxc-webpanel.github.io/install.html. Tras seguir las instrucciones del enlace para instalarla, puede acceder a la misma con la dirección http://ip-de-tu-ODROID:5000/. Te permite realizar la mayor parte de las tareas ya mencionadas y explorar algunas opciones avanzadas.

Opciones avanzadas

Con la configuración mostrada hasta ahora, puedes empezar a usar LXC sin añadir demasiada complejidad a tu instalación. Sin embargo, los contenedores presentan una gran flexibilidad en relación al control de asignación de recursos que vamos a comentar brevemente a continuación.

Espacio del disco

Los contenedores que acabas de crear usan un directorio del sistema para almacenar su sistema de archivos root. Aunque esto es fácil de implementar y entender, ofrece un rendimiento E/S mediocre. Otras opciones incluyen un dispositivo de bloqueo LVM, un dispositivo de bloqueo de bucle (que puede ser un archivo o un dispositivo de almacenamiento físico), sistema de archivos brtfs o zfs. Estos te permiten especificar un tamaño máximo y además, ofrecen características para las capturas de pantalla, la eliminación de duplicados y la rápida clonación (copy-on-write). Si fuera necesario, también puede limitar la cantidad de operaciones de E/S que al contenedor le está permitido hacer para que no afecten a otros contenedores o al host.

Memoria

Para ver la memoria que utiliza actualmente un contenedor en funcionamiento, puedes ejecutar el siguiente comando:

```
# cat /sys/fs/cgroup/memory/lxc/\
ubuntucontainer/memory.usage_in_bytes
20341040
```

Por defecto, el contenedor es capaz de utilizar toda la memoria del sistema. Si no quieres que lo haga, puede limitar la memoria con los siguientes comandos:

```
# lxc-cgroup -n ubuntucontainer memory.limit_in_bytes
40M
# cat /sys/fs/cgroup/memory/lxc/\
ubuntucontainer/memory.limit_in_bytes
41943040
```

Los cambios se reflejarán inmediatamente en tu contenedor en funcionamiento:

```
root@ubuntucontainer:~# free -m
             total       used       free     shared
buffers       cached
Mem:            40         31          8         31
0             23
-/+ buffers/cache:          7         32
Swap:          0          0          0
```

Si excedes el límite de memoria, el kernel intentará eliminar memoria caché, pero al final si no hay más memoria, verás que se detienen procesos con el error “No se puede asignar memoria”

```
root@ubuntucontainer:~# mount -t tmpfs -o size=50m
tmpfs /mnt/ramdisk3
root@ubuntucontainer:~# dd if=/dev/zero \
of=/mnt/ramdisk3/1
dd: writing to ‘/mnt/ramdisk3/1’: Cannot allocate
memory
```

Otra posibilidad es añadir el límite de memoria directamente a la configuración del contenedor incluyendo la siguiente línea:

```
lxc.cgroup.memory.limit_in_bytes = 40M
```

CPU

Puedes asignar núcleos específicos de la CPU a un contenedor, o asignar un número limitado de recursos de la CPU a ese contenedor para restringir el uso de CPU. Por defecto cada contenedor recibe 1024 porciones de recursos:

```
# cat /sys/fs/cgroup/cpu/lxc/\
ubuntucontainer/cpu.shares
1024
# echo 256 > /sys/fs/cgroup/cpu/lxc/\
ubuntucontainer/cpu.shares
# cat /sys/fs/cgroup/cpu/lxc/\
ubuntucontainer/cpu.shares
256
```

Para fijarlo en la configuración del contenedor, agrega las siguientes directrices:

```
lxc.cgroup.cpuset.cpus = 1,2
lxc.cgroup.cpu.shares = 256
```

Módulos del kernel

Para utilizar módulos específicos del kernel como iptables dentro de un contenedor LXC, primero tienes que cargar ese módulo en el host.

Ficheros especiales

Es similar a la configuración especial que se necesita para añadir una interfaz wifi a un LXC en funcionamiento, puedes vincular archivos especiales desde el host para ser utilizados exclusivamente por el contenedor. Por ejemplo, para poder utilizar un adaptador USB-Serial en el contenedor, puedes ejecutar este comando en el host:

```
# lxc-device add -n ubuntucontainer \
/dev/ttyUSB0 /dev/ttyS0
```

Posibles usos

Los contenedores pueden ser muy útiles como sistemas de pruebas donde puedes experimentar sin riesgo de romper cosas. Puedes dar acceso root a tus amigos y compartir varios entornos independientes sobre el mismo hardware.

Aprendí a utilizar LXC y compre algunos ODROIDs para realizar pruebas de red utilizando múltiples tarjetas de red desde diferentes ubicaciones. Mi jefe ejecutó varias pruebas con SmokePing sobre múltiples proveedores para medir el tiempo de respuesta web, la descarga de videos de Youtube y los resultados de Speedtest.net desde dos contenedores independientes ejecutados sobre un ODROID. Los contenedores nos permitieron usar redes puenteadas para acceder a recursos remotos a través de ambos enlaces simultáneamente y manteniendo diferentes tablas de enrutamiento. Debido a que la aplicación no necesita muchos ciclos de CPU o memoria, los ODROIDs eran perfectos para la tarea.

Mi plan para el futuro es conseguir ejecutar Android dentro de un contenedor, según el artículo disponible en `http://bit.ly/1QO855N` es posible. Diviértete jugando con los contenedores, y no dudes en publicar tus comentarios en el hilo de soporte en `http://bit.ly/1PANHG0` si logras usar otros contenedores.

Lecturas recomendadas

Pequeño tutorial sobre LXC: `http://bit.ly/1WZO8ht`

Guía avanzada: `http://bit.ly/1S5j9tW`

FARO

EL ROBOT PORTERO HUMANOIDE

por Ilham Imaduddin

La tendencia de los robots de fútbol es realmente fascinante hoy en día. La gente de la Robocup Federation incluso llegaron a decir que "A mediados del siglo 21, un equipo de jugadores de fútbol formado por robots humanoides totalmente autónomos debería ganar un partido de fútbol, cumpliendo las normas oficiales de la FIFA, jugando contra los ganadores de la última Copa del Mundo".

Bueno, eso sí que es un sueño ambicioso. Aunque la tecnología de hoy en día está muy lejos de este objetivo, el desarrollo de la robótica de fútbol cuenta con una investigación muy activa en numeroreos laboratorios y universidades. En la Universidad de Gadjah Mada en Indonesia, el Gadjah Mada Robotic Team (GMRT) está desarrollando un equipo de fútbol con robots del tamaño de un niño. En este artículo, me centraré en Faro, el portero que usa ODROID-C1 como cerebro.

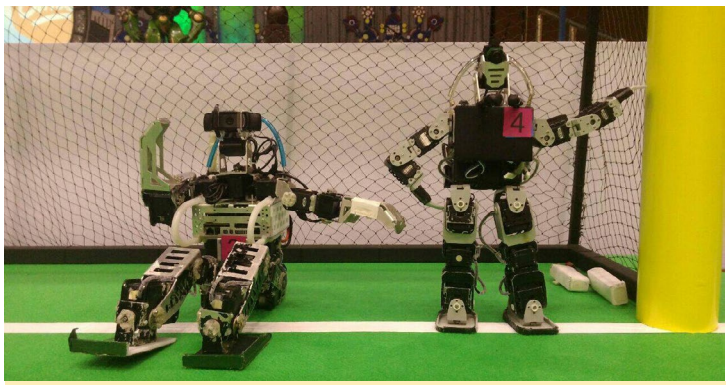


Figura 1 - Faro es el de la derecha

Hardware

Faro está creado con las siguientes piezas y componentes:

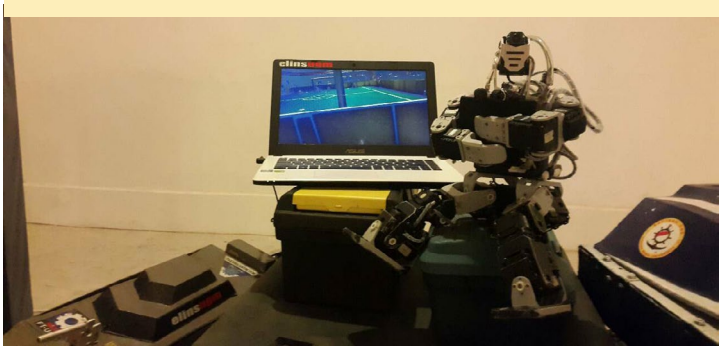
- ODROID-C1 con Ubuntu 14.04
- Controlador CM-530
- Cámara Logitech C905
- 12 actuadores Dynamixel AX-18A
- 8 actuadores Dynamixel AX-12A
- Batería LiPo de 3 celdas de 1800 mAh
- Sensor de giro
- Brújula
- Estructura Bioloid para robots

Las partes mecánicas de Faro son compatibles con una estructura Bioloid, que es un kit para robot de Robotis. Las manos tienen 3 DOF (grados de libertad) y las piernas 6 DOF. Se necesitan estos DOFs adicionales para una mayor flexibilidad a la hora de maniobrar. En cada mano se usan tres servomotores AX-12A, mientras que cada pierna utiliza seis servomotores AX-18A para una mayor potencia. Se usan otros dos servomotores AX-12A para apoyar la cámara sobre la cabeza de Faro.

Estos servomotores son controlados por el controlador CM-530, que está conectado al controlador principal por una interfaz USB. Cada movimiento está programado en este controlador. Se recibe una secuencia de comandos de acción desde el controlador principal para determinar qué movimiento utilizar. También tiene un sensor de giro para estabilizar el cuerpo.

Se utiliza un ODROID-C1 como controlador principal. La cámara Logitech C905 está conectado al puerto USB del C1. Esta cámara es el sensor principal de Faro. Permite que Faro vea el mundo, o al menos el campo de fútbol. El ODROID-C1 también tiene conectada una brújula que proporciona información sobre la dirección del cuerpo. Esta asegura que Faro esté de cara a la portería contraria.

Figura 2 – Soy muy chulo, ¿verdad?



La energía procede de una batería LiPo de 3 celdas (11,1 V), que incorpora dos salidas. Una se conecta directamente al controlador CM- 530 y la otra regulada a 5V alimenta el ODROID-C1.

Algoritmo

La tarea de Faro es muy simple: evitar que el balón entre en la portería. Para lograr esto, se aplican varios procesos.

El primer proceso es detectar la pelota. Esto se realiza capturando vídeo desde la cámara y localizando una pelota teniendo en cuenta una serie de características como el color, la forma, los bordes y otros parámetros. La pelota es de color naranja, con un diámetro de 10 cm aproximadamente. Tras localizar la pelota, Faro empieza a seguir el movimiento de la pelota.

Al rastrear el movimiento de la pelota, Faro puede predecir y sentir el peligro de que ésta llegue a su portería. Cuando Faro detecta la pelota moviéndose hacia su posición, contabiliza y predice la trayectoria de la pelota y hace una acción basada en esta predicción, como quedarse quieto o saltar para evitar que la pelota entre en la portería.

Todos estos procesos se realizan dentro del ODROID-C1 con la ayuda de la librería OpenCV. A pesar de que el pro-cesamiento de imágenes requiere de mucha capacidad de procesamiento, el ODROID-C1 es lo suficientemente potente como para soportar la demanda de trabajo.

Si Faro cree que una pelota está amenazando la portería, enviará un comando de acción al controlador CM-530. Éste incluye una serie de movimientos: lanzarse a la izquierda, lanzarse a la derecha o levantarse desde una posición inferior.

Todos estos movimientos se programan en el controlador CM-530, por lo que cuando el CM-530 recibe un comando de acción, elije el movimiento más adecuado y contabilizará cada ángulo del servomotor. Por último, el CM-530 avisará al servomotor para que realice el movimiento.

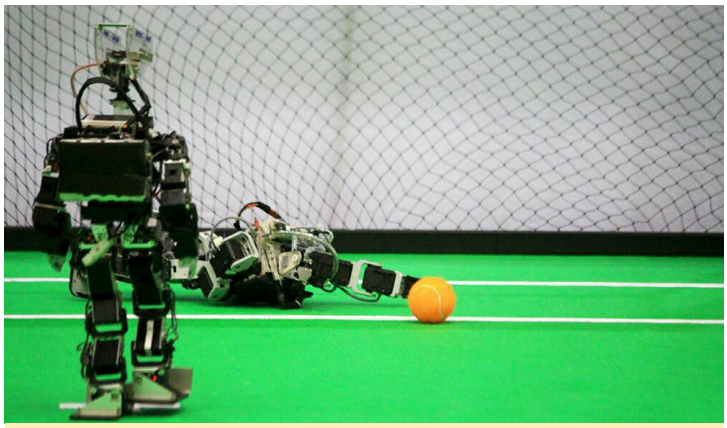


Figura 3 – Protegiendo la porteria

Mientras realiza la rutina de seguimiento de la pelota, Faro se comunica con otro jugador de equipo con el cual se coordina. Cada jugador del equipo compartirá información sobre lo que detecta, especialmente, la posición de la pelota. También comparten la acción que están haciendo en ese momento. Esto permite al equipo coordinarse y hacer su trabajo en equipo, ejecutando la mejor estrategia para vencer a su oponente. La comunicación se realiza a través de una red Wi-Fi.

Uno de los mayores desafíos para un robot portero es, por supuesto, asegurarse de coger el balón. El problema surge cuando la pelota se mueve más rápido que la propia capacidad del robot para detectarla y atraparla en el momento. En ese caso, el sistema no puede seguir la pelota, de modo que se queda quieto sin saber que acaban de marcarle un gol, o el sistema podría seguir la pelota pero lo hace demasiado tarde, se mueve para atraparla pero ésta ya lo ha rebasado en cualquier caso.

El tiempo de procesamiento de todos estas rutinas es lo suficientemente pequeño para que Faro atrape la pelota a tiempo si la velocidad de la pelota es normal, pero algunas ocasiones falla (incluso los seres humanos fallamos a veces, ¿no? Por lo tanto, sería bueno mejorar el tiempo de respuesta usando un ordenador más potente, como el ODROID-XU4. Pero incluso con más capacidad de procesamiento no se puede garantizar que el robot siempre atrape la pelota. La capacidad física del robot también juega un papel importante. Los robots necesitan la suficiente potencia y velocidad para interpretar los comandos, y así poder atrapar la pelota a tiempo y en última instancia, proteger la meta ante el ataque de cualquier rival. ¿En qué momento lo saben sin tener que actuar, de todos modos?

El desarrollo de robots humanoides como equipo de fútbol está lejos de completarse. Los equipos de fútbol con robots de hoy en día son fácilmente derrotados por niños. Se necesita más capacidad de procesamiento, algoritmos más complejos, y más potencial físico para superar las capacidades de los seres humanos. Es necesario investigar mucho más para alcanzar el objetivo final: Ganar un partido de fútbol contra los campeones de la Copa del Mundo.

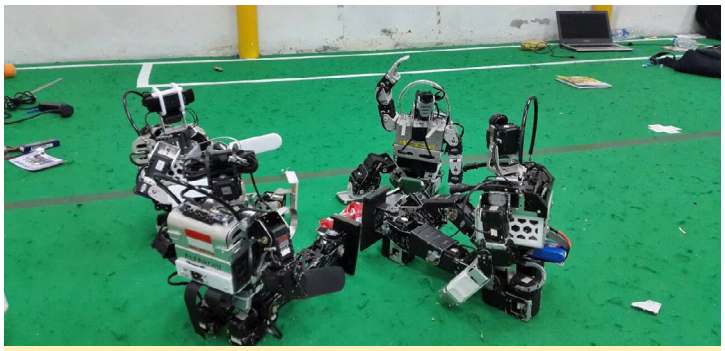


Figura 4 - ¡Saludos!

USANDO ANDROID NDK EN ANDROID STUDIO Y GRADLE

TRABAJANDO CON WIRINGPI EN ANDROID

por Andrew Ruggeri

En este artículo vamos a ver cómo utilizar Android NDK dentro del sistema de desarrollo Android Estudio Gradle. A modo de ejemplo, he usado la aplicación WiringPi para Android de HardKernel en http://bit.ly/1Eq3UpF. Comenzaremos con algunos conceptos básicos, tales como qué es NDK y lo más importante, ¿cómo debería utilizarse?

El NDK de Android (Kit de Desarrollo Nativo), es una forma de que las apps Android centrada en Java puedan interactuar con librerías C o C++. Esto puede sernos útil al permitirnos usar librerías existentes, así como mejorar el rendimiento ya que utilizamos C o C++ que se compilan desde el código fuente a una plataforma específica. Antes de que te emociones demasiado por el hecho de puedas mezclar C y C++ en tu próxima aplicación, debes de tener en cuenta algunas consideraciones. La documentación de Google que se incluyen en el NDK resumen mejor estas cuestiones:

"Antes de descargar el NDK, debes entender que el hecho de usar NDK no tiene un beneficio directo sobre la mayoría de las aplicaciones. Como desarrollador, necesitas sopesar sus ventajas frente a sus inconvenientes. Cabe destacar que usar el código nativo en Android generalmente no repercute en una mejora notable del rendimiento, pero sí que aumenta la complejidad de la aplicación. En general, sólo debes utilizar el NDK si es esencial para tu aplicación y nunca porque simplemente prefieras programar en C/C++".

En mayo de 2013, Google cambió de Android Development Tools (ADT), un plugin para Eclipse a Android Studio, que se basa en IntelliJ de NetBrains, ya que es su IDE por defecto para Android. Studio a diferencia de ADT, utiliza un sistema de desarrollo conocido como Gradle. Aunque no está disponible en el lanzamiento, se ha añadido soporte NDK preliminar para Gradle desde la versión 1.3 de Studio. A pesar de que se ha añadido soporte para el NDK, la integración en el IDE está aún muy limitada, su uso requiere de cambios manuales y utilizar versiones alfa de determinados módulos.

Instalación

Vamos a comenzar con lo más básico, conseguir todas las piezas que necesitamos. La instalación de Android Studio, el SDK y el NDK llevan un tiempo y requieren alrededor de 2 GB de archivos que necesitamos descargar.

Android Studio

Para empezar, debes tener la última versión de Android Estudio instalado. Si no es así, descarga una copia desde http://bit.ly/IKeIqs. La instalación es bastante sencilla para todas las plataformas. Google ofrece una buena documentación en el caso de que encuentres dificultades durante la instalación.

NDK y SDK

Lanza Android Studio y haz clic en el botón SDK Manager en el panel de iconos superior, como se muestra en la Figura 1. Tras cargarse, haz clic en el texto de la parte inferior "Launch Standalone SDK Manager", desde donde queremos instalar, como mínimo, los siguientes elementos:

```
Android 4.4.2 (API 19) → SDK
Platform
Extras → Ndk Bundle
```

Figura I – Botón Android SDK Manager

Se supone que planeas usar la imagen oficial de Android de HardKerned, limitada a Kitkat 4.4.2, pero existen diversas imágenes de la comunidad que se basan en versiones de Android más recientes. Mira las herramientas de desarrollo SDK de Android y toma nota de la versión más reciente que tienes instalada. Como se muestra en la Figura 1, yo sólo tengo la 23.0.1 instalada. Tras seleccionar todos los paquetes haz clic en "Install <X> packages..", siéntate y relájate mientras se descargan e instalan. Una vez finalizado el proceso, reinicia Android Studio.

Editar el proyecto

Como he dicho antes, estoy usando la app wiringPi de HardKernel que importé en Studio. De cualquier modo, es necesario el mismo método y modificaciones NDK para cualquier proyecto

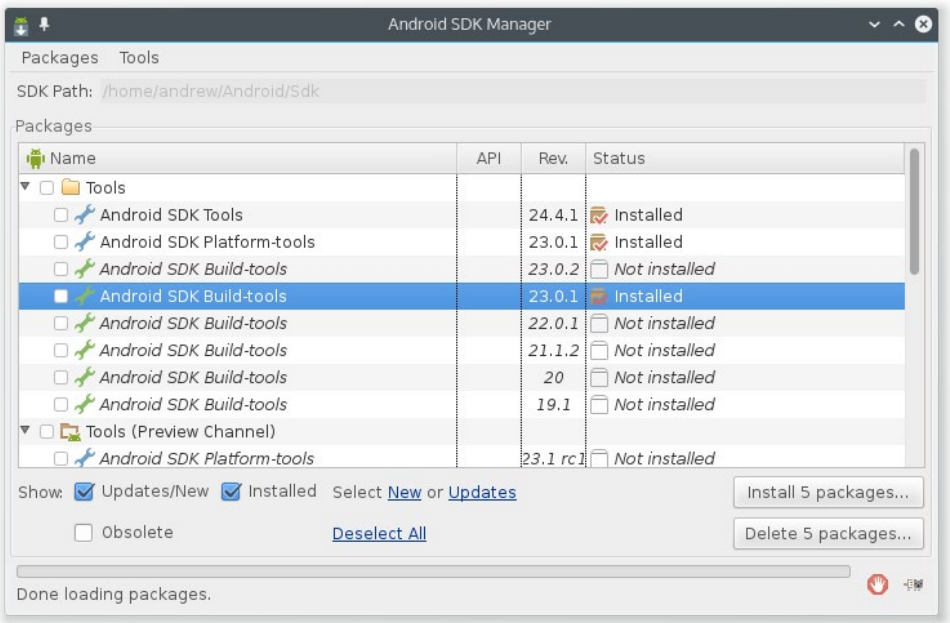


Figura 2 – Versión del Android SDK Manager

creado desde cero. Si decides trabajar con una parte de código existente, asegúrate de entender bien sus archivos android.mk o el propio código, para compilarlo correctamente. Existen tres archivos Gradle que necesitaremos modificar.

```
${project}/build.gradle
${project}/App/build.gradle
${project}/App/gradle/wrapper/
gradle-wrapper.properties
```

Tras realizar todos los cambios, puedes desarrollar, depurar y ejecutar la aplicación desde Android Studio como si se tratara de cualquier otro proyecto de Studio. Una guía rápida sobre el uso de adb sobre WiFi la puedes encontrar en http://bit.ly/1QrEOyE. Si está utilizando una copia de la aplicación wiringPi de Studio desde GitHub, asegúrate de que la ruta SDK y NDK muestra tu sistema. Las rutas NDK y SDK se encuentran en el archivo ${Project}\local.properties.

Los cambios necesarios a nivel de proyecto para build.gradle son bastante simples. Aquí queremos fijar las rutas de clases de los scripts de desarrollo dependientes de "com.android.tools.build:gradle-experimental:0.3.0-alpha5." Esto es necesario para utilizar la última versión experimental de Gradle que incluye soporte NDK, ya que las versiones estables actuales no lo hacen.

```
${project}/App/build.gradle

buildscript {
      repositories {
      jcenter()
      }
      dependencies {
      classpath 'com.android.tools.build:gradle-experimental:0.3.0-alpha5'
      }
}
allprojects {
      repositories {
      jcenter()
      }
}
```

Este archivo necesita de una gran cantidad de trabajo. He añadido comentarios antes de determinadas partes y líneas para explicarlas. Una copia sin comentarios y limpia del archivo la puedes encontrar en http://bit.ly/1QrEVdB.

```
// Take note that all attributes
are set with '=' rather than just
a space as is typical
// Fist off we are going to be
using a different plug than nor-
mal, note the addition
// of 'model' in the name.
apply plugin: 'com.android.model.
application'

// Certain attributes now need
to be placed inside of an extra
'model' namespace
model {
      android {

      // SDK to be compiled
against, you can use an SDK lower
than your device's
      // Android version but, a
greater one may/will cause prob-
lem
      // A mapping of OS to API
version can be found here:
      compileSdkVersion = 19
      // I recommend using the
latest version of build tools you
have
      // when working with the
experimental Gradel versions
      // the buildToolVersions
installed can be found:
      // en.wikipedia.org/wiki/
Android_version_history
      buildToolsVersion =
"23.0.1"

      // Inside of default config
(not appended with .with) are the
same values
      defaultConfig.with {
            applicationId =
"com.hardkernel.wiringpi"
            minSdkVersion.api-
Level = 16
            // I've seen a lot
of misconceptions about what tar-
getSdkVersion really is
            // target Sdk has
NOTHING to do with compilation of
the application
            // It tells An-
droid how to render and theme the
views, here it's set to 19
            // this means on
```

```
Android Lollipop (SDK 23) it will still render as
            // all the views as if they were on 19, unset this default to minSdkVersion
            targetSdkVersion.apiLevel = 19
      }
      }

      // It is advisable to keep both these version the same
      compileOptions.with {
      // Java source libraries that are used
      sourceCompatibility = JavaVersion.VERSION_1_7
      // Java compiler that is used to create bytecode
      targetCompatibility = JavaVersion.VERSION_1_7
      }

      // Here is where all the configuration for the NDK
      android.ndk {
      // moduleName is the name of the library being build by the NDK
      // the name must EXACTLY match the name of the NDK library
      // which is being loaded from java's "loadLibrary()" method
      moduleName = "wpi_android"
      // This is the NDK platform which will be used
      // In the Android.mk this was 'APP_VERSION'
      // if not set platform version is set the compileSdkVersion
      platformVersion = 19
      // For any compiler flags they are set with either CFlags or CPPFlags
      CFlags += "-DRMOLD"
      CFlags += "-UNDEBUG"
      CFlags += "-DANDROID".toString()
      CFlags += "-I${file("src/main/jni")}".toString()
      CFlags += "-I${file("src/main/jni/wiringPi")}".toString()
      CFlags += "-I${file("src/main/jni/devLib")}".toString()
      // external library that need to be referenced
      ldLibs += ["log", "dl"]
      }

      android.buildTypes {
      release {
            minifyEnabled = false
            proguardFiles += file('proguard-rules.txt')
      }
      }

      // This following part of code is set to build NDK for ALL platform types
      // which NDK supports.  For Odroid devices (currently) we only need the arm
      // and x86 (for debug on a host computer) ABIs.  For information about the
      // Support ABIs can be found at the link bellow.
      // https://developer.android.com/ndk/guides/abis.html
      android.productFlavors {

      create("arm") {
            // the abiFilters set the directory name where the lib will be located
            ndk.abiFilters += "armeabi"
      }
      create("arm7") {
            ndk.abiFilters += "armeabi-v7a"
      }
      create("arm8") {
            ndk.abiFilters += "arm64-v8a"
      }
      create("x86") {
            ndk.abiFilters += "x86"
      }
      create("x86-64") {
            ndk.abiFilters += "x86_64"
      }
      create("mips") {
            ndk.abiFilters += "mips"
      }
      create("mips-64") {
            ndk.abiFilters += "mips64"
      }
      // Set to build all ABI flavours
      create("all")
      }
}
```

El cambio final lo tenemos que hacer en los archivos de propiedades para especificar la versión Gradle. La parte final de la URL de distribución tiene que cambiarse "gradle-2.6-all.zip" si está utilizando una versión anterior.

```
${project}/App/gradle/wrapper/gradle-wrapper.properties

#Sat Nov 07 12:41:05 EST 2015
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-2.6-all.zip
```

Figura 3 - Error de Gradle

Otros cambios

Dependiendo de tu configuración, es posible que aparezca un error "Failed to sync Gradle project <nombredelProyecto>", motivado por una incongruencia entre la versión de Gradle del sistema y la versión del proyecto. Si hace clic en el enlace de la descripción del error, lanzará el editor de Gradle. En "Project-level Settings" ajusta la opción a "Use default gradle wrapper (Recommended)" Esto hace que el proyecto sea por defecto a la versión de Gradle que esta especificada en el archivo de propiedades.

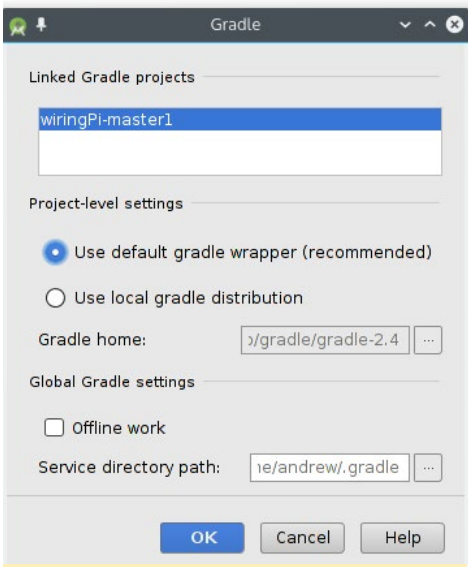


Figura 4 - Ajustes de Gradle

FRETS ON FIRE

LIBERA LA ESTRELLA DE ROCK QUE LLEVAS DENTRO

por @vOltumna

ODROID ya es compatible con los Sofware karaoke (http://bit.ly/1PLowzd) y dancing (http://bit.ly/1NAnHoc), de modo que el siguiente paso lógicamente es agregar algunas guitarras y tambores. Frets on Fire es un clon Python de Guitar Hero disponible en el repositorio de Debian. Vídeo en http://bit.ly/1MSjhZe.

Instalar Frets on Fire es fácil:

```
$ sudo apt-get install fofix
```

Frets on Fire necesita soporte OpenGL, por lo que necesitas GLshim para que el juego se ejecute utilizando OpenGLES:

```
$ cd ~/Downloads
$ mkdir glshim
$ cd glshim
$ wget http://oph.mdrjr.net/meveric/other/freeorion/libgl-odroid_20150922-1_armhf.deb
$ sudo apt-get install gdebi
$ sudo gdebi libgl*.deb
```

Después, vincula los drivers de Mali (en el XU3 y XU4, utiliza libmali.so en lugar de libMali.so):

```
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libEGL.so
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libGLESv1_CM.so
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libGLESv2.so
```

Luego, modifica el archivo /usr/games/fofix para que coincida con lo siguiente:

Figura I – Sistema de juego del Frents on Fire

```
#!/bin/sh -e
export LD_LIBRARY_PATH=/usr/lo-
cal/lib
cd /usr/share/fofix/src
exec ${PYTHON:-python} ${FOFIX_
PYTHON_FLAGS:--OO} FoFiX.py "$@"
```

Ahora podrás iniciar el juego:

```
$ fofix
```

Pero ¿dónde pones tu música o temas personalizados? Normalmente, los deberías poner en el directorio /usr/share/fofix/data, pero yo prefiero ponerlos en mi carpeta de inicio para acceder a ellos con facilidad. Para hacer esto, inicia el juego, luego salte para crear la carpeta ~/.fofix. Mueve los temas a tu carpeta de inicio y crea los enlaces simbólicos para que puedas poner tus canciones en ~/.fofix/songs y temas en ~/.fofix/themes:

```
$ sudo mv /usr/share/fofix/data/
themes /home/odroid/.fofix/
$ sudo chown -R odroid:odroid /
home/odroid/.fofix/themes
$ sudo rmdir /usr/share/fofix/
data/songs
$ mkdir /home/odroid/.fofix/songs
$ sudo chown odroid:odroid /home/
odroid/.fofix/songs
$ sudo ln -s /home/odroid/.fofix/
songs /usr/share/fofix/data/songs
$ sudo ln -s /home/odroid/.fofix/
themes /usr/share/fofix/data/
themes
```

Diviértete haciendo bailar a tu ODROID, y ¡probablemente a tus vecinos! Para comentarios, preguntas y sugerencias, visite el hilo original en http://bit.ly/1l1eyhu.

SCRIPTS MULTI-ARRANQUE PARA ODROID-XU4

HAZLO SENCILLO

por @loboris

A menudo es apropiado tener varios sistemas operativos instalados en la misma tarjeta microSD o módulo eMMC. He escrito algunos scripts para poder hacer esto en un ODROID-XU3 o ODROID-XU4. Esta guía detalla los pasos necesarios para utilizar mis scripts y así poder crear una instalación Multi-arranque

Introducción

- Permite crear un sistema múltiarranque en una única tarjeta SD o módulo eMMC con una combinación de Android, Linux y OpenELEC
- Menú de inicio para elegir el SO
- Probado con las imágenes oficiales de Hardkernel: Android 4.4, CM 12.1 Android 5.1.1, Ubuntu Mate de 15.04, Ubuntu Server 14.04 y OpenELEC 5.0.7.0
- Confirmado que funciona con un ODROID-XU4, aunque también soporta el ODROID-XU3
- Permite crear una instalación directamente desde update.zip de Android, una imagen de instalación de Linux y desde el archivo de actualización de OpenELEC
- La fuente de instalación para Android y Linux puede ser una copia de seguridad de una instalación de Android/Linux existente (se incluye los scripts de backup)
- Se pueden ajustar todos los tamaños de las particiones
- Se recomienda preparar e instalar la tarjeta eMMC usando Linux
- Como ventaja, no tienes que retirar el módulo eMMC del ODROID

Desarrollo

En primer lugar, descarga y descomprimir el paquete de scripts desde http://bit.ly/1MjfYgx. Ten en cuenta que todos los comandos se deben ejecutar en el directorio del script.

Puede que tengas que instalar algunos paquetes auxiliares que son reportados como ausentes por el script durante la instalación usando el comando apt-get.

Preparación

Se recomienda una tarjeta de 16 GB o mayor, como minio se ha de utilizar una de 8GB. Antes de ejecutar el script, puedes editar el archivo prepare_multicard para ajustar los tamaños de las particiones deseadas editando las variables que están al principio del scirpt.

Para empezar, inserta tu tarjeta en el lector USB del host con Linux y ejecuta los siguientes comandos en el directorio del script:

```
$ sudo ./prepare_multicard
<sd|mmc_card> <card_type>
```

<sd | mmc_card> es el dispositivo de la tarjeta (/dev/sdX, /dev/mmcblkX), y <card_type> es "sd" para la instalación de la tarjeta SD o "em" para la instalación de la tarjeta eMMC. Tras ejecutar

el script, tendrás una tarjeta SD o módulo eMMC lista para la instalación de Android, Linux y OpenELEC.

Instalar Android

Inserta tu tarjeta preparada para el múlti-arranque en un lector USB de tarjetas SD en el host Linux y ejecuta los siguientes comandos en el directorio del script:

```
$ sudo ./copy_android <source>
<dest> <card_type> [update]
```

<source> puede ser:

- Archivo de actualización de Android (update.zip)
- Directorio de la backup de Android (creado con el script backup_single_android o copy_android)
- Tarjeta sd/emmc (/dev/sdX, /dev/mmcblkX)) con una instalación multi-arranque válida

<dest> puede ser:

- Tarjeta SD o módulo eMMC (/dev/sdX, /dev/mmcblkX)) preparados para el multi-arranque
- Directorio (crea una backup de la instalación de Android múltiarranque)

<card_type> debe ser "sd" para la instalación de la tarjeta SD o "em" para la instalación del módulo eMMC

Si el 4th parámetro [update] está presente, el script no copia el contenido de la partición de datos. Esta opción está destinada para ser utilizada si queremos actualizar una instalación de Android multi-arranque existente. Entra en ella si sólo vas a actualizar la misma versión de Android.

Instalar Linux

Inserta tu tarjeta para el multiarranque en el lector y ejecuta los siguientes comandos en el directorio del script:

```
$ sudo ./copy_linux <source>
<dest> <card_type>
```

<source> puede ser:

- Imagen de instalación de Linux (linux_ver-xxx.img)
- Directorio de backup de Linux (creado con el script backup_single_linux o copy_linux)
- Tarjeta SD o módulo eMMC (/dev/sdX,/dev/mmcblkX)) con una instalación multi-arranque válida

<dest> puede ser:

- Tarjeta SD o módulo eMMC (/dev/sdX, /dev/mmcblkX)) preparados para el multi-arranque
- Directorio (crea una backup de la instalación de Linux multiarranque)

<card_type> debe ser "sd" para la instalación de la tarjeta SD o "em" para la instalación del módulo eMMC

Instalar OpenELEC

Inserta tu tarjeta (preparada para el multi-arranque) en el lector y ejecuta los siguientes comandos en el directorio del script:

```
$ sudo ./copy_oelec <source>
<dest> <card_type>
```

<source> puede ser:

- Archivo de actualización de OpenELEC (OpenELEC-ODROID-XU3-xxxxtar)

<dest> puede ser:

- Tarjeta SD o módulo eMMC (/dev/sdX, /dev/mmcblkX)) preparados para el multi-arranque

<card_type> debe ser "sd" para la instalación de la tarjeta SD o "em" para la instalación del módulo eMMC

SO por defecto y tiempo de espera

Tras retirar la tarjeta SD o módulo eMMC del host con Linux, insertarla en el ODROID y enciéndelo, aparecerá un menú de arranque en el que podrás seleccionar qué sistema operativo deseas cargar. Puedes editar el archivo "boot.ini.sel" en la partición FAT (userdata) para fijar el sistema operativo por defecto que se cargará si no se pulsa ninguna tecla (variable default_ OS). También puede ajustar el tiempo de espera (en segundos) tras el cual se iniciará el sistema operativo por defecto si no se pulsa ninguna tecla (variable BOOT_DELAY). Por último, se puede ajustar la resolución de la pantalla del menú de arranque (variable videoconfig), pero no todas las resoluciones funcionaran con tu monitor.

Copiar una backup

Puedes hacer una backup de tu tarjeta SD o modulo eMMC con Android/Linux y usarla para la instalación multi-arranque. Inserta la tarjeta SD o módulo eMMC que contiene el sistema operativo en el lector y ejecuta uno de los dos comandos siguientes en el directorio del script, dependiendo de si el sistema operativo fuente es Android o Linux:

```
$ sudo ./backup_single_android
<sd|mmc> <backup_dir>
```

o

```
$ sudo ./backup_single_linux
<sd|mmc> <backup_dir>
```

En este comando, <sd | mmc> es tu tarjeta SD o módulo eMMC con la antigua instalación de Android/Linux (/dev/sdX, /dev/mmcblkX), y <backup_dir> es el nombre del directorio de la backup. El script creará subdirectorios para las particiones de la tarjeta, así que puedes hacer copias de seguridad de la tarjeta con Android y Linux en el mismo directorio.

Para comentarios, sugerencias y preguntas, por favor visita el hilo original del foro en http://bit.ly/1j9r6TG.

JUEGOS LINUX

FALLOUT: UN JUEGO DE ROL POS-NUCLEAR

por Tobias Schaaf

Siempre he sido un fan de la serie Fallout. Cuando salió por primera vez, llegue a jugar durante horas y horas tratando de encontrar cada pequeño secreto. Por mucho que me gustara el juego, seguía muriendo una y otra vez, era demasiado joven e inexperto en 1997. Sin embargo, eso nunca me impidió continuar, y seguí jugando y jugando hasta que finalmente supere el juego. Hoy en día, tengo mucha más experiencia en este tipo de juegos, pero también menos pacienta. La serie Fallout, especialmente el primero, me traía muy buenos recuerdos, y es por ello que quería jugarlo de nuevo en mi ODROID y ver qué tal se ejecutaba.

Introducción

Como su nombre indica, el juego se desarrolla en un futuro en el que la mayor parte de la tierra ha sido destruida por ataques nucleares. Una guerra sin cuartel ha dejado la tierra devastada, aniquilando la mayor parte de la humanidad, los animales y la vegetación. Hay algunos sobrevivientes en búnker subterráneos llamados “Vaults”. Algunas otras criaturas menos afortunadas han sobrevivido a las explosiones nucleares, pero han sufrido mutaciones a causa de la radiación nuclear. Lo que siguió fue un ambiente muy hostil en el que las personas trata de sobrevivir. Algunos sólo quieren vivir pacíficamente y reconstruir la sociedad, y otros piensan que es más fácil robar y aterrorizar a los débiles.

En el juego, eres un habitante del Vault 13, que va creciendo con una relativa seguridad. Te encomiendas la tarea de salvar tu Vault, la instalación de depuración de agua se ha roto y dentro de 150 días tu refugio se quedará sin agua. Se supone que encontrando un chip de agua podrás reemplazar la parte rota en tu Vaults.

El juego tiene un estilo isométrico, y alterna el modo por turnos y el modo en tiempo real. Cada vez que entras en una pelea, se activa el sistema por turnos en el que tienes una cierta cantidad de Puntos de Acción (PA) para realizar diferentes tareas, como caminar, usar elementos o atacar.

El juego ofrece muchas y diferentes armaduras, armas y otros elementos para que los utilices o comerciales con ellos. Cuentas con diferentes habilidades, como primeros auxilios, abrir cerraduras o hacer cosas sin ser visto, que puedes entrenar para mejorar tu personaje. Fallout es muy divertido, a pesar de que se desarrolla en un ambiente muy siniestro, lo cual pone de relieve lo mala que es siempre una guerra.

Requisitos previos

El juego está disponible tanto para DOS como Windows. Puesto que tenemos una versión funcional de DOSBox en mi repositorio, un emulador de DOS, debería ser bastante fácil conseguir que Fallout se ejecute en DOSBox. Si sólo posees la versión para Windows del juego (o la versión de GoG.com) no te preocupes, hay una forma muy simple de ejecutar el juego también en ODROID.

Instalación

```
$ sudo apt-get install dosbox-
odroid
```

Configuración

Inicia DOSBox tras crear el archivo de configuración por defecto. Luego, salte de inmediato y abre /home/odroid/.dosbox/dosbox-SVN.conf en un editor de texto y cambia las siguientes líneas:

```
[sdl]
fullscreen=true
fullresolution=desktop
output=overlay
[dosbox]
memsize=31
[render]
 frameskip=3
aspect=true
[cpu]
core=dynamic
```

Antes de iniciar de DOSBox, creé una carpeta para poner mis juegos:

```
$ mkdir DOS
```

Después, copié la ISO de Fallout y la puse en una carpeta llamada "CDs" en mi ODROID. Para facilitar las cosas, agregue las siguientes líneas al final del archivo de configuración de DOSBox, de modo que no necesito escribirlas cada vez que quiera ejecutar el juego:

```
[autoexec]
mount c: /home/odroid/DOS -free-
size 1024
imgmount d: /home/odroid/CDs/
Fallout.iso -t iso
c:
```

Ahora que el juego está preparado, podemos iniciar el emulador. La carpeta DOS será montada automáticamente como unidad C:, y el CD será montado como una unidad de CD-ROM en D:. Probablemente habrás notado que he añadido la opción "-freesize 1024" para montar la unidad C: Esto es necesario para que Fallout se instale completamente ya que necesitae casi 600 MB. Sin esta opción, la unidad C: se monta con menos de 300 MB libres, aunque sea del mismo tamaño que la tarjeta SD o módulo eMMC. Esta opción a veces es necesaria para que los grandes juegos comprueban el espacio disponible en disco antes de instalarse.

Después de esto, cambié a la letra de unidad D: e inicié el instalador. Decidí hacer una instalación completa ya que esto nos permitirá ejecutar el juego sin el CD. Una vez instalado el juego, cambia a C: y entra en la carpeta donde instalaste el juego, después inicia la herramienta de configuración del sonido. Realiza una búsqueda automática de los ajustes de sonido. Con esto, habrás finalizado la instalación del juego y podrás iniciarlo introduciendo "fallout":

```
> c:
> cd fallout
> sound
> fallout
```

Instalar la versión de Windows

Para la versión de Windows de Fallout, primero tienes que instalar el juego en tu PC con Windows. Copia la carpeta de instalación a tu ODROID en la carpeta que creaste para DOSBox, como /home/odroid/DOS. Necesitas descargar el parche 1.1 para DOS de Fallout, así como algunos archivos básicos de DOS. Los puse en mi espacio web gestionado por @mdrjr, por lo que todos los archivos necesarios se pueden descargar desde http://bit.ly/1HZAHSt. El parche 1.1 de DOS para Fallout incluye un ejecutable de arranque DOS, que se puede utilizar para jugar a la versión Windows de Fallout bajo DOS.

El resto de los archivos son necesarios

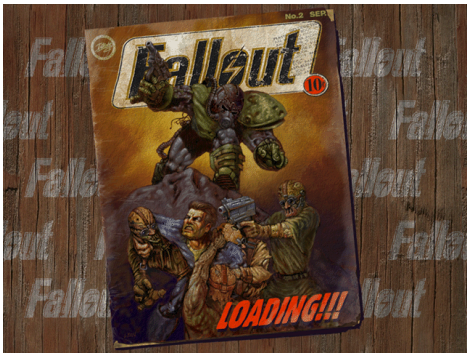
Figura 1 - Instalador alemán de la versión DOS de Fallout I

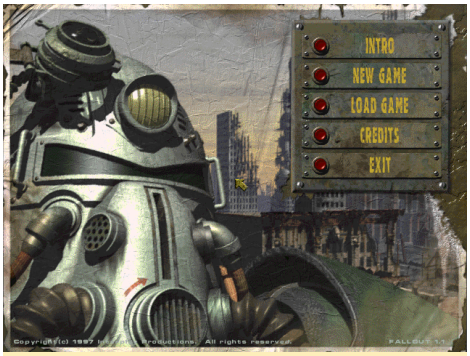
para lanzar fallout.exe. Tras copiar todos los archivos en tu ODROID en la carpeta Fallout desde tu versión de Windows, todo debería estar listo para jugar.

Introducción

El juego empieza con una introducción más bien siniestra, que explica los fundamentos y la configuración del juego, después te aparecerá el menú principal. Puedes comenzar un nuevo juego o cargar una de tus partidas guardadas. Iniciando un nuevo juego aparecerá una pantalla para seleccionar el personaje, donde puede elegir jugar a uno de los tres personajes predefinidos o crear tu propio personaje personal. Normalmente yo elijo el segundo.

La creación de personajes y más adelante subir de nivel, es una de las cosas más importantes que se hacen en el juego. Aquí elijes las habilidades básicas, rasgos y destrezas. Te llevará tiempo crear un personaje y seleccionar los atributos correctos. No puedes aprovechar todos sus atributos, así que tienes que elegir los que consideres más importantes, y centrar tu entrenamiento en algunas habilidades, más que en todas ellas.





Figuras 2 y 3 - Pantalla de Carga y menú principal de Fallout I. Por extraño que parezca, el menú principal no incluye Preferencias

Tus habilidades especiales definen los atributos básicos de tu personaje, como cuánto pueden levantar, cuánto daño pueden soportar, y cuántos puntos de movimiento tienes. Puede aumentar un atributo a costa de otro. Si quieres transportar una gran cantidad de artículos, necesitas mucha fuerza. Para regatear y conseguir mejores precios o persuadir a los demás, necesitan aumentar el carisma. La suerte te da una mayor posibilidad de realizar golpes críticos. Si aumentas uno de tus atributos, otro podría sufrir, así que elige bien los que consideres más importantes.

Las cualidades son como habilidades especiales que tiene tu personaje. Estas habilidades a menudo también tienen

Figura 4 - La pantalla de creación de personajes de Fallout I utiliza un sistema especial (Fuerza, Percepción, Resistencia, Carisma, Inteligencia, Agilidad, Suerte)

un precio, así que ten cuidado con lo que seleccionas. Por ejemplo, la habilidad de disparo rápido reduce el tiempo de uso de cualquier arma, pero también te impedirá que apuntes a las partes del cuerpo de tu enemigo, por jemplo a la cabeza. La habilidad de inteligencia, que aumenta tus atributos y habilidades iniciales, hará que el inicio del juego sea más fácil, pero también reduce la velocidad con la que aprendes cosas nuevas, haciéndolo más difícil al final. Un metabolismo rápido te sanará más rápido, pero también hará que el veneno se extienda más rápido por tu cuerpo. La resistencia química te evitará el consumo de drogas, pero también hará que las drogas que mejoran tus habilidades desaparezcan mucho más rápido, mientras que la dependencia química hará exactamente todo lo contrario. La selección de cualidades además de ser muy importantes, afectará al desarrollo de tu personaje.

Habilidades

Las habilidades definen lo bueno que eres con el uso de determinados tipos de armas o estilos de lucha, así como la posibilidad de conocer primeros auxilios y de abrir cerraduras. Puedes elegir tres "habilidades de etiqueta", que tienen mayores atributos de partida, así como un doble nivel de velocidad si les distribuyes los puntos cuando subes de nivel.

Hay muchas cosas a tener en cuenta cuando se crea un nuevo personaje, aunque también ofrece muchas oportunidades y le da al juego un valor añadido para poder poner en práctica un enfoque diferente cada vez que juegas.

Jugabilidad

Tras elegir a tu personaje, una segunda de video te dice que debes conseguir un nuevo chip de agua para tu Vault. También te dice que hay otro Vault cercano que debes visitar. Esta es toda la información de la que dispones. El comienzo del juego puede ser un poco difícil, ya que no existe un tutorial real a excepción de un NPC que te indica cómo jugar. Cuando el juego salió a la luz, se supone que incluía un manual donde encontrabas más información.

Empiezas en una cueva infestada por un puñado de ratas, que puedes utilizar para entrenar un poco y aprender a luchar. Aunque probablemente tengas un arma, es mejor que uses un cuchillo al principio, de modo que abre tu inventario, prepara a tu personaje y empieza a explorar el mundo.

Se juega en tercera persona con vista isométrica. Sigues a tu personaje, pero también puedes alejarte de él para explorar su entorno. Puedes caminar libremente hasta que te metes en un combate, que es cuando el juego cambia al modo por turnos.

Figura 5 - El juego comienza cuando abandonamos el Vault 13 nuestro hogar

Figura 6 - Luchar contra las ratas no es difícil, pero es un buen entrenamiento

Durante los combates, tienes que elegir las acciones que deseas llevar a cabo y planificar estratégicamente tus próximos movimientos, al mismo tiempo que tienes en cuenta los movimientos de tus enemigos. Al igual que tu personaje, tus enemigos tienen una cierta cantidad de PA, que les permiten moverse y atacar. A veces es mejor alejarse unos pasos para que el enemigo tenga que caminar hacia ti y por lo tanto tener que utilizar la mayor parte de su PA, lo que significa que no puede atacar. En tu siguiente turno, tienes tu PA al completo y puede golpear primero.

Si nos fijamos en la imagen de nuestro personaje luchando contra una rata en la Figura 5, verás que muestra la probabilidad de golpear al enemigo sobre el enemigo cuando intentas atacarlo. De este modo, te haces una idea de la probabilidad de herir o matar a un enemigo. Cuanto mejor sea tu nivel de habilidad de una determinada arma, mayor probabilidad tendrás de golpear al enemigo.

El juego es similar al resto de juegos de rol. Caminas por la zona, hablar con la gente, aceptas misiones, luchas, subes de nivel y mejoras tus habilidades. Puedes mejorar tu personaje y equipamiento, comprar o robar elementos. El dinero es siempre útil, pero nunca tendrás suficiente (como en la vida real). Lo que hace único al juego es su entorno siniestro, los personales a los que te enfrentas y las cosas que encuentras.

Fallout tiene ciclos de día y noche y un mapa inmenso para explorar, con eventos aleatorios y encuentros durante el viaje por el terreno. Eres virtualmente libre de hacer lo que quieras en este juego. ¿Te resulta molesto resolver misiones para algunos aldeanos? ¿Están empezando a enfadarte porque no obtienes ninguna recompensa por tu duro trabajo? Si desea simplemente acabar con todo un pueblo y coger lo que te deben, lo puede hacer en Fallout. Si un

NPC te está molestando, sigue adelante y dispárale. Si tus amigos intentan conspirar contra ti, también puedes hacerlos estallar. ¿Se ha quedado un aldeano atrapado en el fuego cruzado y ahora te está atacando? ¡Simplemente dispara! Los accidentes ocurren, y algunas veces tienen como resultado la aniquilación de toda un pueblo en Fallout. Puedes elegir si deseas liberar a algunos esclavos de sus dueños, o tomar alguna gente cautiva y venderlos como esclavos. Eres libre de hacer lo que quieras.

Las cosas que puede hacer en este juego son inmensas, como abrir cerraduras para obtener todos los objetos que la gente ha escondido. Puede intentar timar a algunos comerciantes, curando sus piernas rotas, o contrarrestando el veneno de su cuerpo. También puedes utilizar tus habilidades para arreglar las armas y máquinas que te encuentran en el camino, o usar la ciencia para piratear ordenadores. Hay muchas cosas ocultas por descubrir, como una nave extraterrestre. También puede reunir compañeros para que luchen junto a ti. Usando habilidades aumentará tu experiencia y te ayudará a subir de nivel más rápido.

Figura 7 - Viajando por el terreno con mapa

Figura 8 - Tras jugar un poco ya dispongo de algo de material muy bueno

Figura 9 - Exterminando un campamento

Consejos Prácticos

El juego puede ser muy difícil al principio, ya que la mayoría de los enemigos, a excepción de las ratas, pueden matarte muy rápido. Por ejemplo, los Radscorpions son escorpiones gigantes muy peligrosos que a menudo aparecen en grupos de cuatro o más cuando te mueves por el terreno. Debes guardar muy a menudo, como en la mayoría de los juegos.

Fallout está motivado por el azar, si intentas robar la cartera a alguien, puede que falles, pero si vuelve a cargar el juego y lo intentas de nuevo, es posible que lo consigas. Incluso si la habilidad para robar es muy baja, entre guardar, recargar y con un poco de paciencia, es posible que llegues a ser un buen ladrón.

Robar es una buena manera de conseguir objetos, y la puedes utilizar para quitarle las armas a un enemigo. Si robas todas las armas a un grupo antes de entrar en combate, es probable que sólo ataquen con puños y cuchillos, lo cual te da una gran ventaja.

Usa tus habilidades tanto como te sea posible. Curar y abrir cerraduras son buenas habilidades que merece la pena tener. No puedes llevar a todas partes demasiados artículos sin sobrecargarte, así que intenta llevar tus pertenencias a un lugar del que te acuerdes, como un armario o algo similar. Los artículos como armas y armaduras pueden ser muy valiosos con los que negociar.

Me encanta Fallout

Disfruto de Fallout sobre todo por su entorno. Muestra las terribles cosas que podrían suceder si la humanidad participara en una guerra destructiva. El juego presenta un mundo tenebroso con la mayoría de los seres humanos muertos, y el resto están luchando entre ellos. Los lugares que se pueden explorar junto con los elementos que puedes encontrar, hacen que este juego sea único.

Aunque Fallout 3 y 4 ofrecen mejores gráficos, el ambiente tan oscuro de Fallout 1 y 2 es mucho mejor en mi opinión. Es un clásico juego de rol, más que un RPG de acción en primera persona como Fallout 3 y 4. Fallout 1 es bastante difícil, pero si guardas a menudo y planear tus movimientos es muy divertido explorar un mundo tan grande e interesante. ¡Simplemente evita que te maten!

Figura 10 - Game Over: si ves esto en pantalla, ¡sabrás que realmente las has jodido!

LEER LA TEMPERATURA Y LA HUMEDAD DESDE UN SENSOR SHT15

UNA INTRODUCCION A LA INTERFAZ GPIO

por Jon Petty

El objetivo de este proyecto es utilizar un ODROID para leer los datos de temperatura y humedad de un sensor SHT15, así como explicar cómo se comunica un ODROID con un SHT15 con los pines GPIO. Los sensores SHT15 están fabricados por Sensirion y miden tanto la temperatura como la humedad de un entorno. La comunicación con el sensor se lleva a cabo a través de los pines GPIO de un ODROID. Un pin GPIO conecta al pin SCK del sensor, que controla la rapidez con la que se produce la comunicación. El segundo pin GPIO se conecta al pin DATA del sensor, que se utiliza para enviar comandos y leer los resultados. Una vez que todo esté configurado, el ODROID enviará una solicitud para medir la temperatura o la humedad a través del pin DATA, espera a que el sensor complete la medición y luego, lee el resultado sobre el pin DATA.

Conectar el sendor SHT15

El siguiente diagrama describe cómo conectar un sensor SHT15 a un ODROID.

Hay dos cuestiones a tener en cuenta. En primer lugar, las fichas técnicas son un buen lugar para obtener información sobre cómo usar los componentes electrónicos. El circuito de la figura 1 ha sido copiado de la ficha técnica del sensor. Es la configuración recomendada por el fabricante que permiten buenas mediciones. En segundo lugar, es algo complicado soldar un SHT15. Para facilitar las cosas, este tutorial utiliza una placa prefabricada con el sensor SHT15.

Suministros necesarios

Para empezar, necesitas los siguientes componentes y herramientas:

- ODROID (http://bit.ly/1QPVZa9)
- Kit de componentes ODROID (http://bit.ly/1LmFcdf)
- Placa de sensores SHT15 (http://bit.ly/1qd22ZL)
- Cables
- Soldador y Estaño

Una vez que tengas la placa del sensor SHT15, realiza las siguientes conexiones tras soldar los cables a la misma:

- Conecta VCC a la fuente de alimentación de +3.3V del ODROID
- Conecta DATA al pin #100 GPIO del ODROID
- Conecta SCK al pin #97 GPIO del ODROID
- Conecta GND al GND del ODROID

Deberías terminar con algo que se parezca a la figura 2.

Figura I – Imagen del diagrama del SHT15

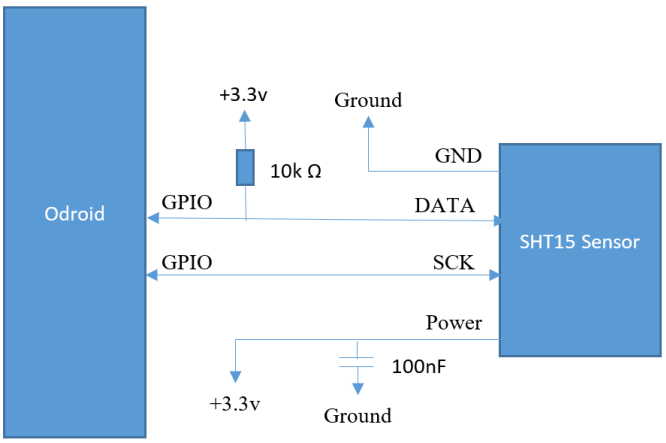
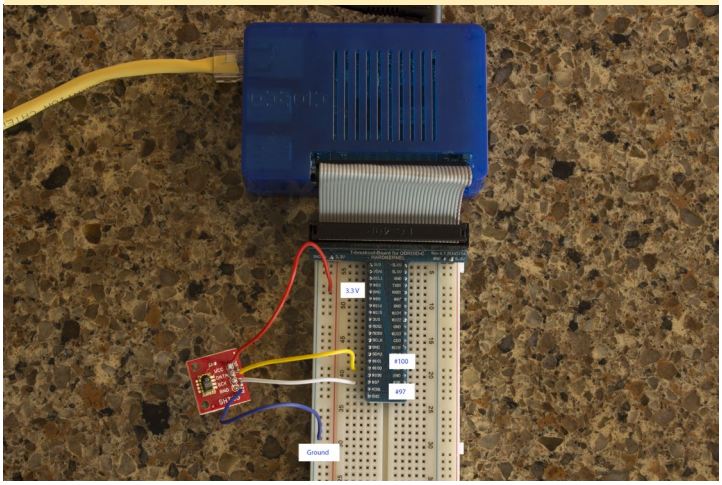


Figura 2 –Conexiones del SHT15



Lectura y grabación de los valores GPIO

GPIO significa pin Entrada/Salida de propósito general. De cuántos de ellos habrá dependido tu ODROID durante los prototipos, aunque en todos los casos, fueron usados para leer y escribir datos binarios. Los datos binarios son datos con sólo dos estados, comúnmente conocidos como HIGH y LOW, o 1 y 0. Físicamente, un valor HIGH significa que el voltaje de pin es de +3,3 voltios, y un valor LOW significa que el voltaje del pin es de +0.0 voltios. Ten en cuenta que el nivel de voltaje depende del dispositivo. Por ejemplo, un Arduino funciona desde los +5,0 voltios a los +0.0 voltios. Si el ODROID escribe datos en un pin GPIO, este cambiará el voltaje entre los +3,3 y +0,0 voltios dependiendo de si se ha escrito HIGH o LOW. Si el ODROID está leyendo datos, éste tomará HIGH cuando los +3,3 voltios sean aplicados al pin, y LOW cuando los +0.0 voltios sean aplicados al pin.

Para este proyecto, vamos a leer y escribir datos hacia y desde dos pines GPIO. Conlleva los siguientes pasos:

- Conecta los pines GPIO del ODROID al sensor
- Inicia sesión con Linux en el ODROID y navega hasta el directorio GPIO
- Inicia una conexión con los dos pines GPIO conectados (uno para DATA y otro para SCK)
- Cuando sea necesario, ajusta los pines en modo escritura y escribe los datos
- Cuando sea necesario, ajusta los pines en modo lectura y lee los datos

Para empezar, inicia sesión en tu ODROID y abre un terminal de línea de comandos. Algunos de los siguientes comandos deben ejecutarse como root, lo cual se puede hacer con el siguiente comando:

```
$ sudo su -
```

Los pines GPIO se encuentran en el directorio /sys/class/gpio

```
$ cd /sys/class/gpio
```

En el directorio hay un programa denominado “export”, que activa las conexiones con los pines GPIO. Se debe activar el pin antes de que éste pueda leer o escribir datos. Para iniciar una conexión, usa el número de identificación del pin.

En este tutorial, conectaremos el pin DATA del sensor SHT15 al pin GPIO 100, y el pin SCK del sensor al pin GPIO 97. Estas dos conexiones se inician con los siguientes comandos.

```
$ echo 100 > /sys/class/gpio/ex-
port
$ echo 97 > /sys/class/gpio/ex-
port
```

Una vez completados estos comandos, deberías encontrar los siguientes directorios creados:

```
/sys/class/gpio/gpio100
/sys/class/gpio/gpio97
```

Estos directorios contienen todo lo necesario para leer y escribir datos desde sus correspondientes pines GPIO. El primer archivo importante es “direction”. Para el pin GPIO 100, se encuentra en el archivo /sys/class/GPIO/gpio100/direction. El archivo “direction” cambia un pin entre el modo lectura y el modo escritura. No se puede leer y escribir datos de forma simultánea al mismo tiempo en un único pin. Puedes, sin embargo, tener múltiples pines donde algunos leen y otros escriben datos.

Un pin se puede cambiar a modo escritura escribiendo el valor “out” en el archivo “direction”. Del mismo modo, un pin se puede cambiar a modo lectura, escribiendo el valor “in” en el fichero “direction”.Ej. el siguiente comando cambia el pin GPIO 100 a modo escritura:

```
$ echo out > \
/sys/class/gpio/gpio100/direction
```

El siguiente comando cambia el pin GPIO 100 a modo lectura:

```
$ echo in > \
/sys/class/gpio/gpio100/direction
```

Para ver el modo en el que está un pin GPIO, puedes leer el valor “direction”. Por ejemplo, el siguiente comando determina si el pin GPIO 100 está en modo escritura o en modo lectura.

```
$ cat \
/sys/class/gpio/gpio100/direction
```

El segundo archivo importante a tomar en cuenta es “value”. Para el pin GPIO 100, se encuentra en /sys/class/GPIO/gpio100/value. La lectura y escritura de datos binarios se realiza utilizando el archivo “value”. Si el pin está en modo escritura, el archivo “value” se utiliza para enviar los datos binarios. Si el pin está en modo lectura, se utiliza el mismo archivo, pero en este caso lee los datos binarios desde el pin. Para demostrar esto, podemos realizar una pequeña prueba para ver si la placa de circuitos está conectada correctamente. Cuando nos conectamos al principio, el pin DATA debería estar en HIGH y el pin SCK debería estar en LOW. Para determinar sí es el caso, primero cambiar ambos pines al modo lectura.

```
$ echo in > \
/sys/class/gpio/gpio100/direction
$ echo in > \
/sys/class/gpio/gpio97/direction
```

Luego, lee el valor GPIO para cada pin.

```
$ cat \
/sys/class/gpio/gpio100/value
$ cat \
/sys/class/gpio/gpio97/value
```

El Pin 100 (DATA) debería reflejar un valor “1”, y el pin 97 (SCK) debería reflejar un valor “0”. Si ese no es el caso

las posibles soluciones al problema pasan por comprobar las conexiones de los cables utilizando como referencia el diagrama del cableado anterior, y verificar que los pines GPIO están en modo lectura comprobando los valores del archivo "direction" :

```
$ cat /sys/class/gpio/gpio100/direction
$ cat /sys/class/gpio/gpio97/direction
```

Comunicación con el SHT15

A un nivel alto, los siguientes pasos dan como resultado datos de humedad o temperatura que son leídos desde un sensor:

1. El ODROID envía una solicitud al sensor para registrar la temperatura o la humedad. Ten en cuenta que el sensor no puede leer la temperatura y la humedad al mismo tiempo. Si tienes que tomar ambas mediciones, éstas deben realizarse una tras otra.
2. El sensor empieza a tomar una medida, y ODROID espera.
3. Una vez completada la medición, ODROID procede a leer el resultado dese el sensor.
4. El ODROID convierte la medición en un formato legible.

Para solicitar que se haga una medición, ODROID envía un número binario al sensor. Por ejemplo, con el número 00000011 se solicita que se mida la temperatura y con el número 00000101 se pide que se mida la humedad. Los números se envían con un bit cada vez sobre al pin DATA. El pin SCK controla la rapidez con la que se envían los valores. Echa un vistazo a la figura 3, muestra los valores de pin GPIO al transmitir el número 00000101 (solicitud para hacer una medición de la humedad).

En la figura 3 podemos diferenciar tres partes importantes. La primera es la secuencia de inicio de la transmisión. Se trata de una combinación de valores HIGH y LOW transmitidos a

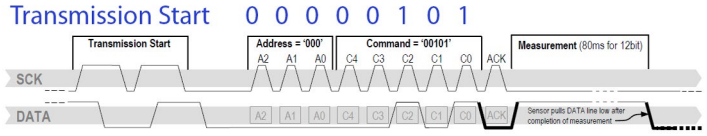


Figura 3 - Petición para medir la Humedad

través de DATA y SCK que avisan al sensor de que un comando está a punto de ser enviado. La segunda parte es el número de solicitud. En ella, el pin DATA transmite cada bit 0-0-0-0-0-1-0-1 y el pin SCK cambia entre 1 y 0. El pin SCK controla el tiempo empleado para transmitir los datos. Cuando SCK es 0, indica que no está listo para leer. Cuando SCK es 1, esto indica que está listo para leer. El hecho de que SCK alterne entre 1 y 0 mientras se transmite cada bit a través de DATA permite al ODROID enviar solicitudes de medición al sensor.

La última parte del diagrama es la sección ACK, también conocida como la sección de aceptación. En esta sección, ODROID cambia el pin DATA a modo lectura. Esto lo hace para poder leer los valores escritos por el sensor. Si el sensor SHT15 recibido correctamente el comando, escribirá un valor 0 a DATA durante la sección ACK, luego, cambiará DATA a 1. El ODROID continúa controlando el valor de SCK en modo escritura, y necesita un momento para que el sensor registre una medición.

Cuando la medición se haya completado, el sensor cambiará el pin DATA a 1. Esto indica que el ODROID está libre para leer el resultado del sensor. Los resultados constan de dos bytes, un total de 16 bits. La Figura 4 muestra el ODROID leyendo el resultado de una medición.

Como vemos en la Figura 4, ODROID lee el número en dos partes, 00000100 y 00110001. Cada una de ellas represen-

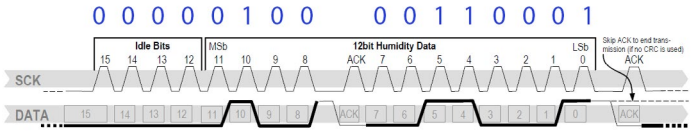


Figura 4 - Lectura de una Medición

ta un byte. Esto ocurre en tres tramos. El primero y el tercero transmiten los bytes reales. Estas transmisiones se producen bit a bit ya que el ODROID alterna SCK entre 0 y 1, mientras que lee DATA. El segundo tramo es otra señal ACK. Tras enviar el primer byte, el sensor cambia DATA a 1. Para enviar una señal ACK, el ODROID necesita cambiar DATA a 0 y alternar SCK entre 0 y 1. Esto indica al sensor que ODROID está listo para leer el segundo byte. La lectura del número desde el sensor está en binario y se debe convertir a un sistema numérico de base 10. Más adelante, utilizaremos el software para hacer esto. Pero por ahora, ten en cuenta que 0000010000110001 es igual a 1073.

Una vez que la medición haya sido grabada y convertida a un sistema numérico de base 10, debe añadirse a una ecuación para obtener el resultado final. Si la medición tomada es de temperatura, se utiliza la siguiente ecuación:

$$T = -39.7 + 0.04x$$

En estas ecuaciones, x es el número de base 10 registrado por el sensor SHT15 y T es el resultado final. Por ejemplo, un valor de 1617,5 tomado desde el sensor tras una medición de temperatura indica una temperatura de 25 ºC. Si la medición tomada es de humedad, se utiliza la siguiente ecuación.

$$H = -2.0468 + 0.0367x - 0.0000015955x2$$

En esta ecuación, x es el número de base 10 registrado por el

sensor SHT15 y H es el resultado final. Por ejemplo, un valor de 1.073 tomado desde el sensor tras una medición de humedad indica una humedad del 35,5%.

Usar PHP para leer los datos de humedad y temperatura

Tas echar un ojeada a la sección anterior, la idea de controlar los pines SCK y DATA a través de la línea de comandos de Linux para solicitar y leer las mediciones puede sonar poco atractivo. Si ese es el caso, ¡Estoy totalmente de acuerdo contigo! Para que esto sea más manejable, escribí dos scripts PHP que se encargan del trabajo pesado. Para descargar estos scripts, navega a un directorio en el que deseas guardarlos y ejecuta los siguientes comandos:

```
$ sudo apt-get install git php5
$ git clone git@github.com:\
jon-petty/shtx_php_example.git
```

El primer comando instala PHP, que es necesario para ejecutar los scripts. El comando también instala un programa llamado git, que se puede utilizar para descargar repositorios de código. El segundo comando usa git para descargar de hecho, los scripts. Si desea analizar los scripts antes de descargarlos, los pueden ver en http://bit.ly/1OGGK5Q.

Para ejecutar estos scripts, primero cambia los directorios y luego, sigue las instrucciones del archivo README. md. Contiene las instrucciones actualizadas sobre cómo ejecutar los scripts:

```
$ cd shtx_php_example
$ less README.md
```

Futuros proyectos

Por ahora, has conectado un sensor SHT15 a tu ODROID y eres capaz de registrar la humedad y la temperatura. También has llegado a entender cómo se controlan los pines GPIO en Linux, y qué protocolo de comunicación se utiliza con un sensor SHT15. Si quieres saber más, te animo a que eches un vistazo a los scripts PHP e intentar casar el protocolo de comunicación y las ecuaciones. También puede buscar las fichas técnicas y aprender cuestiones que están fuera del alcance de este artículo. Por ejemplo, si las temperaturas varían demasido de los 25 ºC, la humedad registrada necesita de una ecuación de compensación para que los resultados sean más precisos.

Referencias

Especifiaciones técnicas SHT1x. Sensirion, Dec. 2011. http://bit.ly/1x0FfqK

Un sensor de temperatura y humedad SHT15 puede comunicarse con software Python a través de pines GPIO del ODROID y así tus mascotas tengan mejor aspecto posible.

FIVE NIGHTS AT FREDDY'S

REPULLOS Y JUGUETES ATERRADORES

por Rob Roy

Tener miedo es muy divertido, y Five Nights at Freddy's es un gran juego para jugar por la noche a solas con las luces apagadas.

Has sido contratado como guarda de seguridad nocturno en casa de Freddy Fazbear. Tu misión es sobrevivir cinco noches a los ataques de animales animatronicos que deambulan por los pasillos durante la noche. Sólo puede verlos a través las cámaras de seguridad montadas en el edificio. ¡Considérate afortunado si lo haces durante la última noche! Actualmente hay cinco entregas del juego llenas de retos, que pueden ser descargadas desde Google Play Store en http://bit.ly/1XSlx8O.

JUEGOS ZELDA HECHOS POR LOS FANS

TU MUNDO FAVORITO DE FANTASIA SE AMPLIA

por Oliver Schmitt

SOLARUS
AN ARPG GAME ENGINE

Te acuerdas de los días en los que jugabas al Zelda: A Link to the Past y no podías parar hasta llegar al final. Pues bien, el pasado continua vivo con algunos juegos realmente buenos, hechos por los fans de Zelda. Esto significa que contamos con nuevos mundos y mazmorras que están esperando ser explorados y perversos enemigos que necesitan ser derrotados, de modo que es momento de volver a coger tu vieja espada 2D. Todos están en código abierto y están hechos por y para los fans y no están ligados a Nintendo de modo alguno, aunque usan la historia y algunos recursos de los juegos originales de Zelda.

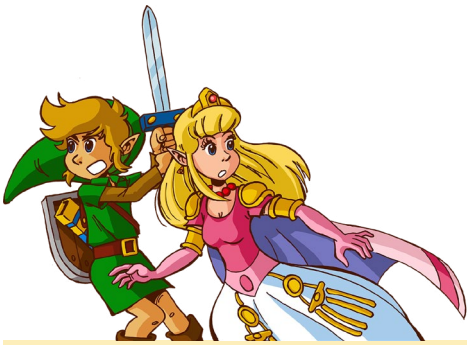


Figura I - Link y Zelda

En ODROID no sólo funcionan el motor y los juegos, sino que también es posible ejecutar el editor Solarus para poder crear nuevos mundos y aventuras. Si te gusta crear historias y dejar al resto que resuelvan tus enigmas, ésta podría ser una buena oportunidad para ti.

Empecemos

Necesitas al menos la versión 2.8.11 de cmake, disponible en la mayoría de los sistemas. Simplemente instala la versión cmake que viene junto con las distribuciones, excepto para Debian Wheezy, que requiere una versión más reciente. La forma más sencilla de hacerlo es instalar las versiones del repositorio backport:

```
$ sudo apt-get install -t wheezy-backports cmake
```

Tras asegurarte que dispones de la versión adecuada de cmake, clona el motor Solarus:

```
$ git clone git://github.com/christopho/solarus
$ cd solarus
```

También necesitarás algunos paquetes dev:

```
$ sudo apt-get install libsdl2-dev \
  libsdl2-image-dev libsdl2-ttf-dev \
  libphysfs-dev libluajit-5.1-dev
```

Solarus utiliza algunas funciones C++ que sólo se aplican en la versión 4.8, de modo que la versión 4.7.2 en Wheezy no las admite. Comprueba tu versión gcc con el comando “gcc --version”. Si tiene la versión 4.8 o posterior, ignora este paso y continúa con la compilación. Si utiliza una versión de gcc más antigua, tienes que aplicar el siguiente parche, que se deshace de estas nuevas funciones sin afectar al sistema de juego. Para asegurarnos que el parche hace su trabajo, llevaremos a cabo una comprobación. Para versiones más recientes, puede que tenga que modificar el código fuente de un modo similar a como lo hace el parche.

```
$ git checkout 4a662991f-967251101a32bd58dced44f0f3cf300
$ wget -O patch.txt http://paste-bin.com/raw.php?i=p5qLknQd
$ patch -p0 < patch.txt
```

Después, puedes compilar e instalar el motor:

```
$ mkdir build
$ cd build
$ cmake ..
$ make -j5
$ sudo make install
```

En este momento, existen tres juegos completos que utilizan el motor Solarus. Tienes que crear gamedata e iniciar Solarus con esos datos para poder jugar.

Antes de instalar los datos del juego, instala glshim para que los juegos se ejecuten utilizando OpenGLES:

```
$ cd ~/Downloads
$ mkdir glshim
$ cd glshim
$ wget http://oph.mdrjr.net/meveric/other/freeorion/libgl-odroid_20150922-1_armhf.deb
$ sudo apt-get install gdebi
$ sudo gdebi libgl*.deb
```

A continuación, vincula los drivers Mali (en el XU3 y XU4, utiliza libmali.so en lugar de libMali.so):

```
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libEGL.so
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libGLESv1_CM.so
$ ln -sf /usr/lib/arm-linux-gnueabihf/mali-egl/libMali.so /usr/lib/arm-linux-gnueabihf/libGLESv2.so
```

Zelda: Mystery of Solarus DX

Figura 2 – Logo del The Legend of Zelda Mystery of Solarus

Figura 3 – Captura de pantalla del The Legend of Zelda Mystery of Solarus

El motor de este juego fue diseñado por completo desde el principio. Se trata de la continuación de A Link to the Past. Tienes que proteger la Trifuerza, salvar algunos niños y al final, restaurar la paz en Hyrule. En primer lugar, clona el juego y compila el paquete:

```
$ git clone git://github.com/christopho/zsdx
$ cd zsdx/build
$ cmake ..
$ make -j5
```

Modifica el archivo zsdx para que se parezca a esto:

```
#!/bin/sh
export LD_LIBRARY_PATH="/usr/local/lib/arm-linux-gnueabihf/;/usr/local/lib/"
solarus_run /usr/local/share/solarus/zsdx
```

Por último, instala e inicia el juego:

```
$ sudo make install
$ zsdx
```

Una vez que inicies el juego, la configuración se guardará en la carpeta ~/.solarus/zsdx. La configuración general se almacena en settings.dat. Los parámetros del juego guardados tienen sus propios archivos, como el mapeo de tu joypad o teclado. Para usar un mando de la Xbox 360, ejecuta los siguientes comandos antes de iniciar el juego:

```
$ sudo apt-get install xboxdrv
$ sudo xboxdrv --dpad-only --silent &
```

Zelda: Mystery of Solarus XD

Mystery of Solarus XD empezó como una broma de 1º abril, pero aún así vale la pena echarle un vistazo, con dos enormes mazmorras y varias horas de juego. Es muy divertido, ya que no sólo hay mazmorras sino también un montón

Figura 4 – Logo del The Legend of Zelda Mystery of Solarus XD

Figura 5 – Captura de pantalla con Yoda del The Legend of Zelda Mystery of Solarus XD

de chistes, e incluso Yoda aparece como invitado. Si alguna vez te has preguntado cómo es la burocracia en Hyrule, este juego te dará la respuesta.

Primero clónalo y compílalo:

```
$ git clone git://github.com/christopho/zsxd
$ cd zsxd/build
$ cmake ..
$ make -j5
```

Modifica el archivo zsxd para que coincida con el siguiente contenido:

```
#!/bin/sh
export LD_LIBRARY_PATH="/usr/local/lib/arm-linux-gnueabihf/;/usr/local/lib/"
solarus_run /usr/local/share/solarus/zsxd
```

Por último, instala y lanza el juego:

```
$ sudo make install
$ zsxd
```

Zelda: Return of the Hylian SE

Figura 6 – Logo del The Legend of Zelda Return of the Hylian Solarus Edition

Figura 7 – Captura de pantalla de The Legend of Zelda Return of the Hylian Solarus Edition

Una nueva versión del juego de Vincent Jouillat también fue desarrollada con este motor. En primer lugar, clona el código fuente y compílalo:

```
$ git clone --branch solarus-1.5
\
  git://github.com/christopho/
zelda_roth_se
$ cd zelda_roth_se
$ mkdir build
$ cd build
$ cmake ..
$ make -j5
```

Despues, modifica el archivo zelda_roth_se para que coincida con el siguiente contenido:

```
#!/bin/sh
export LD_LIBRARY_PATH="/usr/lo-
cal/lib/arm-linux-gnueabihf/;/
usr/local/lib/"
solarus_run /usr/local/share/so-
larus/zelda_roth_se
```

Finalmente, instala el juego:

```
$ sudo make install
$ zelda_roth_se
```

Crear el tuyo propio

Como ya he mencionado, también puedes crear tus propios juegos con el editor para el motor Solarus. Se ejecuta directamente sobre ODROID. Una vez que lo hayas compilado e instalado con las instrucciones que se detallan a continuación, un buen punto de partida para empezar a desarrollar es http://bit.ly/1OFrdTJ, donde puedes encontrar

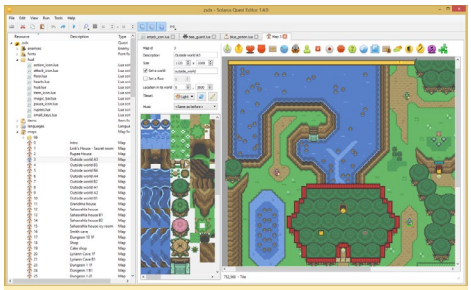


Figura 8 - Editor Solarus

varios paquetes de recursos y algunos tutoriales en vídeo.

Para compilar el editor Solarus, primero clona el repositorio y prepara tu sistema. Necesitarás los siguientes paquetes de desarrollo:

```
$ sudo apt-get install qtbase5-
dev qttools5-dev qttools5-dev-
tools
```

En Debian Wheezy, éstos se deben instalar desde el repositorio backports:

```
$ sudo apt-get install -t wheezy-
backports qtbase5-dev qttools5-
dev qttools5-dev-tools
```

Después de eso, clona y compila:

```
$ git clone git://github.com/
christopho/solarus-quest-editor
$ cd solarus-quest-editor
$ mkdir build
$ cd build
$ cmake ..
$ make -j5
$ sudo make install
```

El editor se puede iniciar con el siguiente comando:

```
$ LD_LIBRARY_PATH="/usr/local/
lib/arm-linux-gnueabihf/;/usr/lo-
cal/lib/" solarus-quest-editor
```

¡Buena suerte con tu creación de juegos de aventuras!

Juegos no Solarus

Además del motor Solarus, también hay otros juegos de fans que funcionan en los ODROIDs. Ya conocemos el primero en la lista, lo vamos a describir en detalle ya que el resto pueden ser

Figura 9 – Portada del juego Return of the Hylian

Figura 10 – Captura de pantalla del sistema de juego del Return of the Hylian

compilados del mismo modo. Uno de estos juegos es Zelda: Return of the Hylian en su versión original.

La historia está resumida en la página de inicio: "Tras la victoria de Link sobre Ganon, nadie sabía que el deseo de Link era la Trifuerza. Pero este deseo reunificó el mundo de la luz y el mundo de la oscuridad, y trajo consigo que los 7 des-

cendientes de los hombres sabios volviesen a la vida. La Paz volvió a Hyrule. Lamentable, este deseo también resucitó a Ganon y a sus secuaces. Él preparaba su venganza, pero no podía hacer nada sin la Trifuerza. Una noche, una voz familiar habló a Link en sueños... “

El juego está disponible en Inglés, alemán, español y francés, el idioma se puede seleccionar haciendo clic sobre las banderas de los países en http://bit.ly/1PDC2G2. Escribe los siguientes comandos para instalar la versión en Inglés (pulsa Ctrl+Intro para cambiar a pantalla completa):

```
$ wget http://www.zeldaroth.fr/us/files/\
  ROTH/Linux/ZeldaROTH_US-src-linux.zip
$ unzip ZeldaROTH_US-src-linux.zip
$ cd ZeldaROTH_US-src-linux/src
$ make -j5
$ ./ZeldaROTH_US
```

Más aventuras

Si has jugado a todos estos juegos y quieras más, sólo tienes que visitar la página de Vicente Jouillat en http://bit.ly/1LAhCI7 y descargar otros juegos, que se pueden compilar e instalar como los ejemplos que hemos visto. Así podrás acompañar a Link en otras aventuras como “On Link Begins,”, “Time to Triumph,” y “Navi’s Quest. Otro lugar para buscar es la página de inicio del equipo Solarus en http://bit.ly/1RTgUKv. Al menos dos juegos de la comunidad ya se pueden descargar en sus primeras versiones, y continúan trabajando en su desarrollo. Buena suerte en tu viaje, y ¡Divertirse!

JUMPER OTG ODROID-C1+

ALIMENTAR POR USB SIN SOLDADURAS

editado por Rob Roy

Se ha añadido un nuevo jumper a la placa ODROID-C1+ en la revisión del PCB del 30/09/2015 que permite activar la alimentación por OTG. Cuando se coloca el jumper, la placa puede alimentarse vía USB con un cable USB OTG. En visiones anteriores de la placa, esta opción requiere desoldar la conexión R94, como se describe en el post de http://bit.ly/1NBoyon.

El juamper J8 se puede retirar para conseguir un acceso estable a modo de dispositivo, como si fuera un conductor gadget o interfaz ADB/Fastboot. Para obtener más información, visita el artículo original en http://bit.ly/1XLeuUT.

Figura 1 – La Revisión 0.4 del ODROID-C1+ permite alimentar fácilmente la placa vía USB

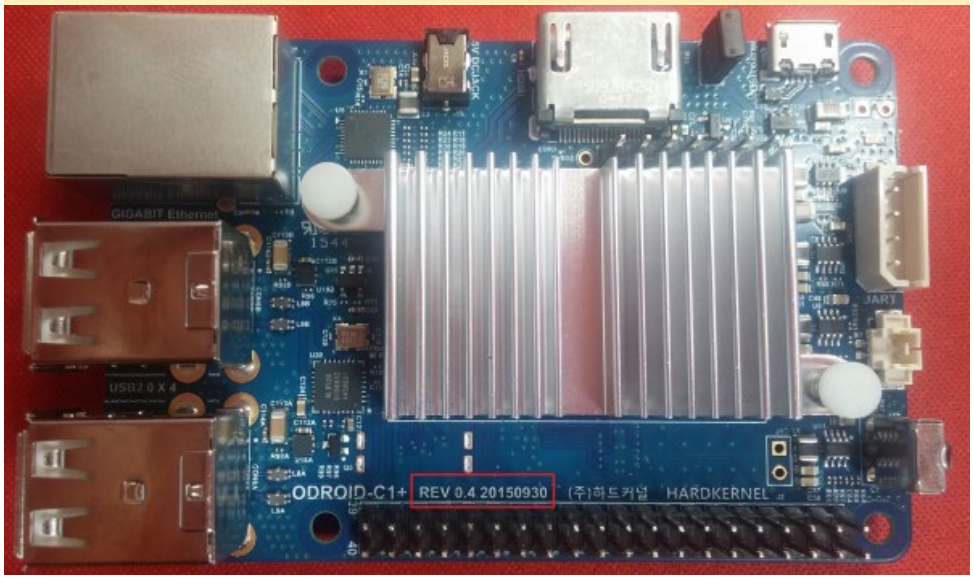
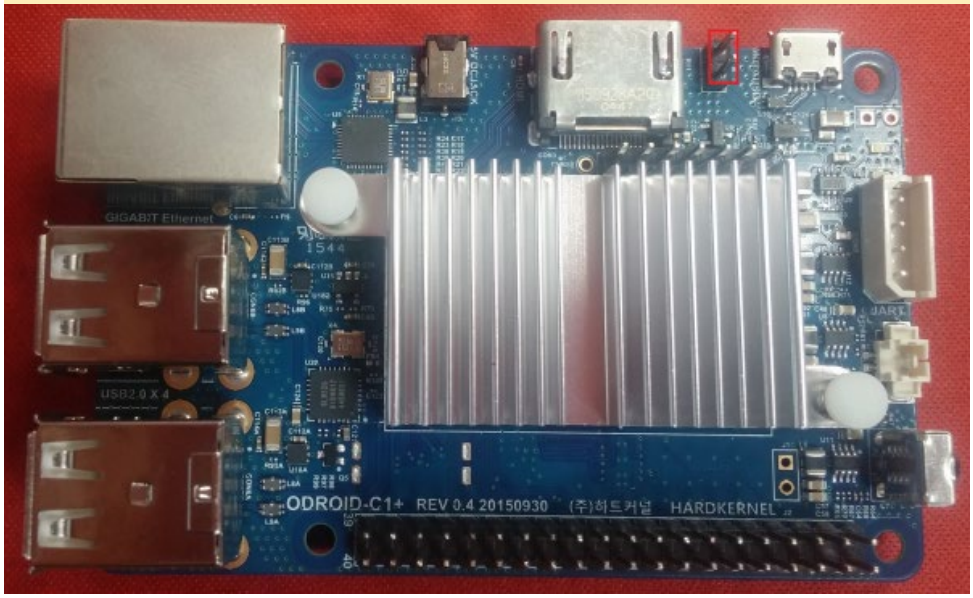


Figura 2 - Ubicación del jumper J8 en la última revisión del ODROID-C1+



DESARROLLO ANDROID

DENTRO DEL SERVIDOR DEL SISTEMA

por Nanik Tolaram

En este artículo, vamos a echar un vistazo al servidor del sistema Android. El servidor del sistema es la principal aplicación interna que se encarga del inicio de una serie de servicios, tales como los servicios del sistema de hardware necesarios que están disponibles en tu dispositivo, el servicio responsable de paquetes (que se encarga de la gestión de paquetes, como la actualización, la instalación y eliminación de archivos .apk), y muchas otras tareas importantes. Si el servidor del sistema falla, Android entrará en un bucle de arranque y volverá inservible tu dispositivo. Las diferentes versiones de Android tienen diferentes servicios, de modo que e este artículo me centraré en Kitkat 4.4.2 para ODROID-C1.

```
private static boolean startSystemServer()
        throws MethodAndArgsCaller, RuntimeException {
    long capabilities = posixCapabilitiesAsBits(
        OsConstants.CAP_KILL,
        OsConstants.CAP_NET_ADMIN,
        OsConstants.CAP_NET_BIND_SERVICE,
        OsConstants.CAP_NET_BROADCAST,
        OsConstants.CAP_NET_RAW,
        OsConstants.CAP_SYS_MODULE,
        OsConstants.CAP_SYS_NICE,
        OsConstants.CAP_SYS_RESOURCE,
        OsConstants.CAP_SYS_TIME,
        OsConstants.CAP_SYS_TTY_CONFIG
    );
    /* Hardcoded command line to start the system server */
    String args[] = {
        "--setuid=1000",
        "--setgid=1000",
        "--setgroups=1001,1002,1003,1004,1005,1006,1007,1008,1009,1010,1018,1032,3001,3002,3003,3006,3007",
        "--capabilities=" + capabilities + "," + capabilities,
        "--runtime-init",
        "--nice-name=system_server",
        "com.android.server.SystemServer",
    };
    ZygoteConnection.Arguments parsedArgs = null;

    int pid;

    try {
        parsedArgs = new ZygoteConnection.Arguments(args);
        ZygoteConnection.applyDebuggerSystemProperty(parsedArgs);
        ZygoteConnection.applyInvokeWithSystemProperty(parsedArgs);

        /* Request to fork the system server process */
        pid = Zygote.forkSystemServer(
                parsedArgs.uid, parsedArgs.gid,
                parsedArgs.gids,
                parsedArgs.debugFlags,
                null,
                parsedArgs.permittedCapabilities,
                parsedArgs.effectiveCapabilities);
    } catch (IllegalArgumentException ex) {
        throw new RuntimeException(ex);
    }
```

Figura I - ZygoteInit.java

Vida tras Zygotee

La aplicación que inicia el proceso de servidor del sistema se llama Zygote, que se activa durante el proceso init de arranque. Puede aprender más sobre Zygote en la edición de Enero de ODROID en http://bit.ly/1MOtlH5. El código reside dentro ZygoteInit.java, como se muestra en la Figura 1. Este fragmento de código en particular es responsable de arrancar el servidor del sistema.

Servicios Init

Como su nombre indica, la tarea principal del servidor de sistema es inicializar los diferentes servicios del sistema que se deben ejecutar y los pones a disposición. No iniciar estos servicios detendría todo el sistema Android, forzándolo a entrar en lo que se conoce como "bucle de arranque", una situación en la que ves la animación de arranque una y otra vez.

La siguiente lista muestra los servicios que hace accesibles el servidor del sistema, algunos son utilizados por aplicaciones de usuario. Puede utilizar la lista de servicios proporcionada por el comando adb para imprimir los servicios que han sido iniciados por el sistema.

```
0    sip: [android.net.sip.
ISipService]
1    phone: [com.android.inter-
nal.telephony.ITelephony]
2    iphonesubinfo: [com.android.
internal.telephony.IPhoneSubInfo]
3    simphonebook: [com.android.
internal.telephony.IIccPhoneBook]
4    isms: [com.android.internal.
telephony.ISms]
5    media_router: [android.me-
dia.IMediaRouterService]
6    print: [android.print.
IPrintManager]
7    assetatlas: [android.view.
IAssetAtlas]
8    dreams: [android.service.
dreams.IDreamManager]
9    commontime_management: []
10    samplingprofiler: []
11    diskstats: []
……
37    statusbar: [com.android.
internal.statusbar.IStatusBarSer-
vice]
38    device_policy: [android.
app.admin.IDevicePolicyManager]
39    lock_settings: [com.an-
droid.internal.widget.ILockSet-
tings]
40    mount: [IMountService]
……
50    alarm: [android.app.IAlarm-
Manager]
51    consumer_ir: [android.hard-
ware.IConsumerIrService]
52    vibrator: [android.
os.IVibratorService]
……
64    procstats: [com.android.
internal.app.IProcessStats]
65    activity: [android.app.IAc-
tivityManager]
66    package: [android.content.
pm.IPackageManager]
67    schedul-
ing_policy: [android.
os.ISchedulingPolicyService]
75    SurfaceFlinger: [android.
ui.IsurfaceComposer]
```

El número de servicios crece con cada versión de Android. En la siguiente lista, puedes ver los servicios que se ponen a disposición de las aplicaciones de usuario en KitKat (http://bit.ly/1X3EchG):

`WINDOW_SERVICE (window)`

El gestor de ventanas de nivel superior en el que puedes colocar ventanas personalizadas. El objeto devuelto es un WindowManager.

`LAYOUT_INFLATER_SERVICE (layout_inflater)`

Un LayoutInflater para inflar los recursos del formato en este contexto.

`ACTIVITY_SERVICE (activity)`

Un ActivityManager para interactuar con la actividad global del sistema.

`POWER_SERVICE (power)`
Un PowerManager para controlar la gestión de energía.
`ALARM_SERVICE (alarm)`
Un AlarmManager para recibir los objetivos en el momento de su elección.
`NOTIFICATION_SERVICE (notification)`
Un NotificationManager para informar al usuario de los eventos de fondo.
`KEYGUARD_SERVICE (keyguard)`
Un KeyguardManager para controlar bloqueo de teclas.
`LOCATION_SERVICE (location)`
Un LocationManager para controlar las actualizaciones de la ubicación (por ejemplo, GPS).
`SEARCH_SERVICE (search)`
Un SearchManager para gestionar las búsquedas.
`VIBRATOR_SERVICE (vibrator)`
Un vibrador para interactuar con el hardware vibrador.
`CONNECTIVITY_SERVICE (connection)`
Un ConnectivityManager para gestionar de las conexiones de red.
`WIFI_SERVICE (wifi)`
Un WifiManager para la gestión de las conexiones Wi-Fi.
`WIFI_P2P_SERVICE (wifip2p)`

Figura 2 - Flujo de Código que inicia el servidor del sistema

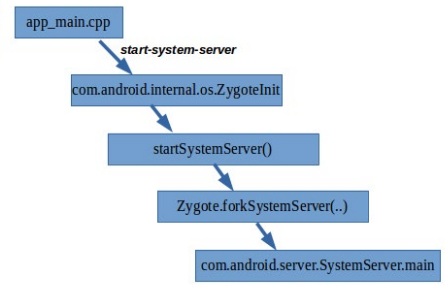
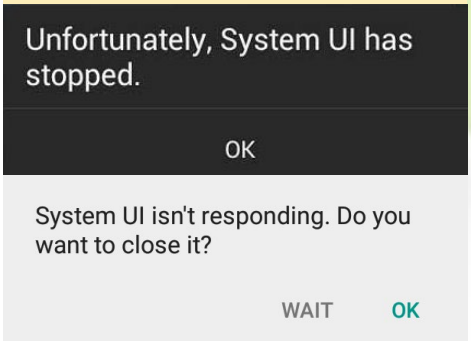


Figura 3 - Mensaje de error y diálogo de confirmación cuando se bloquea la interfaz de usuario del sistema



Un WifiP2pManager para la gestión de las conexiones Wi-Fi Direct.
`INPUT_METHOD_SERVICE (input_method)`
Un InputMethodManager para gestionar los métodos de entrada
`UI_MODE_SERVICE (uimode)`
Un UiModeManager para controlar los modos de interfaz de usuario.
`DOWNLOAD_SERVICE (download)`
Un DownloadManager para solicitar descargas HTTP
`BATTERY_SERVICE (batterymanager)`
A BatteryManager para gestionar el estado de la batería
`JOB_SCHEDULER_SERVICE (taskmanager)`
Un JobScheduler para gestionar las tareas programadas
`NETWORK_STATS_SERVICE (netstats)`
Un NetworkStatsManager para consultar las estadísticas de uso de la red.

¿Te has dado cuenta que la segunda lista muestra un menor número de servicios que la primera lista? La razón es que SDK sólo proporciona servicios que son útiles para desarrolladores con el objeto de que desarrollen las aplicaciones Android que quieran. Sin embargo, como Android es de código abierto, puedes crear tu propio SDK y presentar otros servicios a tu desarrollador de aplicaciones para usarlos en tu dispositivo. La Figura 2 muestra el flujo de código que al final inicia el servidor del sistema.

Vamos a ver dos servicios básicos que son cruciales para los desarrolladores de Android. Un fallo en estos servicios hará que la aplicación deje de funcionar.

Gestor de Actividad

Este es uno de los servicios cruciales del que depende la mayoría de los desarrolladores Android. La ejecución de aplicaciones a través del ciclo de vida de actividad es controlada dentro de este servicio. La creación, reactivación, destrucción y otras operaciones relacionadas con la actividad se ejecutan aquí. La forma normal de obtener este servicio desde la aplicación Android es usar la siguiente llamada API:

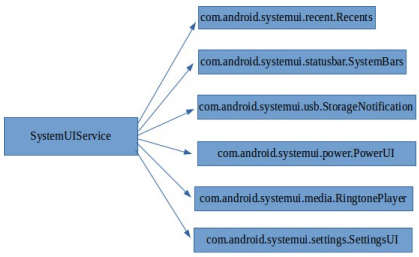


Figura 4 - Los servicios que son lanzados cuando se ejecuta la interfaz de usuario del sistema

```
ActivityManager am =
(ActivityManager)getActivity().
getSystemService(
Context.ACTIVITY_SERVICE);
```

Usando el Gestor de Actividad, puedes acceder a información como la memoria y los procesos actualmente en ejecución. Encontraras más información sobre el Gestor de Actividad en http://bit.ly/1N80KhR. Internamente, este servicio almacena información de todas las aplicaciones Android que se estén ejecutando en el momento, como la seguridad, los permisos, la información del proceso (nombre, fecha y tamaño), información de la memoria y mucho más.

IU del Sistema

Este servicio no está disponible para la capa de aplicación, ya que es un servicio interno que es importante para el entrono de trabajo de Android. Si llevas utilizando Android desde hace algún tiempo, es posible que hayas visto el error que se muestra en la Figura 3.

Cuando aparece este tipo de error, Android entra en un estado de "reinicio Suave", donde todo el entorno de trabajo será reiniciado tan pronto como se pulse el botón OK. Como su nombre indica, la IU del Sistema realiza muchas operaciones críticas, mientras que proporciona los servicios necesarios para la interacción con el usuario, como la notificación, el encendido y el volumen. Internamente, la IU del sistema es el resultado de una gran variedad de servicios que trabajan en armonía. La figura 4 muestra algunos de los servicios que se ponen en marcha cuando se ejecuta.

CONOCIENDO UN ODROIDIAN

SALEEM ALMAJED (@XEOSAL) EXPERTO EN TECNOLOGIAS EMERGENTES Y APASIONADO DE LA MUSICA

editado por Rob Roy

Por favor, hablanos un poco sobre ti.

Mi nombre es Saleem Almajed. Tengo 22 años, y vivo en un pequeño país llamado Bahrein. Siempre he sido un enamorado de la tecnología. Me gusta todo lo relacionado con los ordenadores, teléfonos inteligentes y sistemas operativos.

¿Cómo fueron tus inicios con los ordenadores?

Mi profesor de primaria, mientras me enseñaba conocimientos básicos de informática, vio que tenía talento para ser algo en el futuro. Insistió a mi madre en que tenía mucho potencial para ser un experto en TI, y le aconsejo que me consiguiera un PC. Así lo hizo y empecé a jugar con los ordenadores desde muy pequeño. Conseguí mi primer sistema operativo Linux, después usar durante dos años Windows ME, cuando localice un artículo que decía que podía pedir un CD de instalación de Ubuntu de forma gratuita, gracias a Canonical.

Aunque parezca que se está divirtiendo, Saleem está realmente pensando en todos los proyectos que se puede desarrollar con su CI

Saleem tiene mucho talento en lo que respecta a la electrónica

¿Qué te atrajo de la plataforma ODROID?

Me introduje en el mundo de la informática integrada ARM a través de la Raspberry Pi. Tras usarla durante aproximadamente un mes, decidí buscar alternativas y fue entonces cuando descubrí el ODROID-C1. Resultó ser mucho mejor, especialmente por el doble bus USB 2.0, la GPU Malí, mejor CPU y RAM. También me llamo la atención que había otros ODROIDs más potentes. Tuve la oportunidad de aprender mucho sobre estos pequeños ordenadores y sus especificaciones usándolos y explorando los foros ODROID, y fue entonces cuando decidí apoyar a HardKernel.

¿Cómo utilizas tus ODROIDs?

Los utilizo sobre todo como sustituto del PC de escritorio con fines de desarrollo del kernel, sistema operativo y software. También los uso como centro multimedia, servidores locales y emuladores retro para mi hermano menor.

¿Cuál es tu ODROID favorito?

El ODROID-C1.

Saleem disfruta tocando la guitarra cuando no está programando

Tu imagen Odrobian Jessie precompilada para la comunidad es muy popular. ¿Qué te motivó a crear la imagen?

Toda mi vida, he estado leyendo y aprendiendo de los expertos en Internet, y últimamente me he estado preguntando si me había vuelto lo suficientemente bueno como para ser el que yo ayudase a otras personas. Simplemente quería creer en mí mismo creando algo que pudiera utilizar realmente la gente.

¿Qué novedades te gustaría ver en futuros productos de Hardkernel?

Me gustaría que HardKernel creara un SOC de desarrollo hecho a medida para sus ODROIDs. Estoy seguro de que abriría un nuevo mundo de posibilidades. Mi sueño es hacer que los ordenadores ARM integrados lleguen a ser tan personalizables como sea posible. También me gustaría ver RAM actualizable y PCI-E en futuros modelos.

¿Qué aficiones e intereses tienes aparte de los ordenadores?

Me gusta tocar la guitarra, escuchar música, cantar y escribir canciones, y leer acerca de todo lo vanguardista, como la ciencia, la tecnología y los gadgets. También me gusta ver películas y desarrollar proyectos de bricolaje, como amplificadores y otros aparatos electrónicos.

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

Siempre y cuando creas en ti mismo, puedes hacer cualquier cosa. Establécete metas y empieza a trabajar, te sorprenderás de lo lejos que puedes llegar después de tan sólo un par de meses.

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 deseas 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, por favor visite el hilo del foro original ent `http://bit.ly/1QDMNoT`.